**MINOR PROJECT**

**ON**

**E-COMMERCE WEBSITE**



**A Minor Project Report Submitted to SAGE University, Indore**

**Towards Partial fulfilment for the award of**

**Master of Computer Application (MCA) degree**

**Guided by Submitted by**

PROF. PRIYA RATHORE VIKASH SATHE

22COA4MCA0224

**Institute of Computer Application**

**SAGE University, Indore**

[www.sageuniversity](http://www.sageuniversity)

|  |  |  |
| --- | --- | --- |
| **S. NO** | **CONTANTS** | **PAGE NO.** |
| 1 | Approval Sheet |  |
| 2 | Certificate |  |
| 3 | Recommendation |  |
| 4 | Acknowledgements |  |
| 5 | Candidate Declaration |  |
| 6 | Introduction |  |
| 7 | Problem Statement/Abstract |  |
| 8 | Objectives |  |
| 9 | Hypothesis |  |
| 10 | Methodology/ Project Plan |  |
| 11 | Feasibility Study |  |
| 12 | Functional requirement |  |
| 13 | Non- functional requirement |  |
| 14 | Software Requirements |  |
| 15 | Hardware Requirements |  |
| 16 | DFD Diagram |  |
| 17 | ER Diagram |  |
| 18 | Use Case Diagram / Class Diagram |  |
| 19 | Class Diagram |  |
| 20 | Database Tables |  |
| 21 | Testing |  |
| 22 | Limitations |  |
| 23 | Future Scope |  |
| 24 | Conclusion |  |
| 25 | References |  |

**INDEX**

****

**Approval Sheet**

The project entitled “**E-commerce Website**” submitted by **VIKASH SATHE** approved as partial fulfilment for the award of the **MASTER OF COMPUTER APPLICATION (MCA)** by SAGE University, Indore.

**Internal Examiner External Examiner**

**Date: Date:**

**SAGE University, Indore**



**CERTIFICATE**

This is to certify that the project work entitled “**E-COMMERCE WEBSITE**” has been carried out by **VIKASH SATHE** student of **MASTERS OF COMPUTER APPLICATION** under our supervision and guidance. They have submitted this project report towards partial fulfilment for the award of the **Master of Computer Application** by **SAGE University**, Indore.

**DR. SANJAY DUBEY PROF. PRIYA RATHORE**

**(HOD) (Supervisor)**

**RECOMMENDATION**

The project entitled “**E-COMMERCE WEBSITE**” submitted by **VIKASH SATHE** is a satisfactory account of the bona fide work done under our supervision is recommended towards partial fulfilment for the award of the **Master of Computer Application** by **SAGE University**, Indore.

**Date:**

**DR. SANJAY DUBEY RPOF. PRIYA RATHORE**

**(HOD) (Supervisor)**

**ACKNOWLEDGEMENTS**

First and foremost, I would like to express our thankfulness towards **PROF. PRIYA RATHORE** of INSTIUTE OF COMPUTER APPLICATION for extending all the facilities needed to carry out this work, I take pride in saying that I have successfully completed our Dissertation/ project work under her able guidance. She was a minor support to us throughout projects, being available at odd hours with her ideas, inspiration and encouragement. It is through her masterful guidance that I have been able to complete our Dissertation/ project work.

I am also thankful to **DR. SANJAY DUBEY (HOD),** for giving their guidance throughout the Dissertation/project phase.

**VIKASH SATHE**

**CANDIDATE DECLARATION**

I hereby declare that the work which is being presented in this project report entitled “**E-COMMERCE WEBSITE**” in partial fulfilment for the award of **Master of Computer Application** is an authentic record of my own work carried out under the supervision and guidance of **PROF. PRIYA RATHORE, SAGE University**, Indore.

I am fully responsible for the matter embodied in this report and it has not been submitted elsewhere for the award of any other degree.

**Date: VIKASH SATHE**

**INTRODUCTION**

In a world shaped by the rapid evolution of digital technologies, the realm of shopping has undergone a revolutionary transformation. This project report serves as a detailed exploration of the inception, development, and implementation of an E-Commerce Shopping platform, aimed at delivering an immersive and frictionless shopping experience to users around the globe.

**Project Background:**

The E-Commerce Shopping project encapsulates a dedicated effort to bridge the gap between traditional and digital retail, offering a comprehensive platform designed to cater to the diverse needs and preferences of modern consumers. The project sets out to not only streamline the online shopping process but to elevate it into an engaging and personalized journey for users seeking convenience, variety, and an unparalleled shopping experience.

