

A
REPORT
Of
MINI PROJECT OR INTERNSHIP ASSESSMENT
At
“TECHVISION TECHNOLOGIES”



On
“Full Stack Development using MERN”
In partial fulfilment of Bachelor of Technology
Department of Computer Science and Engineering



**Shri Ram Murti Smarak College of Engineering & Technology,
Bareilly**

Session: 2024-25

Submitted by:

Name: Robin Singh Yadav

Roll No.: 2100140100075

Submitted to:

Dr. Pradeep Kumar Sharma

(Assistant Professor)

DECLARATION

I, Robin Singh Yadav, hereby, declare that the Project Report submitted to Shri Ram Murti Smarak College of Engineering and Technology, Bareilly in partial fulfilment of requirements for Industrial Training (Internship Assessment), was done under the guidance of A.K. Singh, Techvision Technologies Bareilly U.P India

I also declare that the information in this report is correct as per my knowledge and I bear any responsibilities for any error or omission, if any. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Robin Singh Yadav

Date: 02/09/2024

ACKNOWLEDGEMENT

I am deeply grateful to Techvision Technologies, Bareilly, for providing me with the opportunity to complete my training under their esteemed guidance. This experience has been invaluable, and I would like to extend my heartfelt thanks to the entire team at Techvision Technologies for their support and encouragement throughout the training period.

I would also like to express my sincere gratitude to A.K. Singh, who guided me and my fellow trainees with great patience and expertise. Your mentorship and insights have significantly enriched my learning experience, and I am thankful for the time and effort you invested in our growth.

Furthermore, I would like to extend my appreciation to our Head of Department (CSE), Dr. Shahjahan Ali, and the Dean of our college, Dr. Prabhakar Gupta, for their continuous support and for providing us with the necessary resources and encouragement to pursue this training. Their guidance has been instrumental in shaping my academic journey, and I am grateful for their unwavering support.

Finally, I would like to thank my family and friends for their constant encouragement, which has been a source of strength throughout this journey.

TABLE OF CONTENTS

DECLARATION	iii
ACKNOWLEDGEMENT	iv
LIST OF FIGURES	2
1. COMPANY PROFILE	1
1.1 INTRODUCTION	1
1.2 VISION & MISSION.....	1
1.3 TRAINING APPROACH & METHODOLOGY	1
2. INDUSTRY AT A GLANCE.....	3
2.1. TRAINING & DEVELOPMENT DEPARTMENT	3
2.2 BEST COLLEGE CAMPUS TRAINING	3
2.3 FRESHER TRAINING	4
2.4 INDUSTRIAL TRAINING.....	5
3. TOOL AND TECHNOLOGY USED IN INDUSTRY	6
4. PROJECT UNDERTAKEN	23
4.1 BACKGROUND OF THE STUDY	23
4.2 OBJECTIVE	25
4.3 METHODOLOGY	26
5. SYSTEM DESIGN	28
5.1 SYSTEM ARCHITECTURE	28
5.2 USE CASE DIAGRAM.....	31
5.3 ENTITY-RELATIONSHIP DIAGRAM.....	32
6. IMPLEMENTATION	34
6.1 DEVELOPMENT ENVIRONMENT	34
6.2 TOOLS AND TECHNOLOGIES USED.....	34
7. SNAPSHOTS OF THE PROJECT	36
8. LESSONS LEARNED	41
9. FUTURE SCOPE	44
10. CONCLUSION	45
REFERNCES	46
APPENDIX	47

LIST OF FIGURES

Figure 1 Mongo DB Atlas Logo.....	6
Figure 2 VS Code Logo	7
Figure 3 Postman Logo	7
Figure 4 Nodemon Logo	8
Figure 5 MERN stack Logo	9
Figure 6 MERN Architecture	11
Figure 7 Server	12
Figure 8 DataBase Connectivity	14
Figure 9 React Example	19
Figure 10 Node Logo	21
Figure 11 Use Case Diagram	31
Figure 12 ER Diagram	33
Figure 13 Product Description Page.....	37
Figure 14 Product Review Section	38
Figure 15 Searched Product	38
Figure 16 Cart Item Page.....	39
Figure 17 Login Page	39
Figure 18 Sign Up Page	39
Figure 19 Admin Panel (Add Product)	39
Figure 20 Admin Panel (Product List)	40
Figure 21 App.js.....	47
Figure 22 LoginSignUp.jsx	47
Figure 23 Product.jsx	48
Figure 24 Navbar.jsx	48
Figure 25 Item.jsx	49
Figure 26 ShopContext.jsx	49
Figure 27 Main.jsx for Admin Panel.....	50
Figure 28 App.jsx for Admin Panel	50
Figure 29 AddProduct.jsx for Admin Panel.....	51
Figure 30 Admin.jsx.....	51

Figure 31ListProduct.jsx for Admin Pannel	52
Figure 32 NavBar.jsx for Admin Pannel.....	52
Figure 33 SideBar.jsx for Admin Pannel	53
Figure 34 Index.js (Backend)	54
Figure 35 User Schema (Backend).....	54

1. COMPANY PROFILE

1.1 INTRODUCTION

Located in Bareilly and established in 2008, we are a premier institute dedicated to empowering individuals through cutting-edge training in various disciplines. Whether you're looking to enhance your skills in Computer Science, IT, Office Automation, Data Science, Artificial Intelligence, Machine Learning, Robotics, Blockchain, Graphic Design, or Animation, we have the perfect course tailored to meet your needs.

1.2 VISION & MISSION

Our vision at Techvision Technologies is to emerge as the foremost IT training company, delivering unparalleled education, and training to professionals worldwide. We aim to redefine the landscape of IT training, perpetually innovating across various courses, learning modes, and delivery methodologies.

At the core of our mission is the creation of a dynamic community comprising avid learners, seasoned trainers, and industry experts who share an unwavering passion for their careers. We are dedicated to arming our students with the expertise and knowledge required to emerge as industry trailblazers, propelling innovation and fostering growth in the IT sector.

Our overarching mission is to empower professionals with a tailored blend of skills and knowledge essential for their career triumphs. We are steadfast in our commitment to:

- Providing cutting-edge training and development programs meticulously tailored to cater to the distinctive needs of our learners.
- Furnishing practical skills and immersive hands-on experiences that empower our students with the unwavering confidence to confront real-world challenges head-on.

1.3 TRAINING APPROACH & METHODOLOGY

Techvision Technologies champions a student-centric approach underpinned by a tailored methodology for its training programs. Here are some of the hallmark features of our training approach and methodology:

- a) Diverse Training Programs: We offer an array of training programs designed to cater to varying levels of students. These encompass 2/3/4/6-week training modules, industrial training, winter training, college campus training, and more, ensuring that each student's unique needs are met.
- b) Extensive Technology Coverage: Our comprehensive training curriculum encompasses over 50 leading technologies and software programs, spanning diverse domains such as cloud computing, Java, MERN, Data Science, Artificial Intelligence and many more .
- c) Practical Learning: Our training sessions are characterized by hands-on practical sessions, live project experiences, interactive workshops, and informative seminars. These elements collectively enhance the practical skills and knowledge base of our students, fostering a holistic learning experience.

2. INDUSTRY AT A GLANCE

Techvision Technologies is a well-regarded company specializing in engineering training for both students and professionals. In addition to their educational services, they excel in offering comprehensive solutions, including software development, mobile application development, digital marketing, mechanical design, and advanced chip development services. Notably, they have a track record of successfully placing students in top multinational corporations (MNCs) within a year, underscoring their commitment to talent development and career advancement.

2.1. TRAINING & DEVELOPMENT DEPARTMENT

Techvision Technologies Training & Development Department excels in providing real-time training in cutting-edge engineering domains like AWS, Digital Marketing, DevOps, and Full stack development. Our diverse training programs, including summer, winter, regular, and corporate options, enhance both technical and soft skills. We also offer comprehensive placement assistance to help trainees secure positions in top-tier IT companies. Techvision Technology, are dedicated to empowering individuals for success in the ever-evolving tech industry.

2.2 BEST COLLEGE CAMPUS TRAINING

The College Campus Training Program, an integral initiative within Techvision Technology, represents a comprehensive endeavour aimed at equipping engineering students with cutting-edge technological knowledge within the confines of their college campuses. This program caters to a diverse array of students, including those pursuing B.Tech/M.Tech, B.Sc/M.Sc, BCA/MCA degrees, and more, with a deep-seated recognition of the paramount importance of practical training over theoretical knowledge in the demanding corporate landscape. Techvision Technology, an eminent Technology Promotion Association, extends its influential presence across key Northern Indian cities, including Noida, Roorkee, Lucknow, and Dehradun.

At Techvision Technology, our College Campus Training is meticulously architected to elevate students' "industry readiness," making them exceptionally employable upon their academic journey's culmination. This program envelops an extensive spectrum of technological domains and

is made accessible at a reasonable fee structure. During the training immersion, students are the beneficiaries of a harmonious blend of theoretical acumen and hands-on experiential learning, often culminating in the completion of a substantive project of significance. To facilitate this holistic learning odyssey, we provide comprehensive study materials and state-of-the-art lab resources. Following the triumphant completion of the program, students are bestowed with globally recognized certificates of accomplishment.

The on-campus training program confers multifaceted benefits upon both the students and college authorities alike. It optimally allocates students' financial resources by providing cost-effective training avenues vis-a-vis traditional off-campus alternatives. This remarkable flexibility empowers students to embark on 2-4-week training journeys, thus ensuring the comprehensive coverage of academic syllabus content while meticulously adhering to the exalted quality standards for which Techvision Technology is celebrated.

In essence, on-campus training transcends a mere conceptual premise, serving as an immersive portal into the professional realm, where practical skills and applied knowledge reign supreme. These distinguished courses, conducted within the academic bastions themselves, feature the guidance and expertise of dedicated professionals from Techvision Technology, who bestow knowledge and inspiration through intensive five to six-hour daily sessions. This training regimen encompasses not only technical proficiencies but also the crucial soft skills that empower students for a triumphant transition into the dynamic and competitive corporate sphere. Our fervent mentors, frequently emerging engineers from the industry's vanguard, sustain a culture of engagement and unwavering focus, thereby ensuring a profoundly transformative learning experience.

2.3 FRESHER TRAINING

If you are a recent graduate or a final-year student eager to embark on an enriching journey of long-term industrial training, summer training, winter training, or a concise 6-week training program, Techvision Technology stands as the ideal choice for you. SSDN is a premier training company renowned for its commitment to nurturing engineering students in various northern cities

of India, including Roorkee, Noida, Lucknow, and Dehradun. What sets SSDN apart is its assembly of highly proficient and promising experts who employ an elite training methodology, solidifying Techvision Technology's reputation as the preeminent summer training institute in the Delhi NCR region.

Our fresher training programs at Techvision Technology are characterized by an immersive, hands-on approach. These programs hold immense value for scholars who have recently completed their degree programs. Fresher training serves as a bridge between academic knowledge and real-world industry applications, which is of paramount importance for engineering students. Techvision Technology tailors fresher training across a wide spectrum of engineering disciplines, equipping students with essential employability skills that complement their academic coursework. We understand the significance of this transitional phase, and our training is meticulously designed to empower students for successful career integration.

2.4 INDUSTRIAL TRAINING

Embark on a Transformative Journey with Techvision Technology's 6-Month Industrial Training Program in Gurgaon. Techvision Technology, a paragon of excellence in industrial training, offers a comprehensive learning experience tailored to the unique needs of aspiring CS/CSE/IT/BCA/MCA/B.Tech/M.Tech/B.Sc./M.Sc./BE/Engineering students.

The compelling rationale underpinning the indispensability of our six-month industrial training program for budding engineers lies in its ability to meticulously groom them for the dynamic demands of the corporate world. This immersive training regimen serves as a conduit, introducing students to the intricate tapestry of real-world industry operations. It fosters unwavering self-assurance while unearthing and honing their innate strengths. Beyond bolstering self-confidence, it nurtures leadership acumen and a profound sense of responsibility, ensuring adeptness in the execution of tasks. Furthermore, it serves as an academic bridge, connecting theoretical concepts to tangible, hands-on applications within an authentic job environment.

At Techvision Technology, an unrivalled Industrial Training Centre catering to the diverse spectrum of B.Tech/CS/CSE/IT/BCA/MCA/B.E/M.Tech/B.Sc/M.Sc students, we have set the bar for excellence. Our program seamlessly interlaces cutting-edge theoretical knowledge with practical hands-on experiences in the latest technologies.

3. TOOL AND TECHNOLOGY USED IN INDUSTRY

3.1. MongoDB Atlas

- a) MongoDB Atlas, which launched in June of 2016, is a battle-tested database-as-a-service platform (DBaaS) artfully designed and built by the same team that created and continues to nurture and grow MongoDB. MongoDB Atlas is a true blessing to the development community; it provides all of the features of its database counterpart, without the operation and heavy lifting normally required when building new applications, letting you focus on what you do best.
- b) MongoDB Atlas is a multi-cloud database service by the same people who build MongoDB. Atlas simplifies deploying and managing your databases while offering the versatility you need to build resilient and performant global applications on the cloud providers of your choice.



Figure 1 MongoDB Atlas Logo

3.2. Visual Studio Code

Visual Studio Code is a free, open-source, and lightweight code editor that offers a range of features. It also provides support for a wide range of extensions, making it easier to customize your development environment.



Figure 2 VS Code Logo

Key Features:

- a. IntelliSense provides intelligent suggestions for code completion, parameter hints, and more, making coding faster and more efficient.
- b. Debugging support helps you find and fix errors quickly, with features like breakpoints, call stacks, and console output.
- c. Git integration allows you to manage your code changes directly within the editor, with features like branch switching, commit history, and diff checking.

3.2. Postman

Postman is a popular API development tool that allows you to test and debug your API endpoints. According to a survey conducted by Postman in 2021, 86% of developers agree that using Postman has helped them reduce development time.



Figure 3 Postman Logo

Key Features:

- a. Automated testing allows you to create and run tests for your API endpoints, with support for assertions, data-driven testing, and more.
- b. Request history provides a log of all requests made, making tracking changes and troubleshooting issues easier.
- c. Environment variables allow you to store and manage environment-specific values, such as API keys or authentication tokens, to simplify testing across different environments.

3.3. Nodemon

Nodemon is a utility tool that automatically restarts your Node.js application when changes are detected. It helps you save time by eliminating the need to manually restart the server whenever you make code changes.



Figure 4 Nodemon Logo

Key Features:

- a. Automatic restarts of your Node.js application when changes are detected, saving you time and improving your development workflow.
- b. Monitors specified directories and files for changes, allowing you to focus on coding without worrying about server restarts.

3.2. MERN Stack

MERN Stack is a collection of powerful technologies and robust, used to develop scalable master web applications comprising backend, front-end, and database components. It is JavaScript that is used for the faster and easier development of full-stack web applications. MERN Stack is a technology that is a user-friendly full-stack JavaScript framework for building applications and dynamic websites.

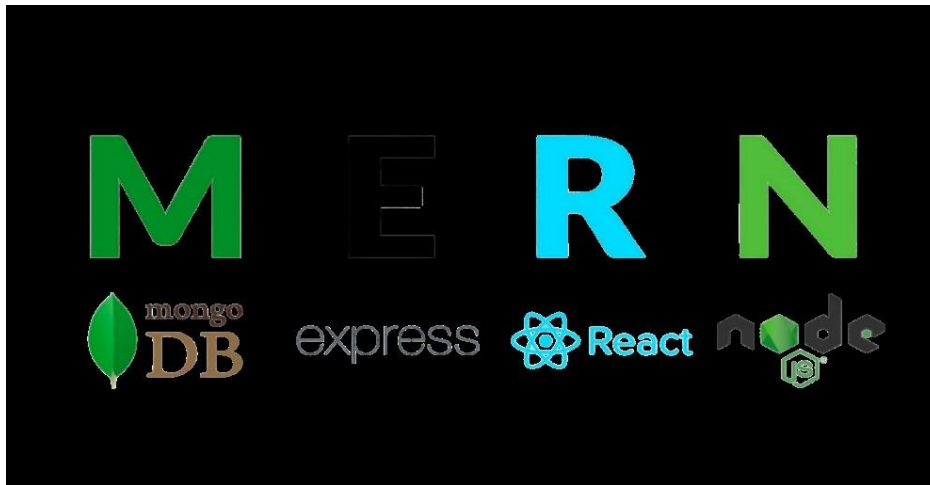


Figure 5 MERN stack Logo

MERN Stack consists of four main components or say four main technologies:

- a) **M** stands for MongoDB (Database), mainly used for preparing document databases and is a NoSQL (Non-Structured Query Language) Database System
- b) **E** stands for Express, mainly used for developing Node.js web framework
- c) **R** stands for React, mainly used for developing a client-side JavaScript framework
- d) **N** stands for JS, mainly used for developing the premier JavaScript web server

Each of these four technologies plays an important role in providing an end-to-end framework for the developers. Even these four technologies play an important role in the development process of web applications.

Before MERN stack, it was earlier named as MEAN stack, MERN Stack is one of the variations of MEAN, here MEAN also comprises four components or say four different technologies, i.e., M is for MongoDB, 'E' is for Express, 'A' is for Angular.js, and 'N' is for Node, here if you will observe,

then you can identify that in MEAN, ' A ', i.e., Angular.js is replaced by ' R ', i.e.,

React.js in MERN, the main reason behind is -MERN Stack is mainly used for faster development of smaller applications as compared with MEAN, MEAN stack is a mainly better option for large-scale applications. Still, it takes more time for the development of smaller applications. They also both have different structures comparatively. We can see that MEAN relies on Angular.js for its front-end JavaScript framework, whereas the MERN stack relies on React and its ecosystem.