**Key Focus Areas:**

**User-Centric Design:** At the heart of this project lies a commitment to understanding and enhancing the user experience. From intuitive navigation to visually appealing interfaces, every aspect is meticulously crafted to ensure that users can effortlessly find, explore, and purchase products in a way that mirrors the ease of an in-person shopping spree.

**Innovation in Product Discovery:** This report delves into the innovative approaches undertaken to facilitate product discovery. From personalized recommendations based on user preferences to smart search functionalities, the project seeks to redefine the way users find and engage with products in the digital marketplace.

**Security and Trust:** In an era where data security is paramount, this project places a strong emphasis on ensuring the confidentiality and integrity of user information. The report outlines the security measures in place to build and maintain trust in the online shopping environment.

**PROBLEM STATEMENT**

Enhancing the E-trade Experience with MERN Stack

The panorama of online purchasing has visible exponential boom, but several demanding situations persist in imparting a best and seamless person revel in. The existing e-commerce structures often face troubles related to scalability, user engagement, and ordinary overall performance. Our goal is to deal with these demanding situations via the improvement of a feature-rich shopping website the use of the MERN (MongoDB, Express.Js, React.Js, NodeJS) stack.

**1. Scalability Issues**

Many e-commerce structures battle to deal with a growing user base and an expanding product catalog. The lack of ability to scale successfully can result in gradual response instances, server downtimes, and a standard subpar user experience. Our challenge targets to leverage the scalability capabilities of the MERN stack to create a platform able to handling expanded consumer site visitors and an enormous array of products.

**2. User Engagement and Experience**

User engagement is a critical factor inside the success of any e-commerce platform. Existing websites frequently lack intuitive interfaces, personalized suggestions, and interactive functions that decorate user engagement. Through the implementation of React.Js in our task, we intend to create a dynamic and responsive person interface, offering an immersive and enjoyable buying revel in.

**3**. **Security Concerns**

Security remains a pinnacle precedence in on line transactions. Many e-trade platforms face demanding situations associated with facts breaches, insecure charge gateways, and inadequate consumer authentication mechanisms. Our challenge aims to implement robust safety features the usage of the MERN stack, including steady authentication protocols, encrypted statistics garage, and secure payment processing.

**4. Limited Customization and Flexibility**

Some e-trade systems lack flexibility, making it challenging for groups to tailor the internet site to their particular wishes. The inflexible architecture of present answers limits the ability to comprise custom functions or adapt to evolving market traits. The MERN stack affords a modular and flexible structure, enabling clean customization and adaptation to the precise necessities of different organizations.

**5**. **Inefficient Search and Navigation**

Finding products efficaciously is vital for person satisfaction. Many e-commerce platforms warfare with inefficient seek algorithms and navigation systems, main to a irritating shopping experience. Through the mixing of MongoDB and Express.Js, our project ambitions to implement advanced search functionalities and an intuitive navigation device to beautify the discoverability of merchandise.

**Abstract**

Modernizing E-commerce with MERN Stack

The landscape of on-line purchasing is usually evolving, imparting each possibilities and challenges. In response to the persisting troubles confronted through present e-trade structures, this task introduces an innovative solution constructed at the MERN (MongoDB, Express.Js, React.Js, NodeJS) stack. The objective is to redefine the web purchasing revel in by means of addressing scalability issues, improving person engagement, making sure robust safety features, and supplying exceptional customization.

The MERN stack, recognised for its flexibility and efficiency, bureaucracy the foundation of our e-commerce platform. MongoDB serves as the dynamic and scalable database, Express.Js enables a unbroken backend, React.Js creates an interactive and person-friendly interface, and Node.Js guarantees a excessive-overall performance server surroundings. This aggregate permits for top of the line scalability, protection, and flexibility to numerous enterprise needs.

Key attention regions include:

Scalability: Leveraging the MERN stack to create a platform able to managing elevated consumer visitors and a various product catalog.

User Engagement: Utilizing React.Js to craft a dynamic and responsive interface, offering users with an immersive and exciting purchasing experience.

Security: Implementing strong security measures, along with stable authentication, encrypted information garage, and secure charge processing.

Customization: Harnessing the ability of the MERN stack to provide companies a platform that may be tailor-made to their particular requirements and evolving market traits.