MERN Stack is designed to make the development process easier and smoother. From the above figure, if we notice, then we can see that Node and Express make up the middle application tier. Node.js very powerful and popular JavaScript server platform and Express.js is a server-side web framework. Unless of which variant you choose, ME(RVA)N is an ideal approach to working with JSON and JavaScript all the way through.

MERN Stack for building Web applications:

- a) **Cost-effective:** All the four technologies that are mentioned above, MERN (MongoDB, Express.js, React.js, and Node.js) are used in MERN Stack is built on JavaScript makes it cost-effective and with less cost investment, the user will get better results or output.
- b) **SEO friendly:** Here, SEO (Search Engine Optimization) friendly means that Google, Yahoo, and other search engines can search each page on the website efficiently and easily, interpret and correlate the content effectively with the searched text, and easily index it in their database. Whenever websites are created using MERN technologies, they are always SEO-friendly.
- c) **Better performance:** Better performance refers to the faster response between the backend and front-end and database, which ultimately improves the website speed and yields better performance, thus providing a smooth user experience.
- d) **Improves Security:** It mainly concerns the security of applications generated using MERN; web application security refers to various processes, methods, or technologies used for protecting web servers and various web applications, such as APIs (Application user interface) from the attack by internet-based threats. Generally, secured hosting providers can easily

integrate applications created using the MERN stack. MongoDB and Node.js security tools are also used for more or better security.

- e) **Provides faster Modifications:** MERN stack technologies support quick modifications as per the client's request in the mobile and web applications.
- f) **Open Source:** All four technologies that are involved in MERN are open-source. This feature allows developers to get solutions to queries that may evolve from the open portals during development. As a result, it will be ultimately beneficial for a developer.
- g) **Easy to switch between client and server:** MERN is very simple and fast because it is written in only one language. Also, it is very easy to switch between client and server.

Architectural Structure of MERN Stack and its working

MERN has a 3-tier Architecture system mainly consisting of 3 layers - These layers are as follows:

- h) Web as front-end tier
- i) Server as the middle tier
- j) Database as backend tier

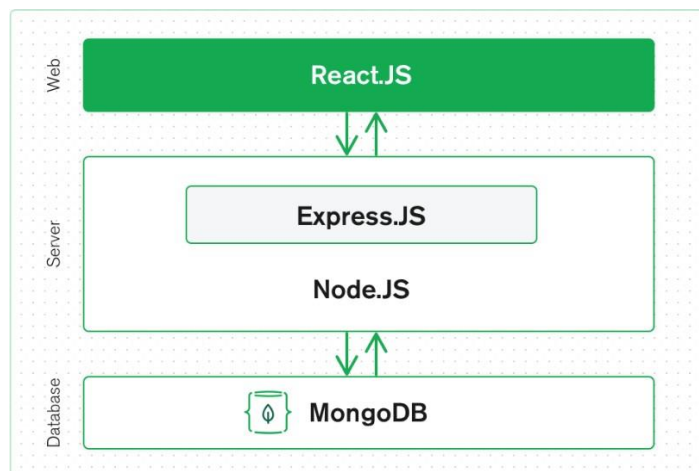


Figure 6 MERN Architecture

We already know that it comprises 4 components, i.e., MongoDB, Express.js, React, and Node.js.

Now let us understand in more detail about these three tiers which are mentioned above –

i. Web or front-end tier -The top tier of the MERN stack is mainly handled by React.js. It is one of the most prominent open-source front-end JavaScript libraries used for building Web applications.

It is famous for creating dynamic client-side applications. React will help you construct complex interfaces by using single components. It also connects those complex interfaces to data available on the backend server. React is used to create mobile applications (React Native) and web applications.

React allows the reusability of code and can easily support it, which has many benefits and is much time saver. It permits users to create large web applications that can easily change the data of the page even without reloading the page.

ii. Server or middle-tier - It is just the next level from the top layer and is mainly handled by two components of the MERN stack, i.e., Express.js and Node.js. These two's components handle it simultaneously because Express.js maintains the Server-side framework, running inside the Node.js server. Express.js is one of the widely used backend development JavaScript Frameworks. It allows developers to spin up robust APIs (Application Programming Interface) and web servers much easier and simpler. It also adds helpful functionalities to Node.js HTTP (**Hypertext Transfer Protocol**) objects. On the other hand, Node.js plays a very important role in itself. It is an open-source server environment, and it is a cross-platform runtime environment for executing JavaScript code outside a browser. Node.js continuously uses JavaScript; thus, it's ultimately helpful for a computer user to quickly create any net service or any net or mobile application.

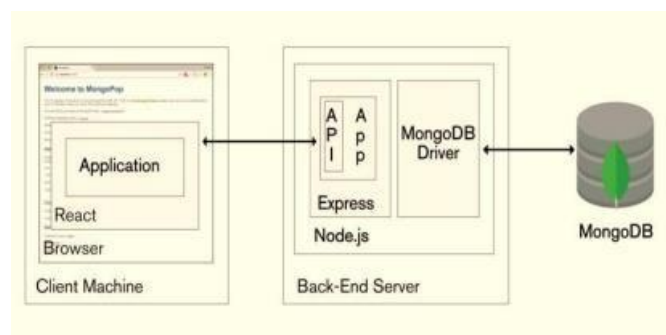


Figure 7Server

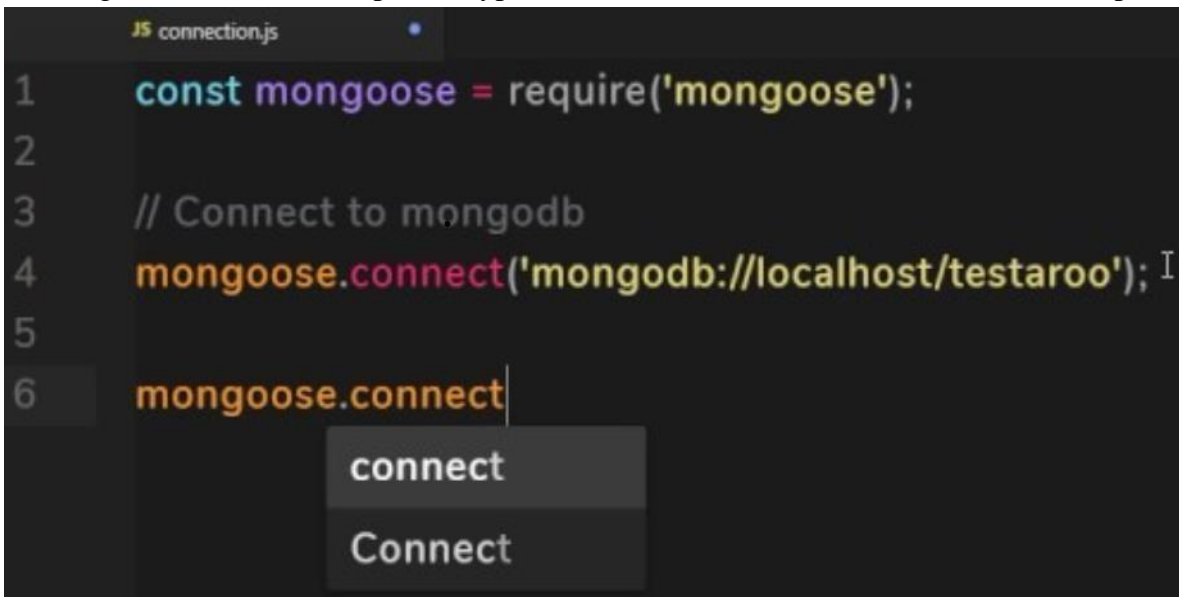
iii. Database as backend tier - It is one of the most important levels of the MERN Stack and is mainly handled by MongoDB; the main role of a database is to store all the data related to your application, for example - content, statistics, information, user profiles, comments and so on.

It mainly stores all the data for safety purposes. It maintains a proper record, which usually returns the data to the user whenever required. It mainly stores the data in the database. It generates two or more replica files of the data so that whenever the system fails, it can retrieve the exact information or data that the user wanted earlier. It implies that MongoDB is not based on the table-like relational database structure. On the other hand, it provides an altogether different mechanism for the retrieval and storage of data. Mongo DB is the most popular NoSQL (NoSQL or Non-Structured Query Language) database, an open-source document-oriented database. The term 'NoSQL' typically means a non-relational database that does not require a fixed schema or proper relational tables to store the necessary data in it. MongoDB stores th data in a different format other than the relational tables, consisting of rows and columns.

3.2.1 MongoDB:

- MongoDB is the most popular NoSQL (NoSQL or Non-Structured Query Language) database, an open-source document-oriented database.
- The term 'NoSQL' typically means a non-relational database that does not require a fixed schema or proper relational tables to store the necessary data in it. MongoDB stores the data in a different format other than the relational tables, consisting of rows and columns.
- It implies that MongoDB is not based on the table-like relational database structure. On the other hand, it provides an altogether different mechanism for the retrieval and storage of data.
- The storage format in which the data is stored is known as BSON, which stands for Binary JavaScript Object Notation; its binary structure encodes length and type of information, which allows it to be parsed much more quickly.
- MongoDB uses BSON when storing documents in collections.
- It allows a highly scalable and flexible document structure.
- It is much faster as compared to RDBMS due to its efficient storage and indexing techniques.

- In MongoDB, complex join operations are not available; hence, it cannot support complex transactions.
- MongoDB uses JavaScript for coding as a language which is one of its great advantages.
- It is schema as any data stored is stored in a separate document.
- In MongoDB, there is no concept of relationships or table formations, as this is happening in RDBMS (Relational Database Management System), in which tables have a certain relation between them.
- It also supports a flexible document model, which is very fast for any developer to create.
- MongoDB is one of the important types of NoSQL Databases. It is more scalable and provides



```

1  const mongoose = require('mongoose');
2
3  // Connect to mongodb
4  mongoose.connect('mongodb://localhost/testaroo');
5
6  mongoose.connect

```

excellent performance if we notice that it will reach its scaling limit whenever a database runs on a single server.

- MongoDB is a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

Some important features of MongoDB

1. **Schema-less Database:** MongoDB has one of the great features, meaning that one collection can hold different types of documents. Due to this extraordinary feature, MongoDB provides great flexibility to databases. In the MongoDB database, a single collection comprises multiple documents,

Figure 8 DataBase Connectivity

and these documents may further comprise different numbers of values, fields, and so on. One document doesn't need to be a must to relate with the other documents, as it happens in relational databases.

2. **Indexing:** One can easily fetch the necessary data from the data pool in the MongoDB database due to this indexing feature. In MongoDB, every data item has provided a particular index, categorized as primary and secondary indices. With this indexing, data retrieval is easier for the user; it saves a lot of time. If the data is not indexed, the database searches each document with the specified query, which takes lots of time and is inefficient.

3. **Document Oriented:** In MongoDB, all the data is stored in documents instead of tables like SQL. Also, these documents have their unique object ID. In these documents, the informative data is stored in fields, i.e., key-value pairs instead of columns and rows, making the data much more flexible and easier to fetch out rather than applying queries for every data compared to RDBMS.

4. **Faster:** MongoDB is very fast compared with relational database (RDBMS), which is document-oriented. Each data item has its index value, making retrieving any data easier without wasting time writing queries and making logic accordingly.

5. **Scalability:** MongoDB is more scalable with the help of sharing. It provides horizontal scalability. Here the term sharing means distributing data on multiple servers; in this, a large amount of data has been divided into multiple small data chunks with the help of a shard key. These types of data chunks are evenly distributed across shards that reside across many physical servers.

6. **High Performance:** MongoDB has very high performance and data persistency as compared to other databases due to the presence of its great features like indexing, scalability, replication, etc.

7. **Replication and Highly Available:** MongoDB increases the availability of data by creating multiple copies of data on different servers. Providing redundancy or data replication ultimately protects the database from any hardware failure and protects the data from being lost in the future. I suppose if one server was not working or clashed due to an error, and then data can easily be retrieved from other active servers, who are currently working at that time, this will all be due to redundancy

of data.

8. **Aggregation:** This feature of MongoDB is quite similar to the SQL GROUPBY clause. This GROUP BY clause performs various operations on the grouped data to get the unique or computed.

9. **Simple Environment Setup:** MongoDB has a very simple environment setup. One can easily set up MongoDB in their system without applying much effort.

3.2.2 Express:

- Express is a JavaScript server-side framework that runs within JS.
- It is one of the best backend development JavaScript Frameworks.
- It provides the developer with a platform to create and maintain robust servers.
- Express is used for building and designing web and mobile applications easily and quickly.
- Express is used to provide server-side logic for mobile and web applications, and as such, it is used all over the place.
- It allows developers to spin up robust APIs (Application Programming Interface) and web servers are much easier and simpler.
- Express makes robust web servers easier to organize your application's functionality with routing and middleware.
- It also adds helpful functionalities to Node.js HTTP (HyperText Transfer Protocol) objects.
- It is an important component of the MERN and MEAN Stack and is used to build fast, maintainable, and robust production web applications.

Some important features of Express:

1. Express makes Node.js web and mobile application development much easier and faster.
2. Express has a very simple environment setup. One can easily set up Express in their system and configure it without applying much effort.
3. Express is very easy to connect with Databases like MongoDB.
4. Based on HTTP methods and URLs, Express allows you to define the routes of your application.

5. Routing mainly aims to describe code that needs to be run in response to any request received by a server. Routing is generally done based on the sequence of URL patterns and the HTTP method, which is associated with the request.
6. If you want to perform additional tasks and functions on any request and response, you can easily use various middleware modules present in Express.
7. The request is a message that arrives at the server requesting something, and a Response is a message sent by the server to a client in the form of the result of whatever the client asked for.
8. If any error occurs and you want to handle it, you can easily handle it by using error handling middleware.
9. Middleware is used somewhere during the lifecycle of a request or response in the form of code. It is mainly used to add functionalities or augment the behavior of the web server.
10. Express also facilitates you to create a REST API (Representational State Transfer Application Programming Interface)
11. The REST API is also known as RESTful API, It mainly conforms to the constraints of REST architectural style, and it also allows for interaction with RESTful web services. The main advantage of REST API is that it provides great flexibility; it uses HTTP requests to access and use data.
12. The data flow into a website structure can easily be facilitated by using the two template engines, EJS and Jade, provided by Express.
13. Express has a gigantic suite of third-party add-ons that developers can use to provide better functionality, help to increase the security level and improve speed.
14. It is very efficient and scalable; one can easily access it from anywhere and use it simultaneously on different systems, and very fast.
15. It is Single-threaded and Asynchronous.
16. It also has the biggest community for Node.js.

17. With its built-in router, it promotes code reusability.

18. If we want to understand the architecture behind web servers and their working along with the organization, then learning Express is the best option.

3.2.3 React Js:

- React is one of the most popular open-source front-end JavaScript libraries used for building Web applications.
- Before using React, it has some prerequisites that one should follow, that you must downloading Node packages in your system with their latest versions. Also, you must have an understanding of HTML, CSS, and JavaScript.
- It is used to build user interfaces, especially for a single-page web application.
- It is not a JavaScript framework. It is just a JavaScript library developed by Facebook to solve problems we could not solve earlier using other libraries while building web and mobile applications.
- React is also used for making a grip over the view layer for mobile and web applications.
- It allows us to create reusable UI (User Interface) components.
- It was first created by software engineer Jordan Walke, who works for Facebook.
- React was first deployed in the Facebook news feed.
- It allows developers to create large web applications that can easily change the data of the page even without reloading the page.
- The main objective of reacting is that it only works on user interfaces in the application, whether mobile or web.
- It is very fast, simple, and scalable.
- React is also used with a combination of other JavaScript libraries or frameworks.
- There are a lot of open-source platforms that are also used to make the front-end web and mobile applications easier, like Angular js in MVC, but still, React replaces Angular from the MEAN stack. Now, most developers are using the MERN stack in which react is used; the main reason is that it is very fast and has more advantages over other front-end frameworks.

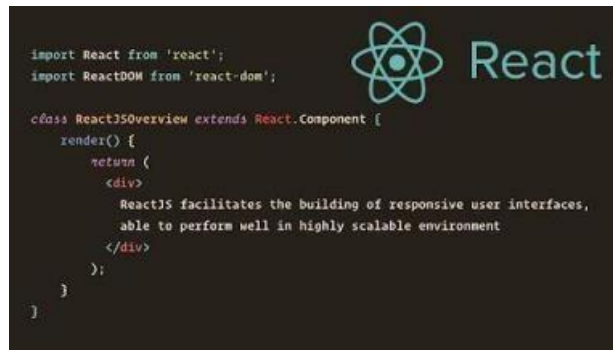


Figure 9 React Example

Some important features of React:

1. **Easy to learn:** One of the great advantages of using React as it is very easy for a beginner to learn it and make web and mobile applications using this front-end framework. Anyone with a piece of previous basic knowledge in programming can easily understand React compared to Angular. Angular is referred to as a 'Domain Specific Language', so it is implied that it is quite difficult to understand it. To learn React, you need a basic knowledge of CSS and HTML.
2. **Simple:** React is one of the simplest open-source JavaScript front-end frameworks for building web and mobile applications. It uses the component-based approach, plain and simple JavaScript, and a well-defined lifecycle, which makes reaction much simpler and easier. So that one can easily learn it and build professional mobile and web applications. It uses a simple syntax named JSX, which allows learners or developers to mix HTML with JavaScript to make it easier for them to apply and use it for making efficient web and mobile applications. However, it is not required to use JSX, you can use plain JavaScript, but as compared to JSX, JSX is the much better option due to its simplicity and easier syntax.
3. **Data Binding:** React uses an application architecture known as Flux to control data flow to components via one control point called the dispatcher. It uses one-way data binding, which makes it easier to debug self-contained components of large React applications.
4. **Native Approach:** React is used to create mobile applications (React Native) and web applications. React allows the reusability of code and can easily support it, which has many benefits and is much time saver. So simultaneously, at the same time, we can make IOS, Web applications,

and Android.