Efficient Search and Navigation: Improving product discoverability via advanced search functionalities and an intuitive navigation device.

This assignment aspires to set new benchmarks for e-commerce structures, introducing an answer that not simplest addresses contemporary demanding situations but also anticipates and adapts to the evolving wishes of on-line customers and businesses. Welcome to a destiny wherein technology meets user experience, creating a modernized and seamless on line purchasing journey.

**OBJECTIVES**

The central concept of the application is to allow the customer to shop virtually using the Internet and allow customers to buy the items and articles of their desire from the store. The information pertaining to the products are stores on an RDBMS at the server side (store).

The Server process the customers and the items are shipped to the address submitted by them. The application was designed into two modules first is for the customers who wish to buy the articles. Second is for the storekeepers who maintains and updates the information pertaining to the articles and those of the customers.

The end user of this product is a departmental store where the application is hosted on the web and the administrator maintains the database. The application which is deployed at the customer database,

the details of the items are brought forward from the database for the customer view based on the selection through the menu and the database of all the products are updated at the end of each transaction. Data entry into the application can be done through various screens designed for various levels of users. Once the authorized personnel feed the relevant data into the system, several reports could be generated as per the security.

**HYPOTHESIS**

The system after careful analysis has been identified to be presented with the following modules and roles.

The modules involved are:

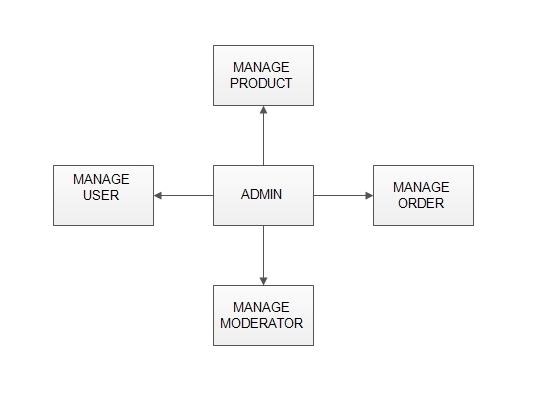
* Administrator
* Users

### ADMINISTRATOR

The administrator is the super user of this application. Only admin have access into this admin page. Admin may be the owner of the shop. The administrator has all the information about all the users and about all products.

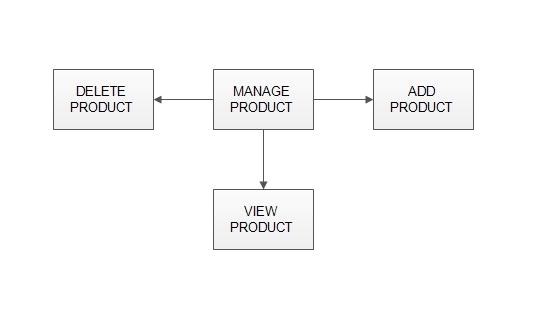
This module is divided into different sub-modules.

1. Manage Products
2. Manage Users
3. Manage Orders



Admin module

* **MANAGE PRODUCTS:**



Manage Products

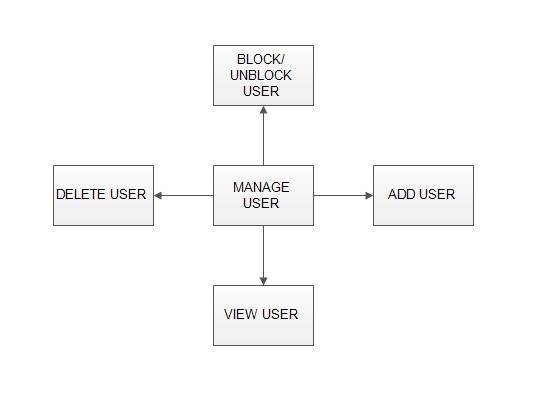
**Add Products:**

The shopping cart project contains different kind of products. The products can be classified into different categories by name. Admin can add new products into the existing system with all its details including an image.

**Delete Products:**

Administrator can delete the products based on the stock of that particular product.

* **MANAGE USER:**



Manage User

**View Users**

The admin will have a list view of all the users registered in the system.

Admin can view all the details of each user in the list except password.

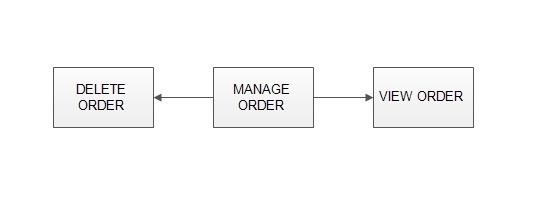
**Add Users**

Admin has privileges to add a user directly by providing the details.