5. **Performance:** React has very fast performance due to the immutability of data. As the name suggests, we can predict that the immutable data structures never change and allow you to compare direct object references instead of doing deep-tree comparisons. The above reason ultimately affects the performance of reacting and makes it faster.

6. **Testability:** React is very easy to test; whatever applications we are generating from React, whether mobile or web applications, it is much easier for us to test it on React.

3.2.4 Node Js -

- Js is an open-source server environment, and it is a cross-platform runtime environment for executing JavaScript code outside a browser.
- Js is not a programming language, and it is not a framework.
- It is often used for building and developing numerous backend services like net applications, and mobile applications.
- Massive corporations principally utilize it in production like Uber, PayPal, Netflix It may be a free ASCII text file platform and may be utilized by anybody.
- It will run on numerous operating systems like Windows, Mac, Linux, Unix, etc.
- It is incredibly simple to get started with it and may even be used for agile development and prototyping.
- It provides extremely ascendable and really quick services to the users.
- It is incredibly consistent and may be used as an ASCII text file cleaner.
- It continuously uses JavaScript; thus, it's ultimately helpful for a computer user to quickly create any net service or any net or mobile application.
- It provides a massive system for any ASCII text file library.
- It contains a non-blocking or, can say, Asynchronous nature.



Figure 10 Node Logo

Some important features of Node Js-

1. **Easy Scalability:** Js is highly scalable because it uses a single-threaded model with event looping. The server usually responds in a non-blocking way due to the help of the event mechanism. It also makes the server very scalable instead of traditional servers that create limited threads to handle requests. Node.js uses a single-threaded program, and this program will be able to provide service to many requests.
2. **Fast:** The event loop in Node.js handles all asynchronous operations, so Node.js acts like a fast suite, and all the operations in Node.js are performed quickly like network connection, reading, or writing in the database, or file system. It runs on the V8 engine developed by Google.
3. **Easy to learn and debug code:** js is quite easy to learn and debug because it uses JavaScript for running the code of web-based projects and various web and mobile applications. If you have excelled in front-end development and have a good command of JavaScript, you can easily build and run the application on Node.js and explore as much as you can; it depends on your capability.
4. **Real-time web apps:** js plays a key role in making real-time web applications. If you are building a mobile or web application, you can also use PHP, although it will take the same time

duration as when you use Node.js. Still, if someone wants to build gaming apps and chat applications, then Node.js is a much better option because of its faster synchronization.

5. **Caching Advantage:** js provides the caching property in which a single module is cached. Sometimes you do not need to re-execute the same lines of code because it has already been cached using Node.js.

6. **Data Streaming:** In Node.js, hypertext transfer protocol (HTTP) requests and responses are units thought of as 2 separate events. They're knowledge streams, thus once you method a file at the time of loading, it'll scale back the time and create it quicker once the info is given within the style of transmissions. It additionally permits you to stream audio and video files at lightning speed.

7. **Object-Oriented Approach:** A huge complaint against Node.js was its JavaScript heritage, which frequently involved many procedural spaghetti codes. Frameworks like Coffee Script and Typescript solved these issues but came as a bolt-on for those who seriously cared about coding standards. With the release and general adoption of ES6, classes are built into the framework, and the code looks syntactically similar to C#, Java, and SWIFT.

8. **Event-driven and synchronization:** All APIs of the Node.js library area unit asynchronous, that is, non-blocking. It suggests a Node.js-based server ne'er waits for associate API to come back to knowledge. The server moves to the consequent API once lines it, and a notification mechanism of Events of Node.js helps the server to urge a response from the previous API decision.

9. **Corporate Support:** There are a lot of famous companies, like PayPal, Wal-Mart, Microsoft, and Google that are using Node.js for building applications. Node.js uses JavaScript, so most companies are combining front-end and backend Teams into a single unit.

4. PROJECT UNDERTAKEN

4.1 BACKGROUND OF THE STUDY

GrocerEase is a comprehensive grocery web app built using the MERN stack, designed to provide users with a seamless online shopping experience. GrocerEase allows users to browse and purchase a wide variety of grocery products, add items to their cart, and search for specific products easily. Additionally, the platform offers admin functionalities, enabling administrators to manage the product inventory by adding new products to the app with ease. GrocerEase is optimized for smooth performance, ensuring that all operations are handled in real-time with minimal delay.

Our platform stands out with its intuitive user interface, allowing for an easy and efficient shopping experience. Users can explore different product categories, manage their cart, and complete their purchases with just a few clicks. Whether you're stocking up on essentials or trying out new products, GrocerEase provides a reliable and user-friendly platform for all your grocery needs.

GrocerEase is designed to run on various devices, making it accessible via web browsers on both desktop and mobile platforms. As an ideal solution for entrepreneurs looking to establish a robust online grocery business, GrocerEase offers a high level of customization, allowing for modifications to suit specific business needs.

GrocerEase is more than just a shopping platform; it's a solution tailored to meet the needs of modern consumers and businesses alike. The platform allows users to efficiently purchase groceries from the comfort of their homes, saving time and effort. With features like a dynamic product search, user-friendly cart management, and streamlined checkout processes, GrocerEase makes grocery shopping both convenient and enjoyable.

For administrators, GrocerEase offers a powerful tool to manage products and inventory effortlessly. Adding new products to the catalog is straightforward, ensuring that the platform remains up-to-date with the latest offerings.

GrocerEase provides an opportunity for entrepreneurs to tap into the growing online grocery market. With the ability to customize the app's features and layout, businesses can create a unique shopping

experience that differentiates them from competitors. The app's robust architecture, built on the MERN stack, ensures scalability and reliability, making it an ideal choice for businesses looking to grow in the digital space.

Whether you're a user looking for a convenient way to shop for groceries or an entrepreneur aiming to build a successful online grocery business, GrocerEase offers the tools and flexibility to meet your needs.

4.2 OBJECTIVE

The primary objective of the Grocery Delivery Web App is to provide an efficient, user-friendly, and secure platform for customers to order groceries online and have them delivered to their doorstep. The app aims to streamline the grocery shopping experience by allowing users to browse a wide range of products, manage their carts, and complete orders seamlessly, all from the convenience of their devices.

Key objectives include:

1. **Enhanced User Experience:** Develop an intuitive and responsive user interface that allows customers to search for products, filter by categories, and easily add items to their carts.
2. **Seamless Ordering Process:** Ensure a smooth and error-free checkout process, including multiple payment options, real-time order tracking, and delivery scheduling.
3. **Efficient Inventory Management:** Implement a real-time inventory management system that keeps track of stock levels, updates product availability, and reduces order cancellations due to out-of-stock items.
4. **Scalability and Performance:** Build a scalable and high-performing application that can handle increased user traffic, large datasets, and multiple transactions simultaneously without compromising performance.
5. **Security and Data Privacy:** Ensure the security of customer data, including payment information and personal details, by implementing secure authentication and encryption protocols.
6. **Customer Retention and Engagement:** Incorporate features like personalized recommendations, promotions, and loyalty programs to enhance customer engagement and encourage repeat purchases.
7. **Mobile Responsiveness:** Design a mobile-responsive web app that provides an optimized shopping experience across various devices, particularly smartphones and tablets.

By achieving these objectives, the Grocery Delivery Web App will cater to the evolving needs of modern consumers while fostering growth and operational efficiency for the business.

4.3 METHODOLOGY

To build a robust and scalable Grocery Delivery Web App, the following methodology will be followed, combining agile development practices with iterative feedback cycles to ensure continuous improvement and alignment with user needs.

8. Requirements Gathering and Analysis

- a) **Market Research:** Analyze existing grocery delivery apps to identify best practices, competitive advantages, and gaps in the market.
- b) **Feature Prioritization:** Define core features (e.g., product search, cart management, checkout, delivery tracking) and prioritize them based on user impact and business value.

9. Design

- a) **UI/UX Design:** Develop wireframes and mockups that focus on a clean, intuitive, and mobile-responsive interface. Ensure ease of navigation, accessibility, and a visually appealing layout.
- b) **Ensure responsiveness:** Design the app to be responsive and accessible on various devices (desktop, tablet, mobile).

10. Technology Stack Selection

- a) **Front-end:** Use **React.js** to create a dynamic, responsive, and user-friendly interface.
- b) **Back-end:** Utilize **Node.js** and **Express.js** to build the server-side logic and API endpoints for handling requests.
- c) **Database:** Implement **MongoDB** for flexible, scalable, and efficient data storage, particularly well-suited for managing dynamic product inventories.
- d) **Third-party Integrations:** Integrate secure payment gateways (e.g., Stripe), user authentication (e.g., JWT), and real-time order tracking APIs.
- e) **Deployment:** Use cloud services like AWS or Heroku for scalable and reliable hosting, and implement CI/CD pipelines for continuous deployment.

11. Backend Development

- a) **API Development:** Develop RESTful APIs to manage CRUD operations for products, orders, and user profiles. Ensure proper authentication and authorization mechanisms.
- b) **Database Design:** Structure the MongoDB database to optimize performance, including indexing for quick product searches and efficient handling of large datasets.

12. Front-end Development

- a) **Component-based Architecture:** Use React's component-based architecture to create reusable UI components that ensure consistent design and functionality across the app.
- b) **State Management:** Implement Context API or Redux for efficient state management, especially for handling cart data, user sessions, and product lists.
- c) **Responsive Design:** Ensure the app is fully responsive, providing a seamless experience across desktop and mobile devices.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The **System Architecture** for a Grocery Delivery Web App involves organizing and defining the structure of the application across various components, including the frontend, backend, database, and third-party services. This architecture ensures that the app is scalable, secure, and capable of handling large volumes of data and transactions. Below is a detailed breakdown of the system architecture:

1. High-Level System Architecture Overview

The architecture is based on a **3-tier model** consisting of:

- a) **Client (Presentation Layer)**
- b) **Application (Business Logic Layer)**
- c) **Database (Data Layer)**

Tech Stack:

- a) **Frontend:** React.js
- b) **Backend:** Node.js, Express.js
- c) **Database:** MongoDB (NoSQL)
- d) **Third-Party Integrations:** Payment Gateway (Stripe/PayPal)

2. Architecture Components

A. Frontend (Client-Side)

- a) **User Interface:** Built with React.js, providing a responsive and interactive experience for users. The frontend handles:
 - i. **Product Browsing:** Displaying product catalogs with filtering and sorting options.
 - ii. **User Authentication:** Login, registration, and profile management.
 - iii. **Shopping Cart:** Adding/removing items, calculating totals, and applying discounts or promo codes.
 - iv. **Checkout Process:** Collecting user details, selecting payment methods, and placing orders.
 - v. **Real-Time Notifications:** Displaying order updates (e.g., order placed, out for

delivery) using WebSockets.

Frontend Key Aspects:

- vi. **React Router:** For navigation and page routing (e.g., Home, Cart, Product Detail).
- vii. **State Management:** Redux or Context API for managing global state (e.g., user data, cart items).
- viii. **API Communication:** Fetch data from the backend via RESTful APIs (Axios or Fetch API).
- ix. **Responsive Design:** Tailored for both desktop and mobile devices using CSS frameworks (e.g., Tailwind CSS, Bootstrap).

B. Backend (Application Layer)

- a) **Node.js & Express.js:** The backend is responsible for processing business logic, handling API requests, and communicating with the database. The core functionalities include:
 - i. **User Authentication and Authorization:** Secure login, registration, and JWT-based session management.
 - ii. **Product Management:** CRUD operations for products (create, read, update, delete).
 - iii. **Order Management:** Creating orders, updating order status, and managing payment processing.
 - iv. **Real-Time Updates:** Implementing WebSocket communication (e.g., with Socket.io) for live notifications about order status and delivery tracking.
 - v. **Admin Panel API:** Separate API endpoints for managing products, viewing sales data, and tracking inventory.

Backend Key Aspects:

- i. **RESTful APIs:** Organized into various routes (e.g., /products, /users, /orders) for different operations.
- ii. **Middleware:** Used for tasks like authentication, validation, and error handling.
- iii. **Scalability:** Designed to scale horizontally, with potential for microservices in the future.
- iv. **Caching:** Using Redis or in-memory caching for frequently accessed data, improving performance.

C. Database (Data Layer)

- a) **MongoDB:** A NoSQL database used for storing and managing data in collections.

MongoDB offers flexibility in handling unstructured or semi-structured data and is well-suited for dynamic inventory management.

i. **Collections:**

- **Users:** Stores user information, authentication credentials, and order history.
- **Products:** Stores product details such as name, price, category, description, and stock levels.
- **Orders:** Stores order details, including user ID, product list, total price, order status, and delivery information.
- **Reviews:** Stores user-generated reviews, linked to product and user IDs.

ii. **Indexes:** Indexes are used on frequently queried fields like product categories and order statuses to optimize search performance.

Database Key Aspects:

- i. **Document-Oriented:** MongoDB stores data in BSON format, which allows for flexible document structures.
- ii. **Data Relationships:** Implement references between collections (e.g., embedding order details within a user document) to optimize data retrieval.
- iii. **Backup and Recovery:** Regular backups and disaster recovery strategies are implemented to protect data integrity.

3. Third-Party Integrations

- a) **Payment Gateway:** Stripe or PayPal integration for handling secure transactions, ensuring PCI compliance, and processing payments within the app.
- b) **Email Services:** Services like Email Js and Formspree are used for sending transactional emails such as order confirmations and receipts.

5.2 USE CASE DIAGRAM

Use case model the system from the end users' point of view, with the following objectives:

- To define the functional and operational requirements of the system by defining a scenario of usage.
- To provide a clear and unambiguous description of how the users and the system interact with one another.

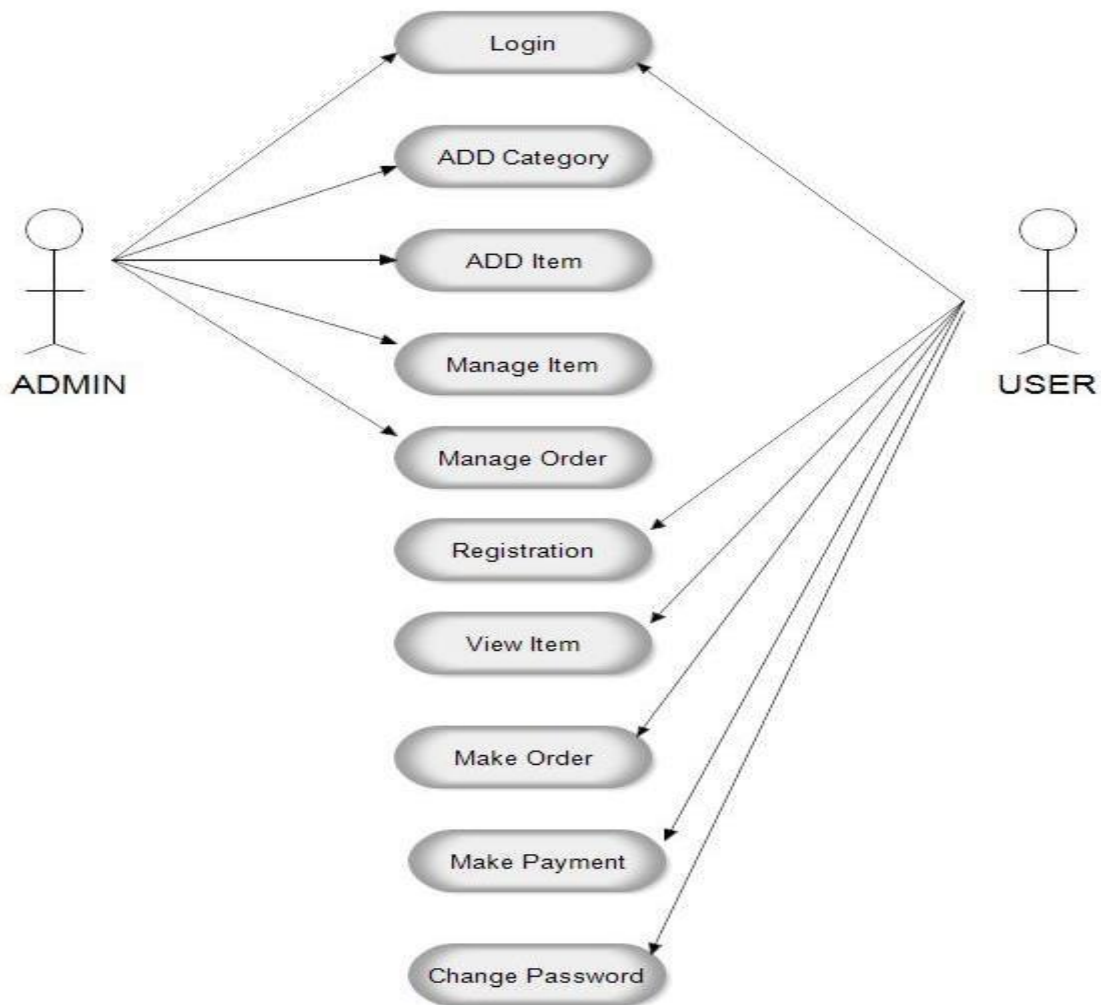


Figure 11 Use Case Diagram

5.3 ENTITY-RELATIONSHIP DIAGRAM

To create an Entity Relationship Diagram (ERD) for a grocery web app, we need to identify the main entities and their relationships. Here's an overview of what an ERD for a grocery delivery web app might include:

1. Entities

a) User

- I. Attributes: user_id (PK), name, email, password, phone_number, address, role (e.g., customer, admin)

b) Product

- I. Attributes: product_id (PK), name, description, price, category_id (FK), stock_quantity, image_url

c) Category

- I. Attributes: category_id (PK), category_name

d) Order

- I. Attributes: order_id (PK), user_id (FK), order_date, total_amount, status (e.g., pending, delivered, canceled)

e) OrderItem

- I. Attributes: order_item_id (PK), order_id (FK), product_id (FK), quantity, price_per_unit

f) Cart

- I. Attributes: cart_id (PK), user_id (FK)

g) CartItem

- I. Attributes: cart_item_id (PK), cart_id (FK), product_id (FK), quantity

h) Review

- I. Attributes: review_id (PK), product_id (FK), user_id (FK), rating, comment, review_date

i) Payment

- I. Attributes: payment_id (PK), order_id (FK), payment_method, payment_date, amount_paid

2. Relationships

- A **User** can have many **Orders** (One-to-Many).
- A **User** can have one **Cart** (One-to-One).
- A **User** can write many **Reviews** for **Products** (One-to-Many).
- A **Product** belongs to one **Category** (Many-to-One).
- A **Product** can be in many **OrderItems** (One-to-Many).
- A **Product** can be in many **CartItems** (One-to-Many).
- An **Order** can have many **OrderItems** (One-to-Many).
- A **Cart** can have many **CartItems** (One-to-Many).
- An **Order** has one **Payment** (One-to-One).