#### Delete &Block Users

Administrator has a right to delete or block a user. The default status of a new user registered is set as blocked. The admin must accept the new user by unblocking him.

* **MANAGE ORDERS**



Manage Orders

#### View Order

Administrator can view the Orders which is generated by the users. He can verify the details of the purchase.

#### Delete order

Admin can delete order from the orders list when the product is taken for delivery.

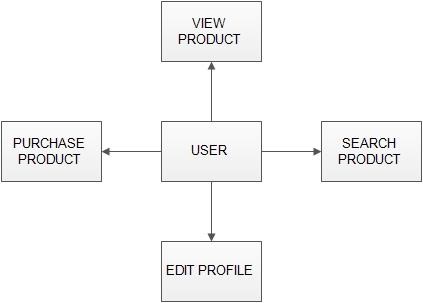
### USERS

####  Registration

A new user will have to register in the system by providing essential details in order to view the products in the system. The admin must accept a new user by unblocking him.

#### Login

A user must login with his user’s name and password to the system after registration.



User Module

#### View Products

User can view the list of products based on their names after successful login. A detailed description of a particular product with product name, products details, product image, price can be viewed by users.

#### Add to cart:

The user can add the desired product into his cart by clicking add to cart option on the product.

He can view his cart by clicking on the cart button. All products added by cart can be viewed in the cart. User can remove an item from the cart by clicking remove.

**Submit Cart:**

After confirming the items in the cart, the user can submit the cart by providing a delivery address. On successful submitting the cart will become empty.

**History**

In the history the user will have a view of pending orders.

#### Edit Profile

The user can view and edit the profile.

**METHODOLOGY**

Developing an e-commerce website using the MERN (MongoDB, Express.js, React, Node.js) stack requires a structured methodology to ensure efficient project management. Below is a sample methodology outlining the key phases and steps involved in the development process:

1. System Architecture and Design:

**Objective: Define the technical architecture and design the user interface.**

Design the database schema using MongoDB.

Define the architecture using Express.js and Node.js.

Create wireframes and UI/UX designs for the React front end.

1. Environment Setup:

**Objective: Set up the development, testing, and production environments.**

Install and configure necessary tools and dependencies.

Set up version control (e.g., Git) and collaboration tools.

3. Backend Development (Node.js and Express.js):

**Objective: Implement the server-side logic and APIs.**

Develop user authentication and authorization mechanisms.

Implement APIs for handling products, orders, and user data.

Set up server-side validation and error handling.

4. Database Integration (MongoDB):

**Objective: Connect the backend to the MongoDB database.**

Implement CRUD operations for products, users, and orders.

Optimize database queries for performance.

5. Frontend Development (React):

**Objective: Create the user interface and integrate with the backend.**

Develop React components for different pages (e.g., product listing, product details, shopping cart).

Implement state management using Redux.

Integrate with backend APIs to fetch and display data.

6. User Authentication and Authorization:

**Objective: Implement secure user authentication and authorization.**

Develop features for user registration, login, and logout.

Set up secure password storage and session management.

Define user roles and permissions.

7. Shopping Cart and Checkout:

**Objective: Develop features for shopping cart management and order checkout.**

Implement the ability to add/remove items from the cart.

Develop the checkout process with order summary and payment integration.

8. Testing:

**Objective: Conduct thorough testing of the application.**

Perform unit testing for backend and frontend components.

Conduct integration testing to ensure seamless communication.

Perform user acceptance testing (UAT) with stakeholders.

9. Training and Support:

**Objective: Provide training and ongoing support.**

Conduct training sessions for end-users and administrators.

Establish a support system for addressing user issues and inquiries.

10. Continuous Improvement:

**Objective: Iteratively improve the website based on feedback and analytics.**

Collect user feedback and prioritize feature enhancements.

Plan and implement regular updates to enhance functionality and user experience.

This methodology provides a structured approach to developing an e-commerce website using the MERN stack. Adjust the steps and timelines based on the specific requirements and constraints of your project. Regularly review and iterate on the process to ensure continuous improvement.

**PROJECT PLAN**

1. Project Initiation:

Define project objectives and scope.

Identify stakeholders and establish communication channels.