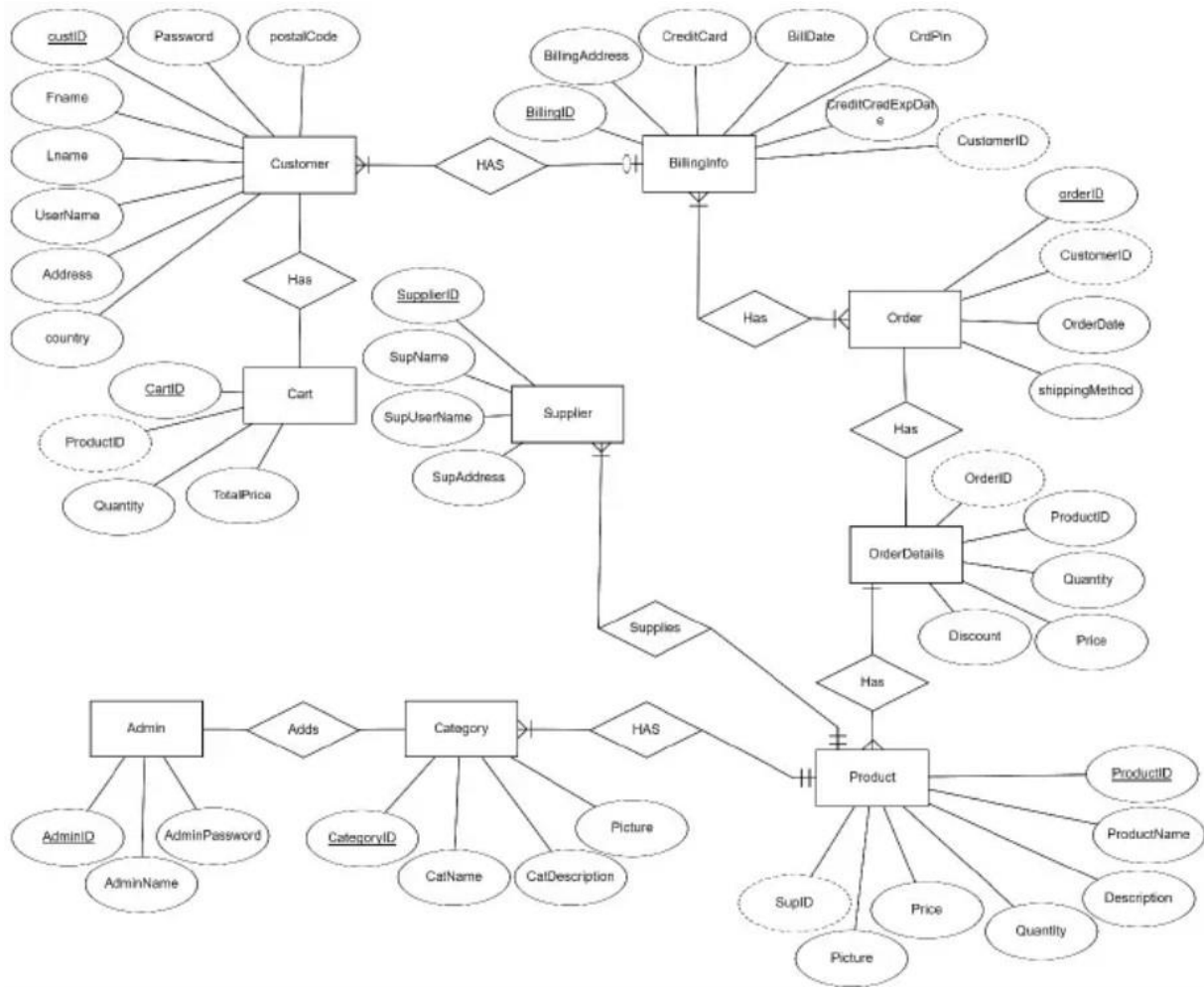


Figure 12 ER Diagram

6. IMPLEMENTATION

6.1 DEVELOPMENT ENVIRONMENT

The development environment for the Restaurant Onboarding Web Application is set up to ensure smooth and efficient development, testing, and deployment. It includes the necessary software and tools to support the development process.

1. Development Tools:

- a) **Integrated Development Environment (IDE):** Visual Studio Code
- b) **Version Control:** Git and GitHub
- c) **Package Management:** npm (Node Package Manager)

2. Local Development Setup:

- a) **Operating System:** Windows
- b) **Web Browser:** Google Chrome (for testing and debugging)
- c) **Database Management:** Mongo DB

6.2 TOOLS AND TECHNOLOGIES USED

1. Frontend:

- a) **React:** For building the user interface and managing application state.
- b) **Tailwind CSS:** For utility-first styling and responsive design.
- c) **HTML/CSS:** For structuring and styling the web pages.
- d) **Axios:** For handling HTTP requests to communicate with the backend.
- e) **React Router:** For handling routing in the single-page application.

2. Backend:

- a) **Node.js:** For server-side logic and handling API requests.
- b) **Express.js:** For building and managing the web server and routes.
- c) **MongoDB:** As a NoSQL database for storing user information, products, and orders.
- d) **Mongoose:** For modeling MongoDB data and interacting with the database.

- e) **bcrypt.js**: For hashing and securing user passwords.
- f) **jsonwebtoken (JWT)**: For user authentication and session management.

3. Version Control:

- a) **Git**: For tracking changes and version control.
- b) **GitHub**: For remote repository hosting and collaboration.

4. Package Managers:

- a) **npm**: For managing project dependencies and scripts.

5. Development Environment:

- a) **Visual Studio Code (VS Code)**: For writing and debugging code.

6. API Testing:

- a) **Postman**: For testing and documenting API endpoints.

7. Environment Management:

- a) **dotenv**: For managing environment variables like API keys and database URLs.

8. Deployment:

- a) **Netlify**: For deploying the frontend React application.
- b) **Heroku**: For deploying the backend Node.js server.

9. Browser Tools:

- a) **Chrome DevTools**: For debugging frontend code and monitoring network activity.

7. SNAPSHOTS OF THE PROJECT

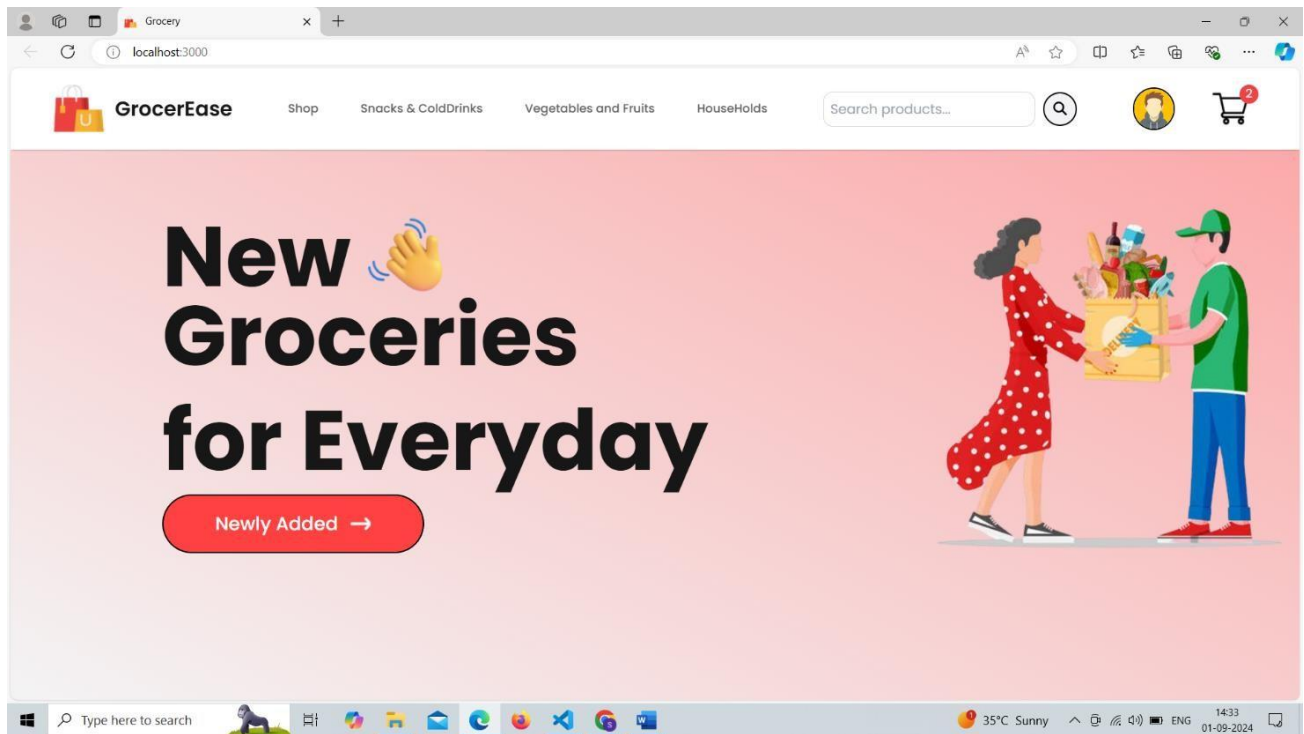


Figure 13 Home Page

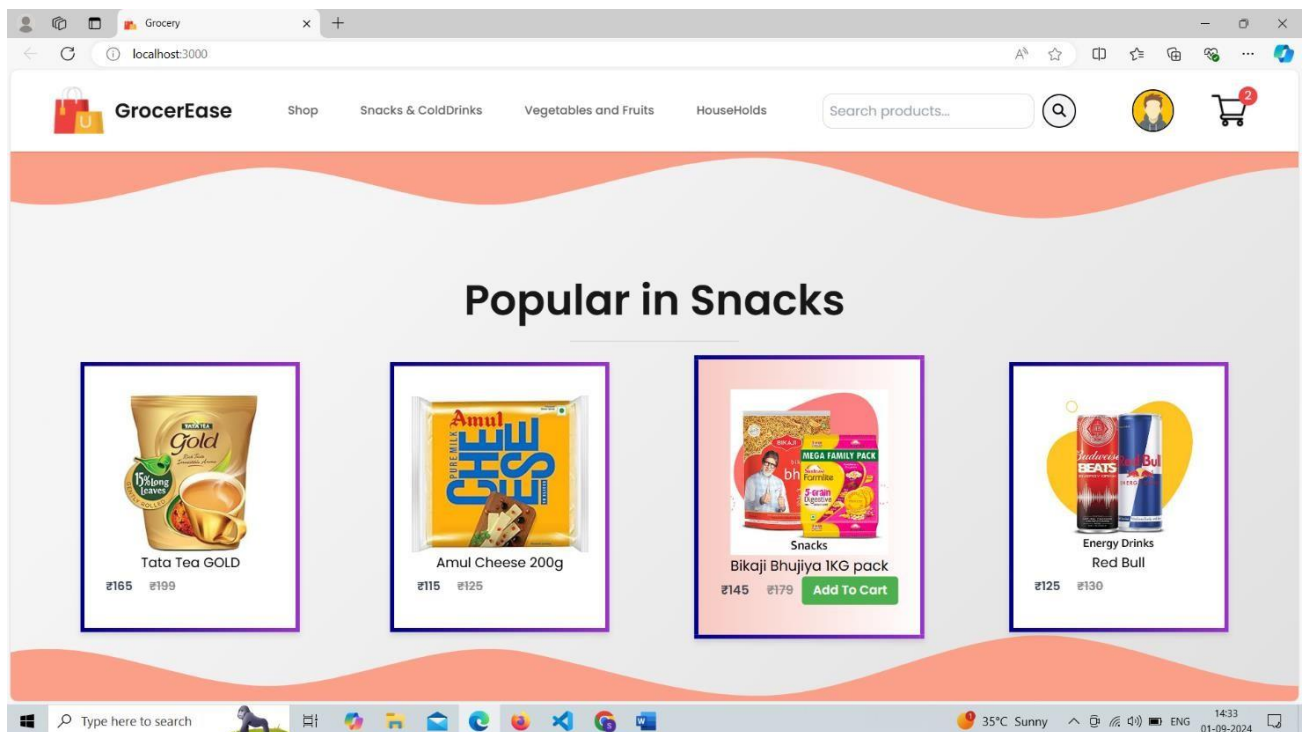


Figure 14 Popular in Snacks Component

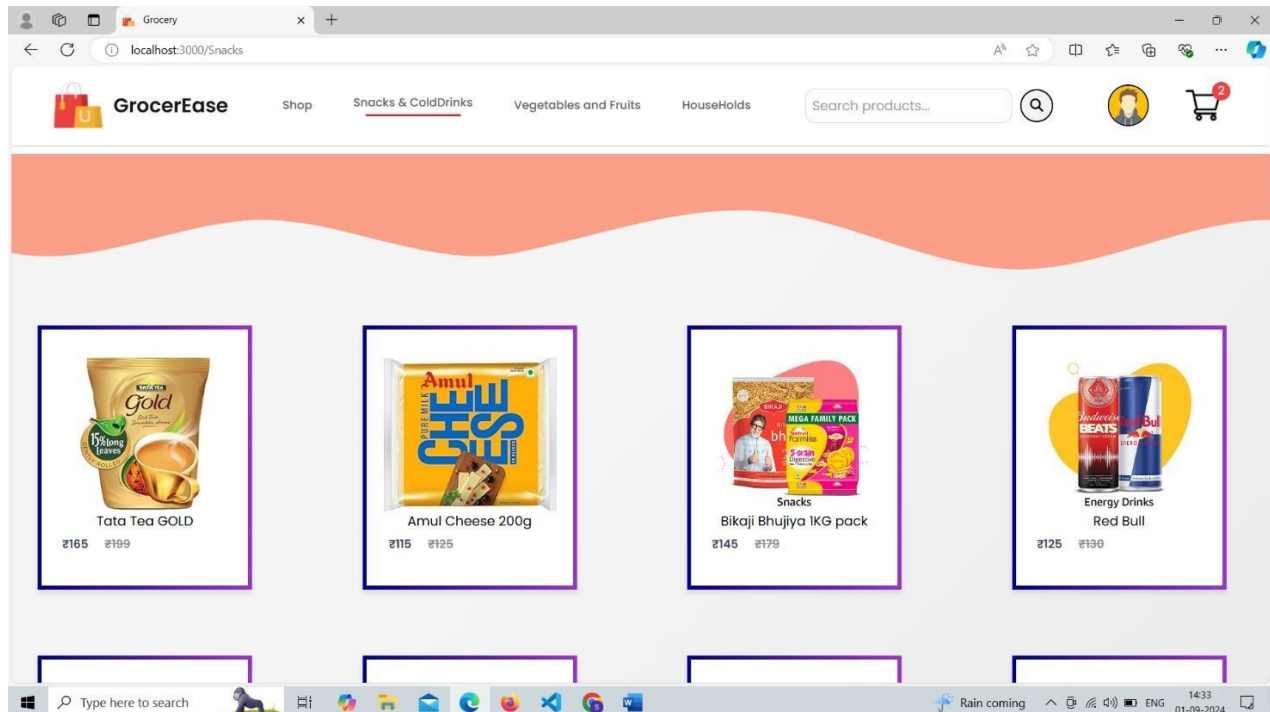


Figure 15 Snacks and Coldrinks Component

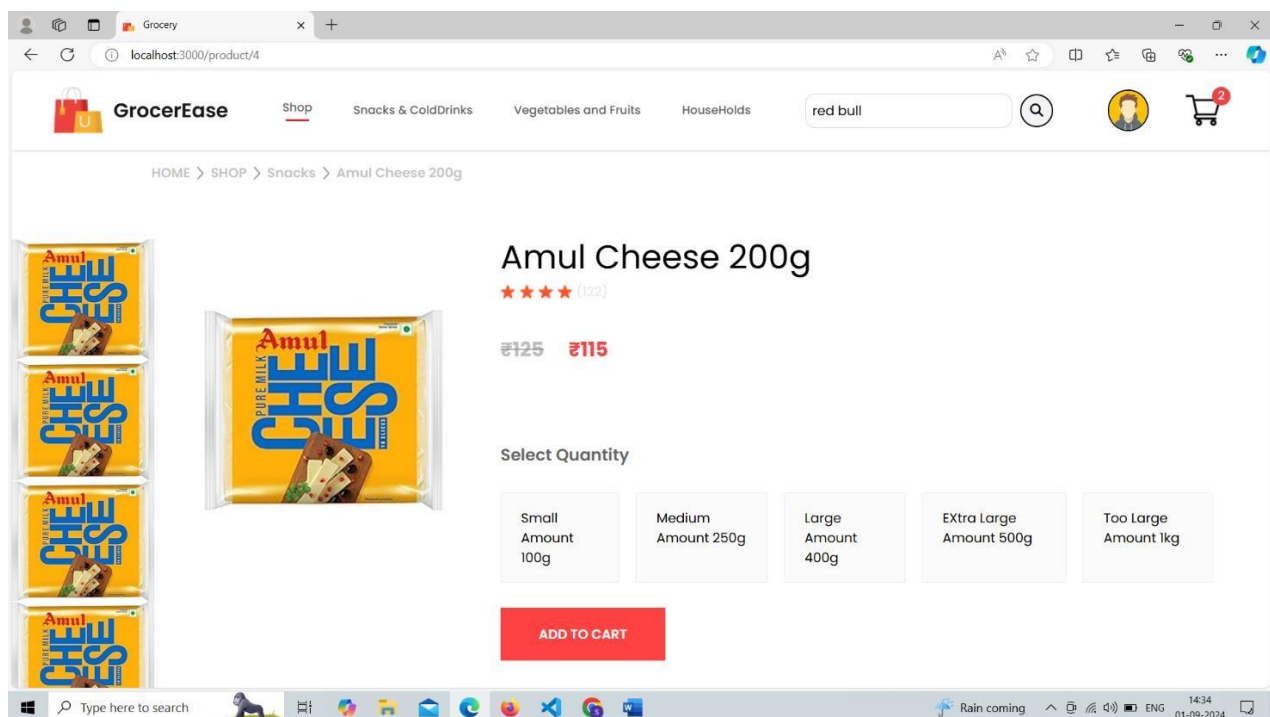


Figure 13 Product Description Page

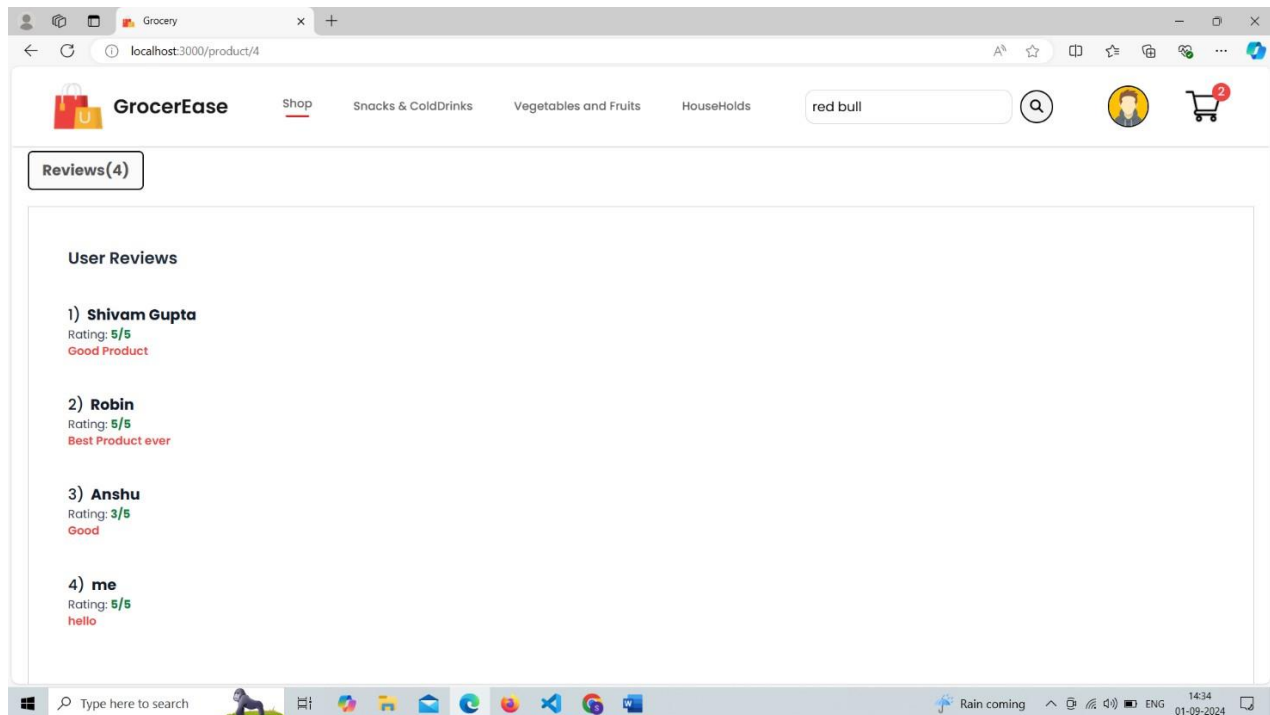


Figure 14 Product Review Section

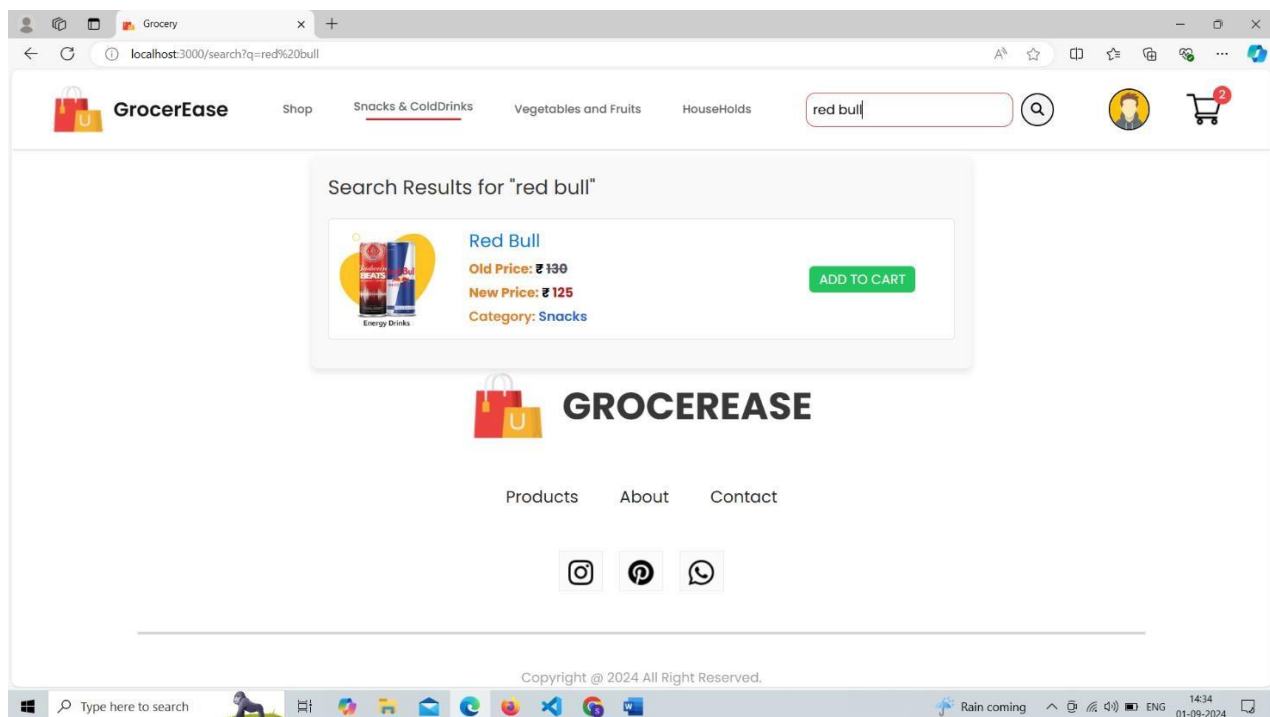


Figure 15 Searched Product

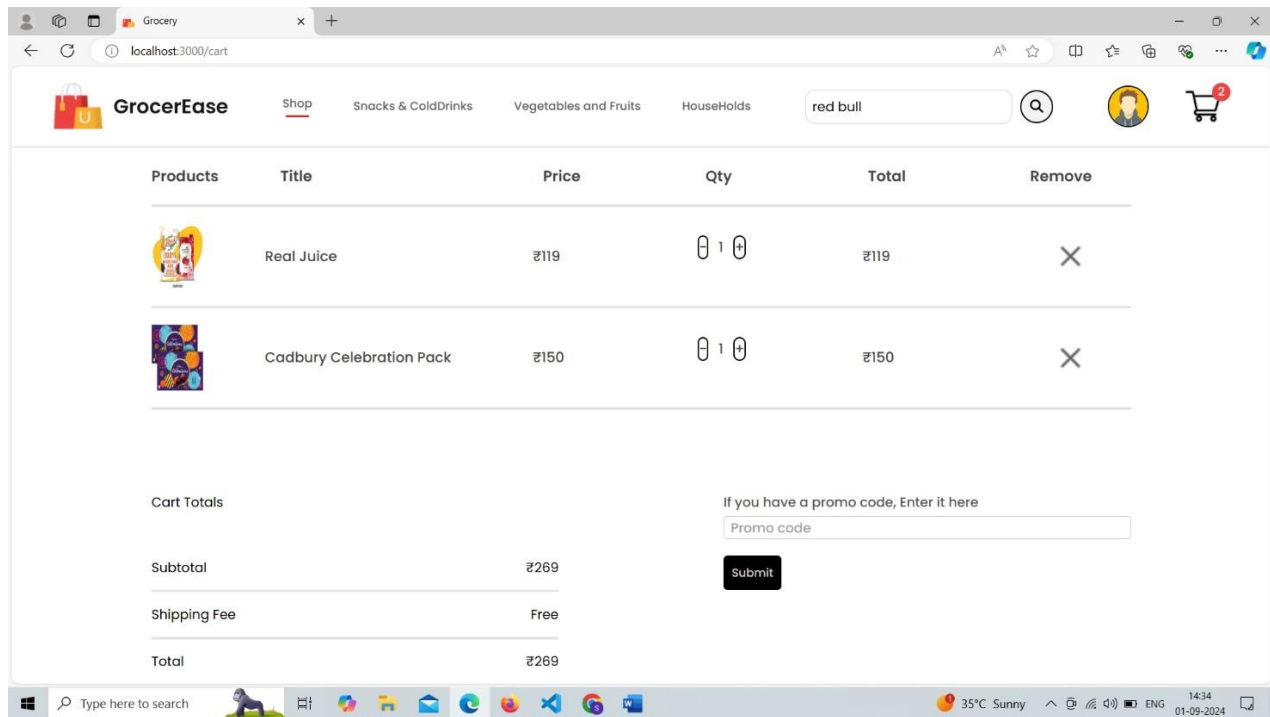


Figure 16 Cart Item Page

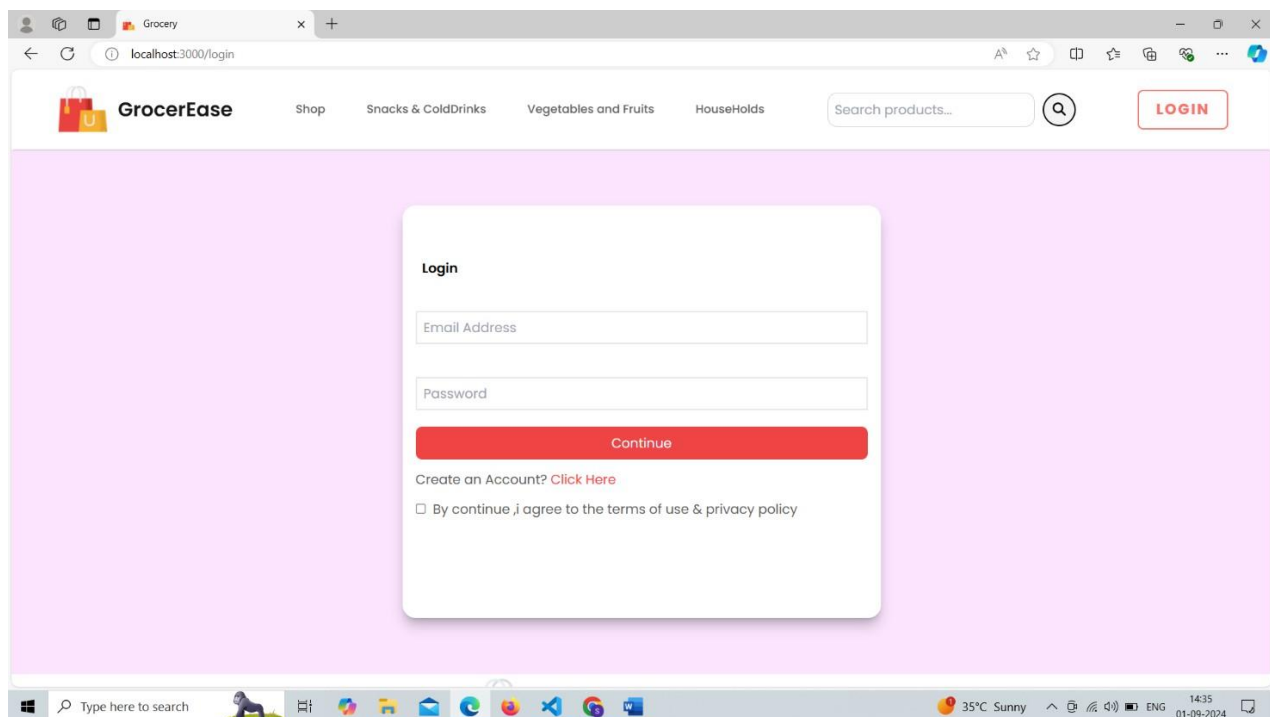


Figure 17 Login Page

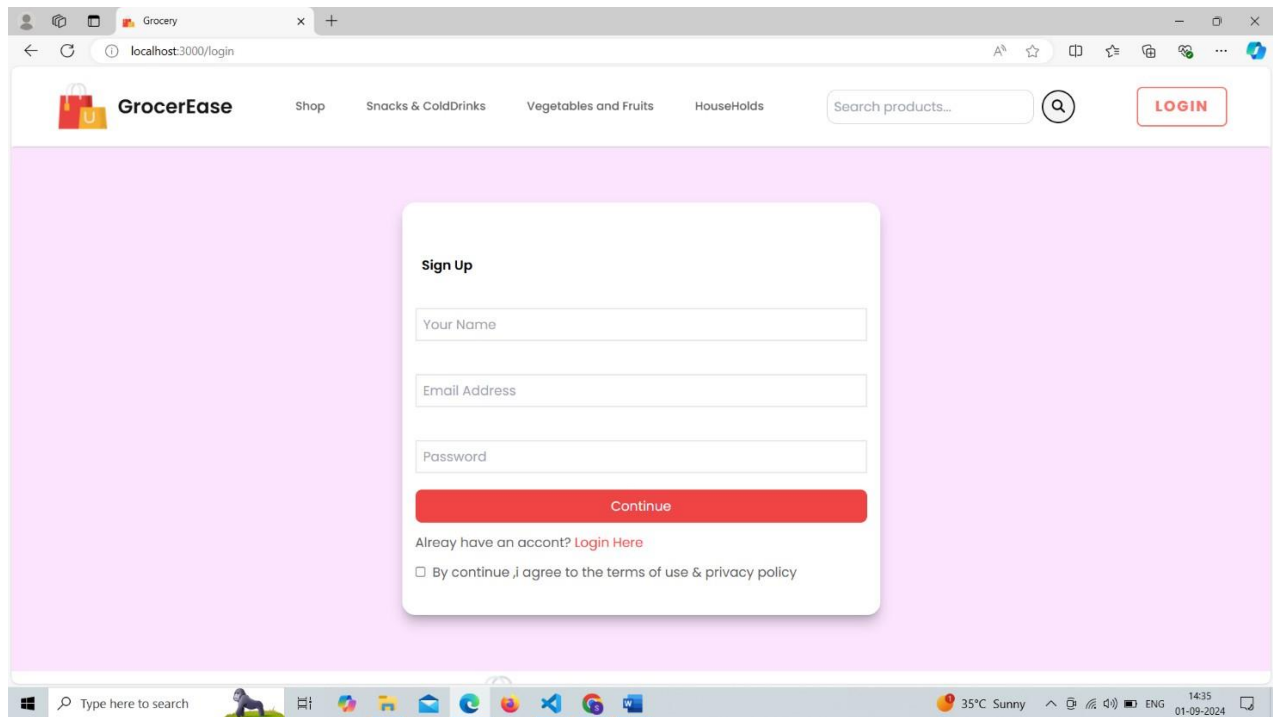


Figure 18 Sign Up Page

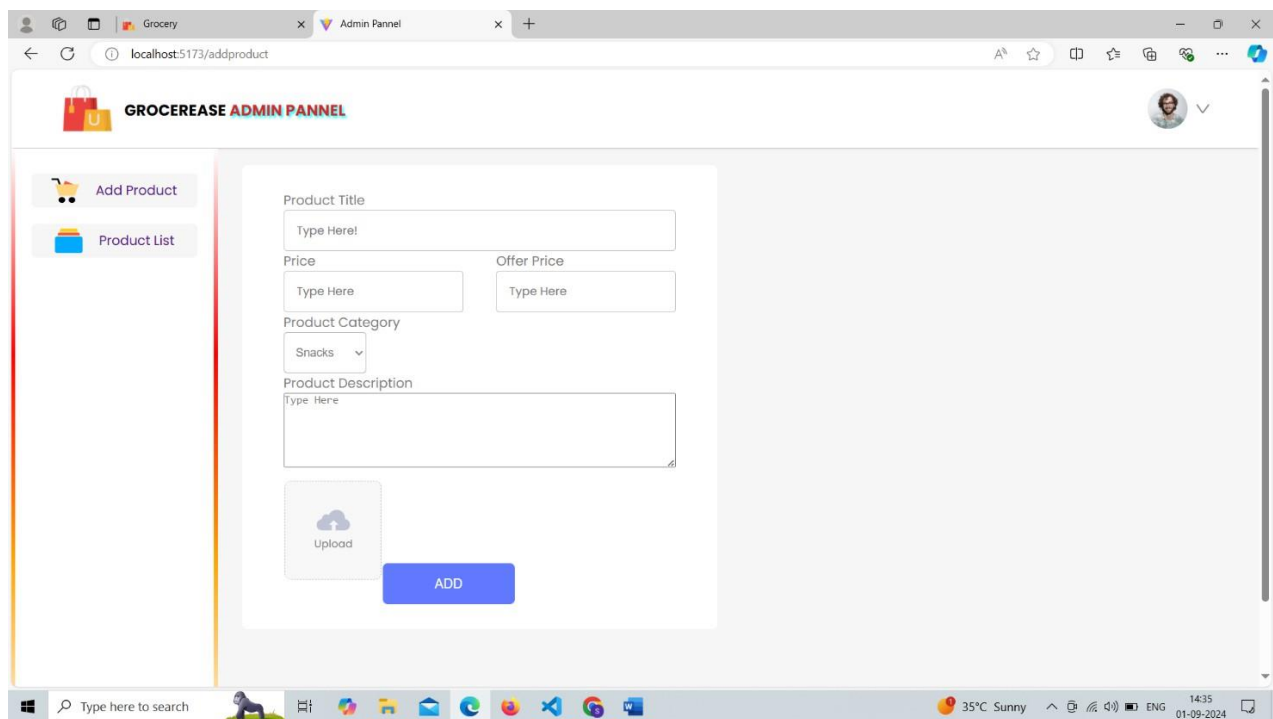


Figure 19 Admin Panel (Add Product)

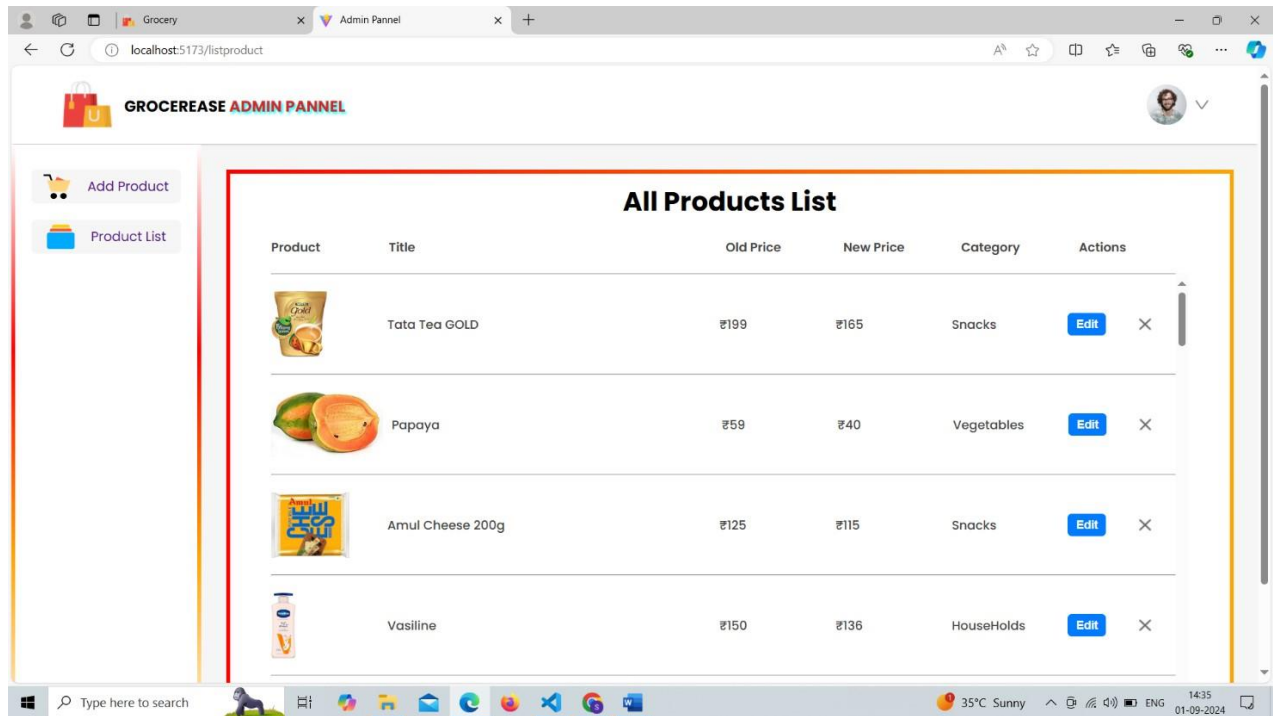


Figure 20 Admin Panel (Product List)

8. LESSONS LEARNED

8.1. Full-Stack Development Integration:

- a) **Learning:** The project reinforced the importance of integrating frontend and backend components seamlessly. Understanding how to connect React with Node.js and Express.js to build a full-stack application was crucial.
- b) **Impact:** This experience improved skills in managing state across frontend components and handling asynchronous data fetching using Axios.

8.2. Responsive Design and UI/UX Considerations:

- a) **Learning:** Building a user-friendly and responsive design using Tailwind CSS was essential in delivering a good user experience. Adapting the layout to different screen sizes (mobile-first design) ensured the app's accessibility across devices.
- b) **Impact:** This enhanced the ability to focus on user interface details, optimize for mobile devices, and ensure that the design remained consistent.

8.3. API Development and Integration:

- a) **Learning:** Developing a RESTful API using Express.js taught valuable lessons in structuring endpoints, handling HTTP requests, and implementing CRUD operations (Create, Read, Update, Delete) efficiently.
- b) **Impact:** This experience enhanced understanding of REST principles, routing, middleware usage, and best practices for securing and documenting APIs.

8.4. Authentication and Authorization:

- a) **Learning:** Implementing user authentication using JWT (JSON Web Tokens) provided insight into securing routes and managing user sessions. Handling user login, registration, and authentication flow was critical to protecting sensitive data.
- b) **Impact:** This experience deepened the knowledge of token-based authentication, protecting routes, and managing security in a web application.

8.5. Database Management with MongoDB:

- a) **Learning:** Working with MongoDB and Mongoose to design schemas and manage the database introduced important lessons in NoSQL database management, especially in terms of schema design and handling relationships between different collections (e.g., users, products, orders).

- b) **Impact:** This improved understanding of data modeling, querying databases, and handling large data sets efficiently.

8.6. State Management in React:

- a) **Learning:** Managing application state across different components using React hooks (like `useState` and `useContext`) and Redux for larger-scale state management highlighted the importance of structured state handling in complex applications.
- b) **Impact:** This experience improved proficiency in managing global state, optimizing rendering, and avoiding common pitfalls like prop drilling.

8.7. Error Handling and Debugging:

- a) **Learning:** Debugging issues both in the frontend and backend, as well as implementing error-handling mechanisms, was a key part of the development process. Learning to handle different types of errors (network errors, server errors, validation errors) was crucial.
- b) **Impact:** This experience sharpened skills in troubleshooting issues effectively and writing more resilient and fault-tolerant code.

8.8. Performance Optimization:

- a) **Learning:** The project emphasized the need to optimize both the frontend and backend for performance. This included lazy loading of components, optimizing database queries, and reducing the bundle size of the React app.
- b) **Impact:** Gained a better understanding of performance bottlenecks and how to address them to improve the overall speed and responsiveness of the app.

8.9. Version Control and Collaboration:

- a) **Learning:** Using Git and GitHub for version control underscored the importance of managing code changes effectively. Collaborating on the project taught valuable lessons in handling merge conflicts, creating meaningful commits, and maintaining clean code history.
- b) **Impact:** Improved proficiency in Git workflows, collaboration in a team setting, and following best practices in version control.

8.10. Deployment and CI/CD Pipelines:

- a) **Learning:** Deploying the app on platforms like Netlify (frontend) and Heroku (backend) provided insights into deployment strategies, environment configuration, and maintaining

uptime. Setting up CI/CD pipelines for automatic deployment was a significant learning curve.

- b) **Impact:** This experience reinforced the importance of deployment automation, environment management, and ensuring smooth transitions from development to production.

8.11. User-Centric Development:

- a) **Learning:** Gathering feedback on the user experience and making iterative improvements to the app emphasized the importance of user-centric design and development. Understanding user needs and continuously refining the app for better usability was a key takeaway.
- b) **Impact:** Gained a better appreciation for the user experience, leading to more thoughtful design decisions and prioritizing user feedback in the development process.

9. FUTURE SCOPE

9.1. AI-Powered Product Recommendations:

- a) **Scope:** Implementing machine learning algorithms to provide personalized product recommendations based on users' past purchases, search history, and browsing behavior.
- b) **Impact:** AI-powered recommendations can increase sales by suggesting relevant products and improve the overall user experience with more personalized shopping.

9.2. Subscription and Delivery Plans:

- a) **Scope:** Introducing subscription services for recurring purchases (e.g., weekly groceries) and offering various delivery plans such as same-day delivery, express delivery, or scheduled delivery.
- b) **Impact:** Subscription models would enhance customer loyalty, provide convenience, and create a steady revenue stream. Flexible delivery options would cater to diverse customer needs.

9.3. Advanced Search and Filtering Options:

- a) **Scope:** Enhancing the search functionality by adding voice search, image-based search, and more advanced filtering and sorting options (e.g., by dietary preferences, organic products, etc.).
- b) **Impact:** This would improve the user experience by allowing customers to find products more easily, catering to specific needs like dietary restrictions or eco-friendly products.

9.4. Enhanced Security and Privacy Features:

- a) **Scope:** Implementing two-factor authentication (2FA), biometric login options (fingerprint/face ID), and data encryption to protect user data and enhance the security of the platform.
- b) **Impact:** Strengthened security would build user trust, protect sensitive information (such as payment details), and reduce the risk of data breaches.

10. CONCLUSION

The development of the grocery web app marked a significant achievement in providing users with a seamless and convenient shopping experience. By leveraging modern web technologies such as React for the frontend and Node.js/Express.js for the backend, along with MongoDB for robust data management, the app successfully addressed the needs of both customers and store owners in the digital grocery market.

The project emphasized the importance of responsive design, user-centric features, and secure authentication, ensuring that users could easily browse, search, and purchase groceries from the comfort of their homes. Additionally, integrating key functionalities like order tracking, payment gateways, and user profiles enhanced the overall user experience and fostered customer trust.

Throughout the project, valuable lessons were learned in full-stack development, state management, API design, and deployment. These insights not only contributed to the success of the grocery web app but also laid the groundwork for future improvements and scalability.

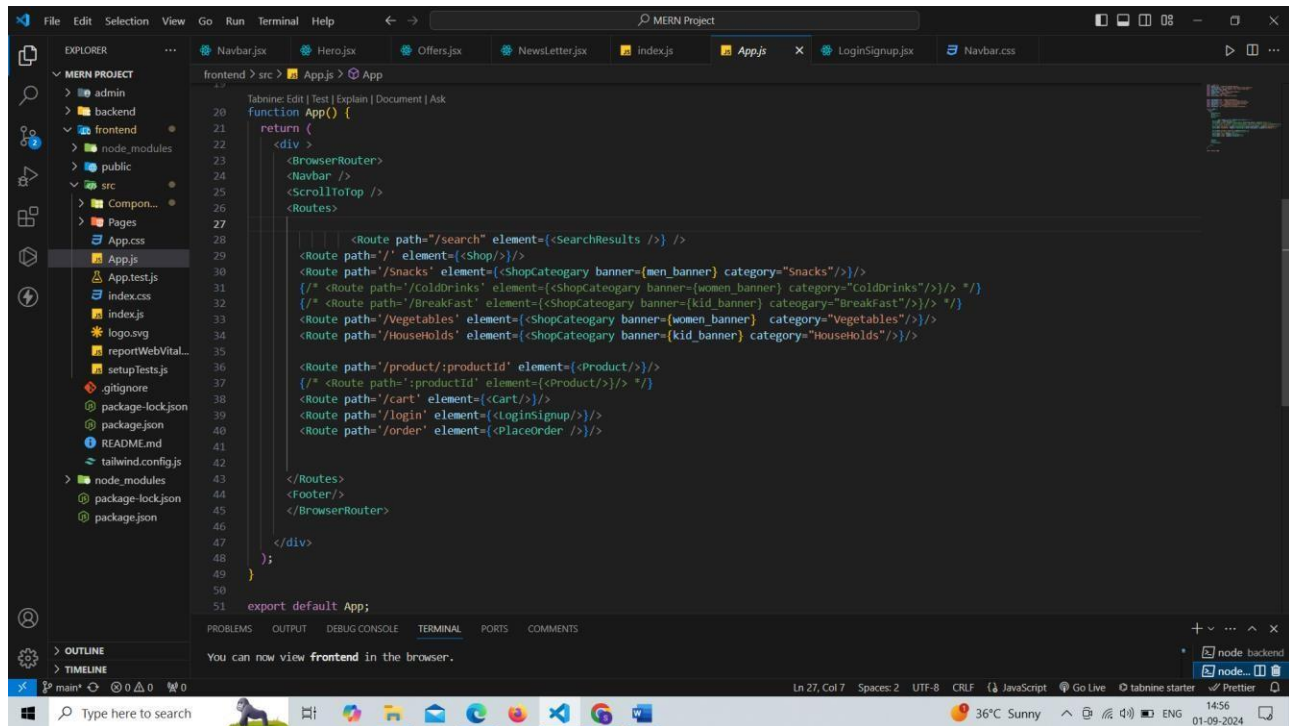
With its potential for growth in areas such as mobile app development, AI-powered features, and expanded delivery options, the grocery web app is poised to adapt to evolving market demands and continue offering an innovative solution in the online grocery space. The project stands as a testament to the team's technical expertise and commitment to creating a high-quality digital product that meets real-world needs.

REFERNCES

- [1] Sommerville, I. (2016). *Software Engineering* (10th Ed.). Boston: Pearson.
- [2] Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th Ed.). McGraw-Hill Education.
- [3] React Js Documentation: “<https://reactjs.org/docs/getting-started.html>”
- [4] Node.js Documentation: “<https://nodejs.org/en/docs/>”
- [5] Tailwind CSS Documentation: “<https://tailwindcss.com/docs>”
- [6] Express.js Documentation: “<https://expressjs.com/>”
- [7] ER Diagram Design: “<https://www.mermaidchart.com/>”
- [8] GitHub: “<https://github.com/>”
- [9] MongoDB: “<https://www.mongodb.com/>”
- [10] Mongoose Documentation:” <https://mongoosejs.com/docs/>”
- [11] JWT (JSON Web Tokens) Documentation:” <https://jwt.io/introduction/>”
- [12] Axios Documentation:” <https://axios-http.com/docs/intro>”
- [13] Postman API Testing:” <https://learning.postman.com/docs/getting-started/introduction/>”
- [14] bcrypt.js Documentation:” <https://www.npmjs.com/package/bcrypt>”

APPENDIX

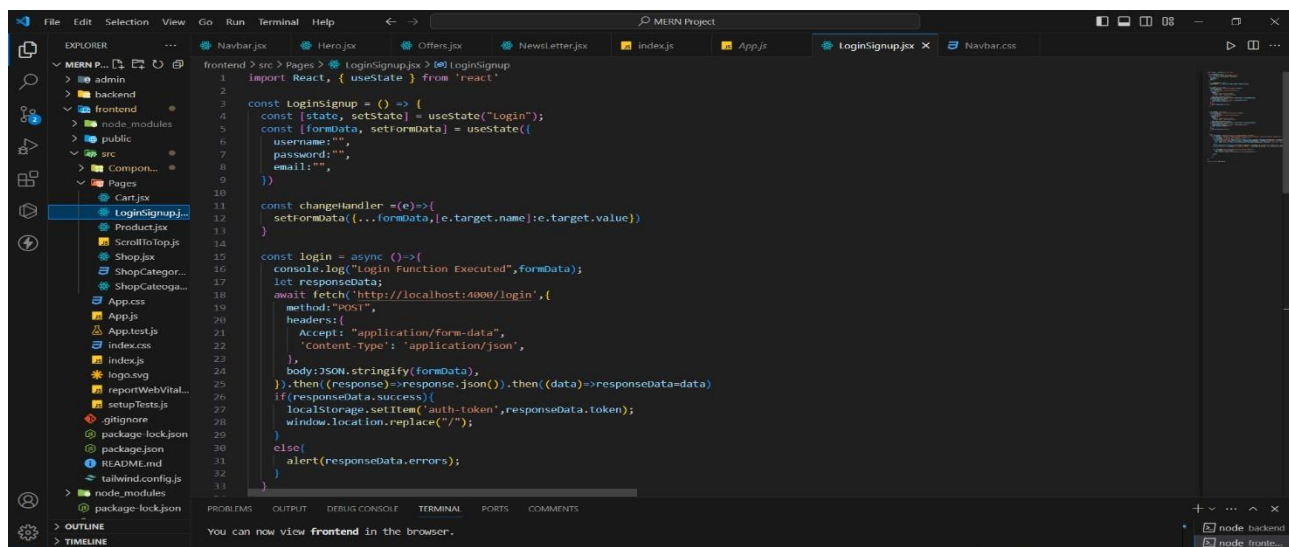
FRONTEND CODE FOR GROCEREASE



```
function App() {
  return (
    <div>
      <BrowserRouter>
      <Navbar />
      <ScrollToTop />
      <Routes>
        <Route path="/" element={}<SearchResults /> />
        <Route path="/" element={}<Shop /> />
        <Route path="/Snacks" element={}<ShopCategory banner={men_banner} category="Snacks" /> />
        <Route path="/ColdDrinks" element={}<ShopCategory banner={women_banner} category="ColdDrinks" /> />
        <Route path="/BreakFast" element={}<ShopCategory banner={kid_banner} category="BreakFast" /> />
        <Route path="/Vegetables" element={}<ShopCategory banner={women_banner} category="Vegetables" /> />
        <Route path="/HouseHolds" element={}<ShopCategory banner={kid_banner} category="HouseHolds" /> />
        <Route path="/product/:productId" element={}<Product /> />
        <Route path="/cart" element={}<Cart /> />
        <Route path="/login" element={}<LoginSignup /> />
        <Route path="/order" element={}<PlaceOrder /> />
      </Routes>
      <Footer />
    </BrowserRouter>
    </div>
  );
}

export default App;
```

Figure 21 App.js



```
const LoginSignup = () => {
  const [state, setState] = useState("Login");
  const [formData, setFormData] = useState({
    username: "",
    password: "",
    email: "",
  });

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const login = async () => {
    console.log("Login Function Executed", formData);
    let responseData;
    await fetch("http://localhost:4000/login", {
      method: "POST",
      headers: {
        Accept: "application/form-data",
        "Content-Type": "application/json",
      },
      body: JSON.stringify(formData),
    }).then((response) => response.json()).then((data) => responseData = data);
    if (responseData.success) {
      localStorage.setItem("auth-token", responseData.token);
      window.location.replace("/");
    } else {
      alert(responseData.errors);
    }
  };
}
```

Figure 22 LoginSignUp.js

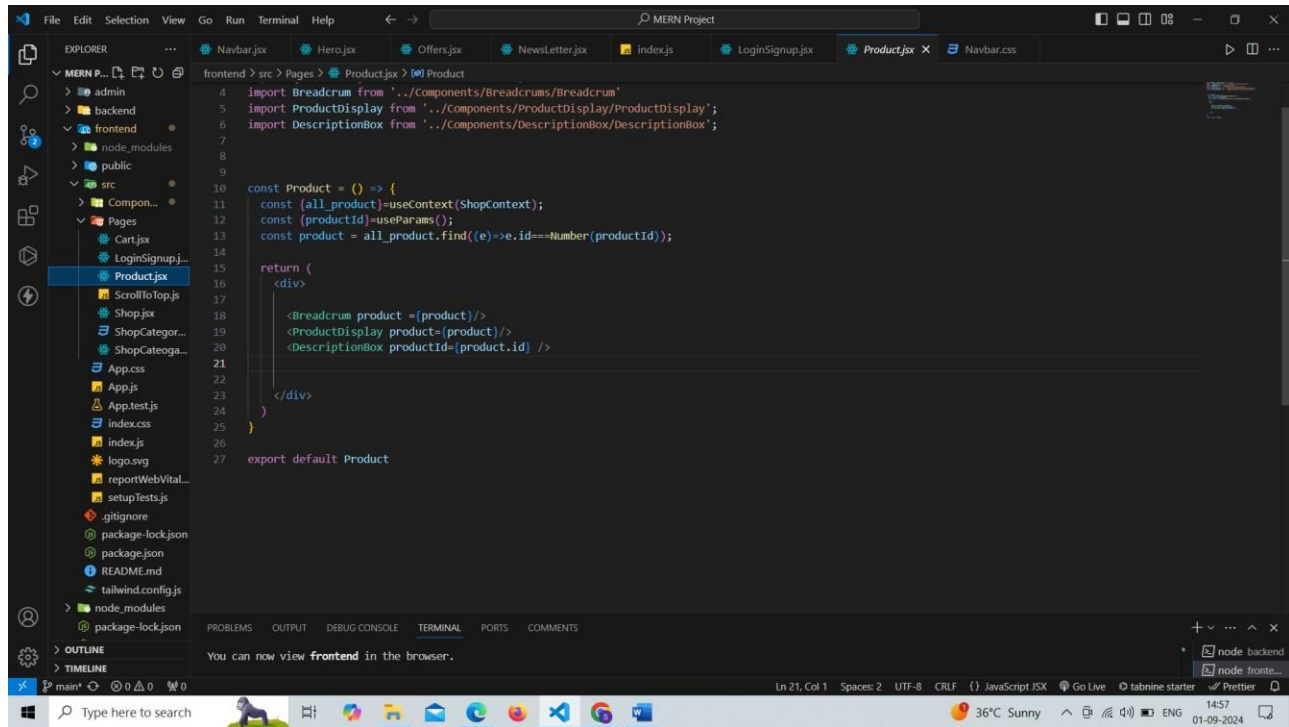


Figure 23 Product.jsx

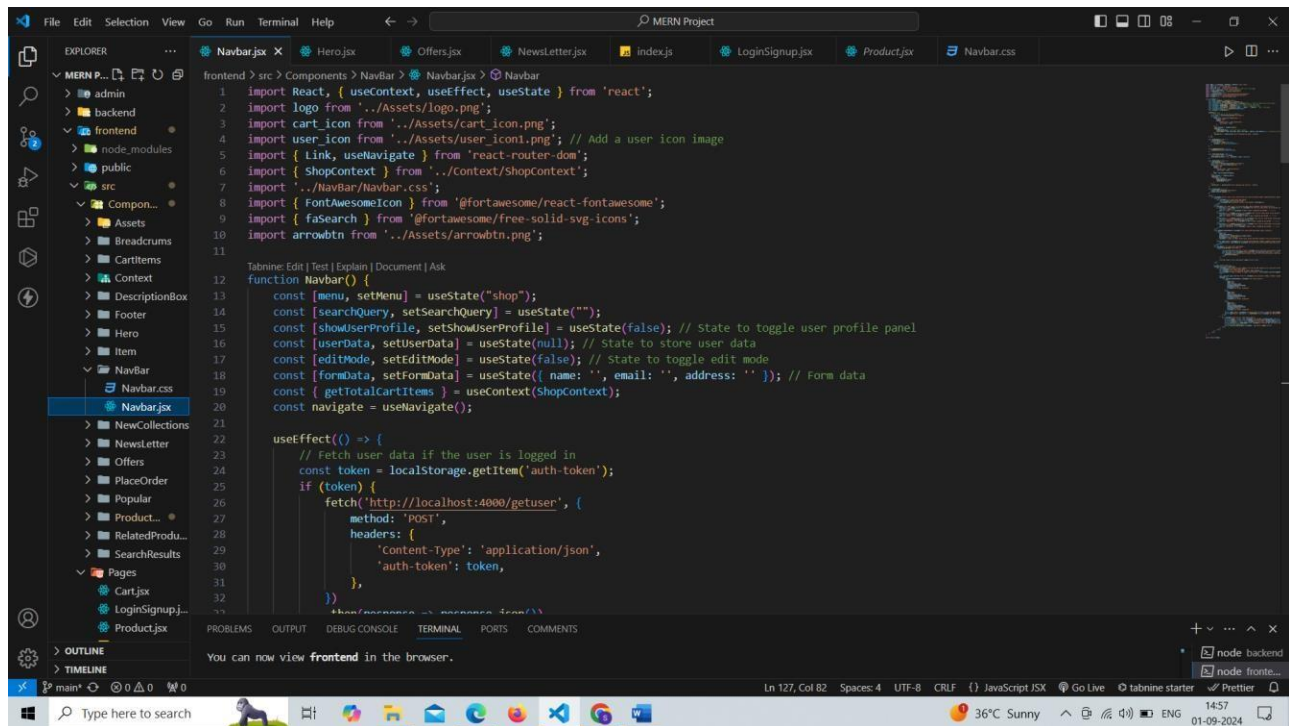


Figure 24 NavBar.jsx

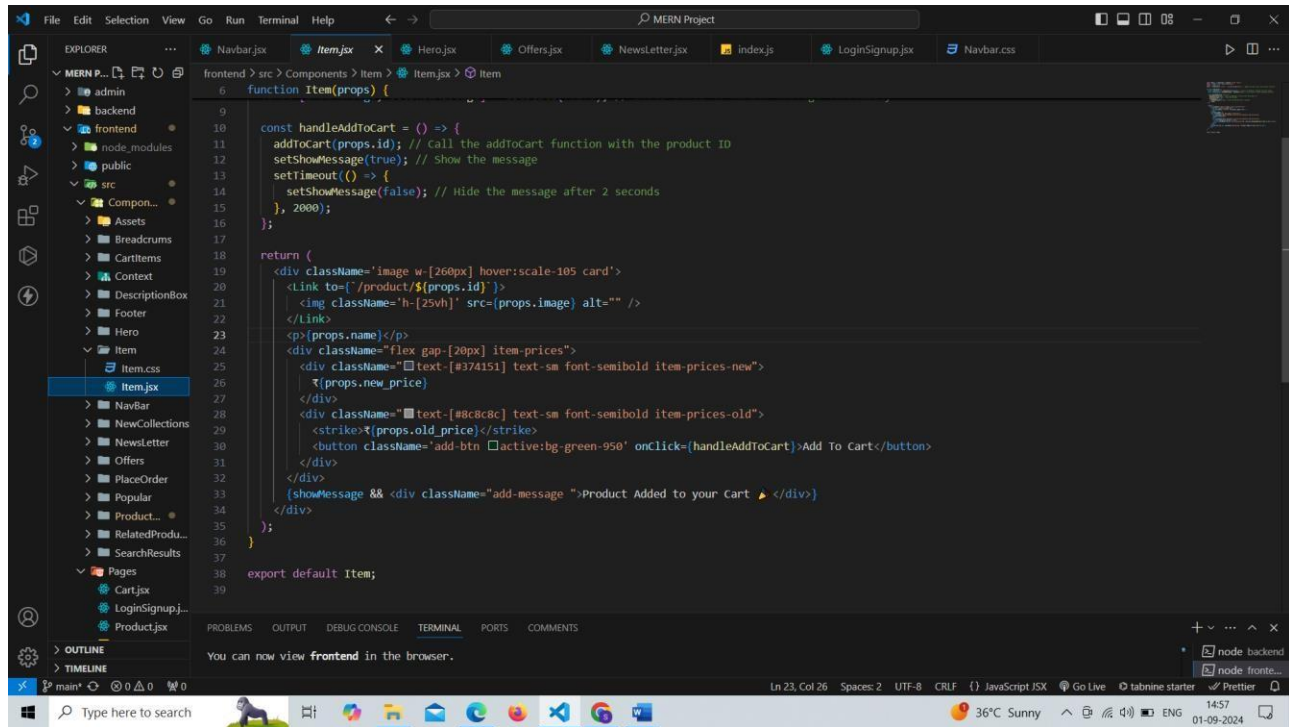


Figure 25 Item.jsx

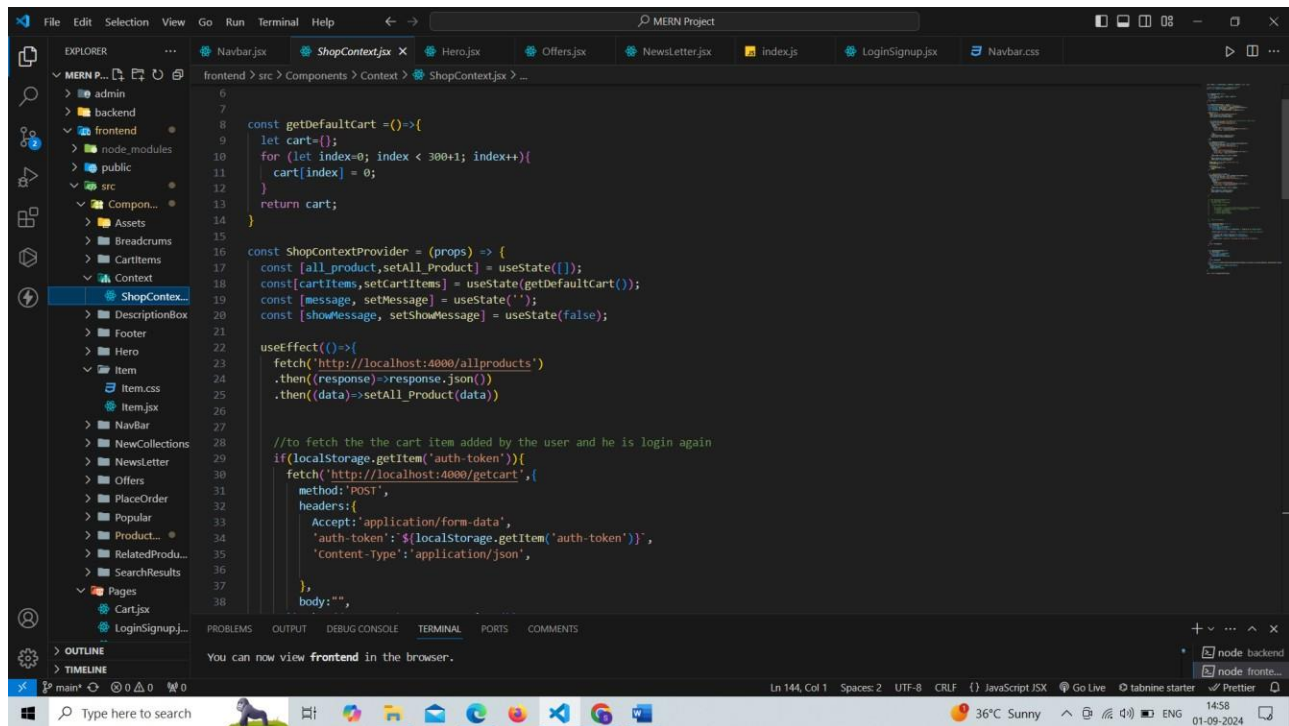


Figure 26 ShopContext.jsx

FRONTEND CODE ADMIN PANNEL

```
1 import React from 'react'
2 import { Routes, Route } from 'react-router-dom'
3 import Admin from './Pages/Admin/Admin'
4 import AddProduct from './Components/AddProduct/AddProduct'
5 import ListProduct from './Components/ListProduct/ListProduct'
6
7 const App = () => {
8   return (
9     <div>
10       <Routes>
11         <Route path="/" element={Admin}/>
12         <Route path="addproduct" element={AddProduct}/>
13         <Route path="listproduct" element={ListProduct}/>
14       </Routes>
15     </div>
16   )
17 }
18
19 export default App
```

Figure 28 App.jsx for Admin Pannel

```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.jsx'
4 import './index.css'
5 import { BrowserRouter } from 'react-router-dom'
6
7
8 ReactDOM.createRoot(document.getElementById('root')).render(
9   <React.StrictMode>
10     <BrowserRouter>
11       <App />
12     </BrowserRouter>
13   </React.StrictMode>,
14 )
```

Figure 27 Main.jsx for Admin Pannel

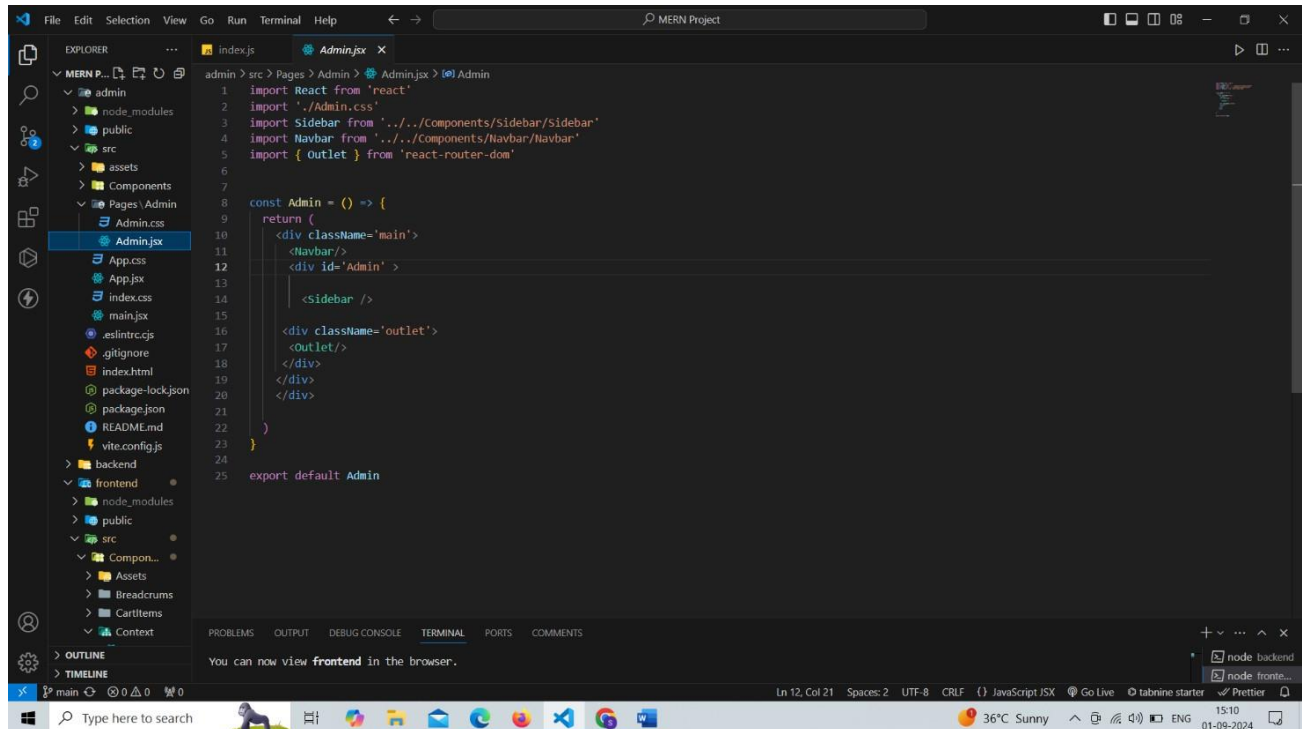


Figure 30 Admin.jsx

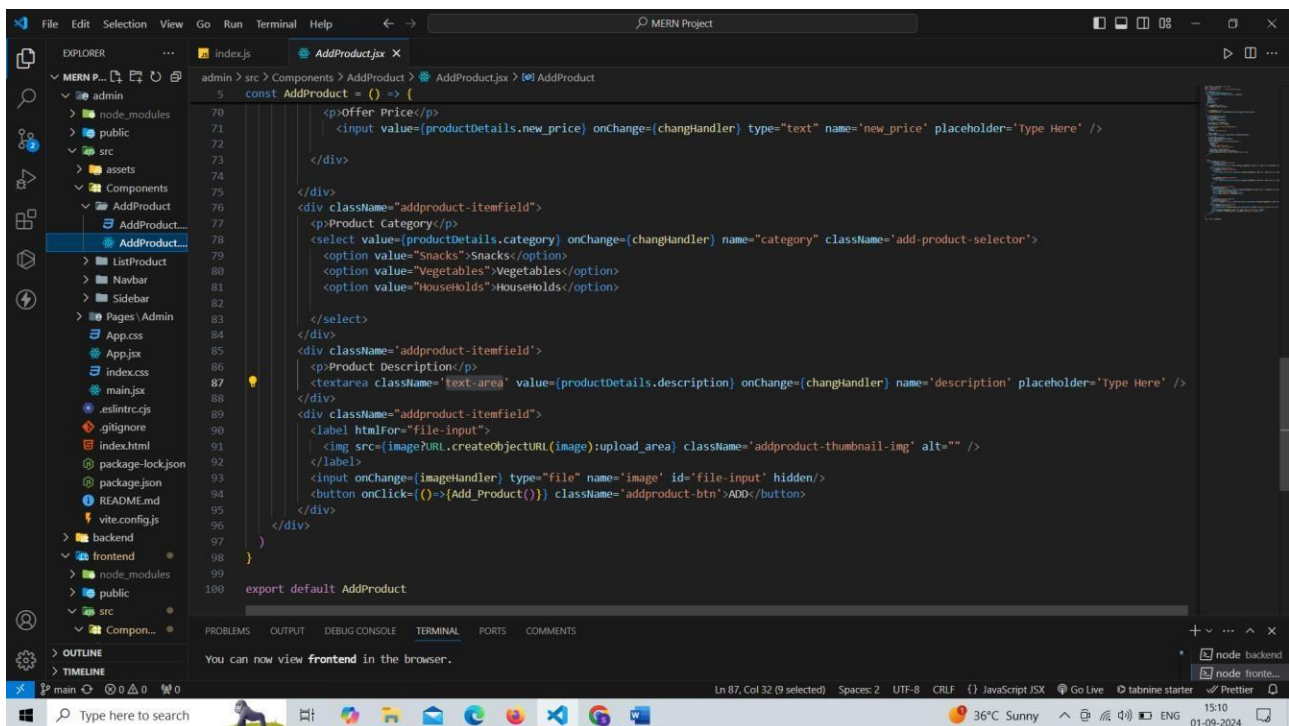


Figure 29 AddProduct.jsx for Admin Pannel

```

1  import React, { useEffect, useState } from 'react';
2  import './ListProduct.css';
3  import cross_icon from '../../assets/cross_icon.png';
4
5  const ListProduct = () => {
6    const [allProducts, setAllProducts] = useState([]);
7    const [editProductId, setEditProductId] = useState(null);
8    const [editFormData, setEditFormData] = useState({
9      name: '',
10     old_price: '',
11     new_price: ''
12   });
13
14   const fetchInfo = async () => {
15     await fetch('http://localhost:4000/allproducts')
16       .then(res => res.json())
17       .then(data => { setAllProducts(data) });
18   }
19
20   useEffect(() => {
21     fetchInfo();
22   }, []);
23
24   const remove_product = async (id) => {
25     await fetch('http://localhost:4000/removeproduct', {
26       method: 'POST',
27       headers: {
28         Accept: 'application/json',
29         'Content-Type': 'application/json',
30       },
31       body: JSON.stringify({ id: id })
32     });
33     await fetchInfo();
34   }
35 }
36
37 export default ListProduct;

```

Figure 31 ListProduct.jsx for Admin Pannel

```

1  import React from 'react'
2  import './NavBar.css'
3  import navlogo from '../../assets/logo.png'
4  import navProfile from '../../assets/nav-profile.svg'
5
6  const NavBar = () => {
7    return (
8      <div className='navbar'>
9        <div className='admin'>
10         <img src={navlogo} alt="" className='nav_logo' />
11         <h3>GROCEREAASE <span>ADMIN PANNEL</span> </h3>
12       </div>
13       <div>
14         <img src={navProfile} className='nav-profile' alt="" />
15       </div>
16     )
17   }
18
19   export default NavBar

```

Figure 32 NavBar.jsx for Admin Pannel

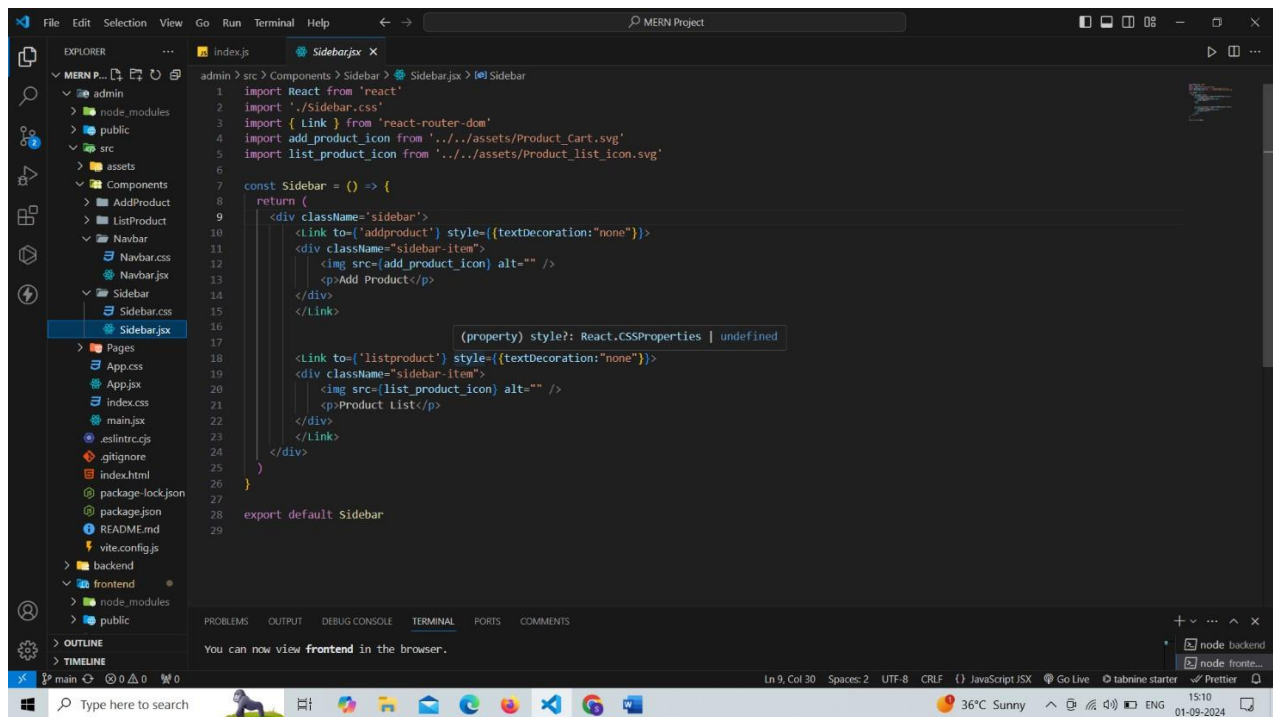
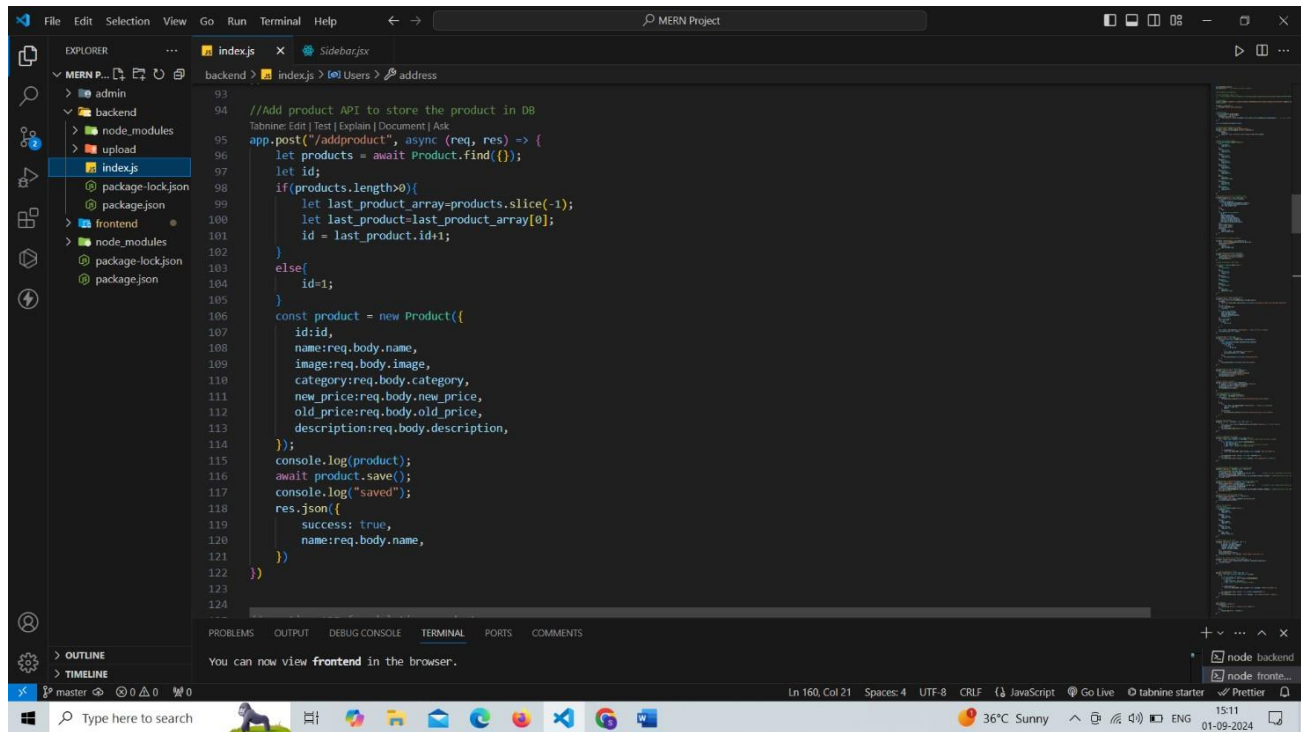


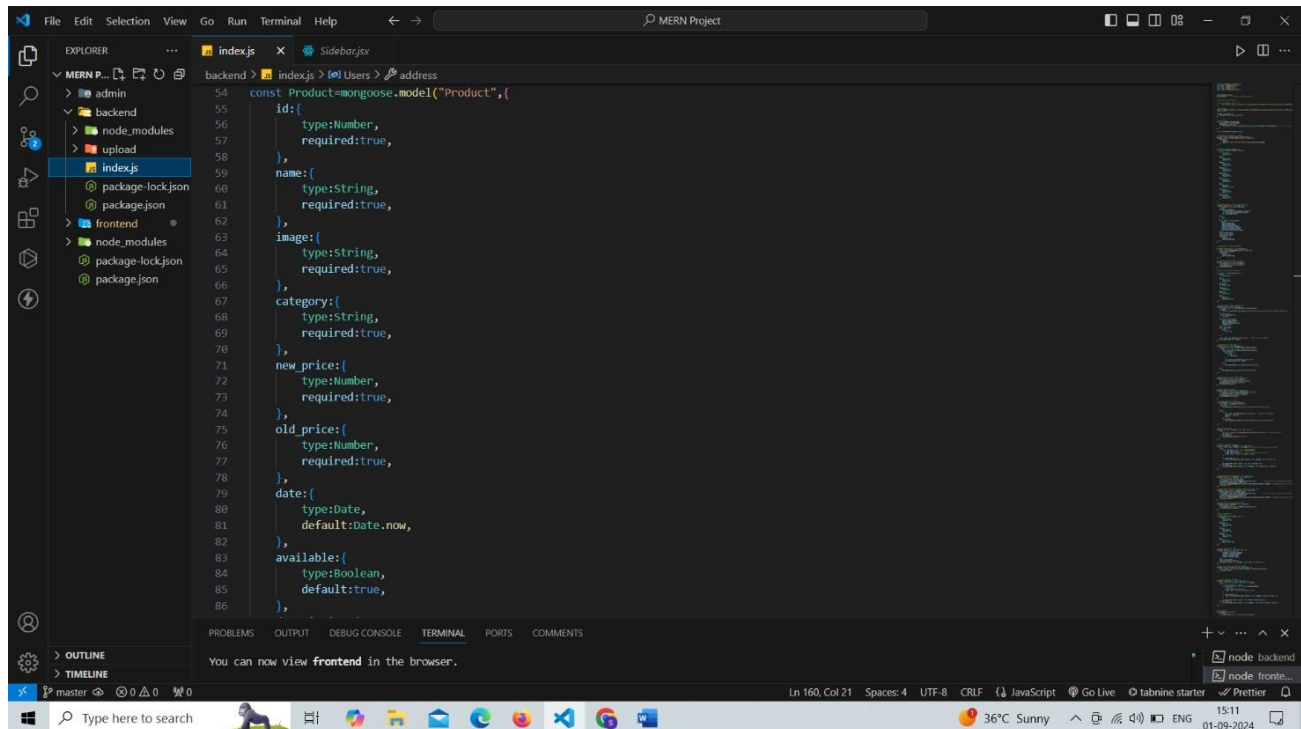
Figure 33 Sidebar.jsx for Admin Pannel

BACKEND CODE:



```
93
94 //Add product API to store the product in DB
95 app.post("/addproduct", async (req, res) => {
96   let products = await Product.find({});
97   let id;
98   if(products.length>0){
99     let last_product_array=products.slice(-1);
100     let last_product=last_product_array[0];
101     id = last_product.id+1;
102   }
103   else{
104     id=1;
105   }
106   const product = new Product({
107     id:id,
108     name:req.body.name,
109     image:req.body.image,
110     category:req.body.category,
111     new_price:req.body.new_price,
112     old_price:req.body.old_price,
113     description:req.body.description,
114   });
115   console.log(product);
116   await product.save();
117   console.log("saved");
118   res.json({
119     success: true,
120     name:req.body.name,
121   })
122 })
123
124
```

Figure 34 Index.js (Backend)



```
54 const Product=mongoose.model("Product",{
55   id:{
56     type:Number,
57     required:true,
58   },
59   name:{
60     type:String,
61     required:true,
62   },
63   image:{
64     type:String,
65     required:true,
66   },
67   category:{
68     type:String,
69     required:true,
70   },
71   new_price:{
72     type:Number,
73     required:true,
74   },
75   old_price:{
76     type:Number,
77     required:true,
78   },
79   date:{
80     type>Date,
81     default:Date.now,
82   },
83   available:{
84     type:Boolean,
85     default:true,
86   },
87 })
```

Figure 35 User Schema (Backend)