Set up version control (e.g., Git) and collaboration tools.

2. Requirements Gathering:

Conduct market research to understand the target audience and competition.

Document functional and non-functional requirements.

Define user stories and use cases.

3. Architecture and Design:

Define the database schema using MongoDB.

Design the application architecture using Express.js and Node.js.

Create wireframes and UI/UX designs for the front end using React.

4. Environment Setup:

Set up the development, testing, and production environments.

Install and configure necessary tools and dependencies.

5. Backend Development (Node.js and Express.js):

Implement user authentication and authorization.

Develop APIs for handling products, orders, and user data.

Set up server-side validation and error handling.

6. Database Integration (MongoDB):

Connect the Node.js backend to MongoDB.

Implement CRUD operations for products, users, and orders.

Optimize database queries for performance.

7. Frontend Development (React):

Create React components for different pages (e.g., product listing, product details, shopping cart).

Implement state management using tools like Redux.

Integrate with backend APIs to fetch and display data.

8. User Authentication and Authorization:

Implement user registration, login, and logout functionality.

Set up secure password storage and session management.

Define user roles and permissions.

9. Shopping Cart and Checkout:

Develop features for adding/removing items from the cart.

Implement the checkout process with order summary and payment integration.

10. Testing:

Conduct unit testing for backend and frontend components.

Perform integration testing to ensure seamless communication between frontend and backend.

Conduct user acceptance testing (UAT) with stakeholders.

**FEASIBILITY STUDY**

A feasibility study for an e-commerce website using the MERN (MongoDB, Express.js, React, Node.js) stack involves assessing the technical, operational, economic, and scheduling aspects of the project. Here's a sample outline for a feasibility study:

Executive Summary:

1. Project Description:

Provide a brief overview of the proposed e-commerce website.

Outline the key features and functionalities.

2. Objective of the Feasibility Study:

Define the purpose of the study, whether to proceed with the MERN stack for the e-commerce project.

Technical Feasibility:

3. Technology Stack:

Evaluate the suitability of the MERN stack for the project.

Assess the availability of skilled developers with MERN expertise.

4. Scalability:

Analyze whether the chosen technologies can scale with the anticipated growth of the website.

Consider potential bottlenecks and scalability solutions.

5. Integration:

Assess the feasibility of integrating with third-party services (e.g., payment gateways, shipping providers).

Ensure compatibility with existing systems if applicable.

6. Security:

Evaluate the security features provided by the MERN stack.

Identify potential security risks and outline strategies for mitigation.

Operational Feasibility:

7. User Experience:

Assess the user-friendliness of the proposed e-commerce website.

Consider user feedback and usability testing.

8. Maintenance:

Evaluate the ease of maintaining and updating the MERN-based application.

Consider tools and processes for ongoing maintenance.

9. Training:

Determine the training requirements for administrators and end-users.

Assess the availability of resources for training.

Economic Feasibility:

10. Cost Estimation:

Estimate the costs associated with development, infrastructure, and ongoing maintenance.

Consider licensing fees, hosting costs, and personnel expenses.

11. Return on Investment (ROI):

Project the potential revenue generated by the e-commerce website.

Compare the projected ROI with the estimated costs.

12. Risk Analysis:

Identify potential risks that could impact the project's economic feasibility.

Develop mitigation strategies for high-risk areas.

Schedule Feasibility:

13. Timeline:

Develop a detailed project timeline from initiation to deployment.

Identify critical milestones and dependencies.

14. Resource Availability:

Assess the availability of human resources and any potential bottlenecks.

Consider outsourcing options if necessary.

**FUNCTIONAL REQUIREMENT**

**USER**

**USER LOG-IN**

##### Description of feature

This feature used by the user to login into system. A user must login with his user name and password to the system after registration. If they are invalid, the user not allowed to enter the system.

##### Functional requirement

* Username and password will be provided after user registration is confirmed.
* Password should be hidden from others while typing it in the field

**REGISTER NEW USER**

##### Description of feature

A new user will have to register in the system by providing essential details in order to view the products in the system. The admin must accept a new user by unblocking him.

##### Functional requirement

* System must be able to verify and validate information.
* The system must encrypt the password of the customer to provide security.

 PURCHASING AN ITEM

#### ADMIN

 MANAGE USER

##### Description of feature

The administrator can add user, delete user, view user and block user.

MANAGE PRODUCTS

##### Description of feature

The administrator can add product, delete product and view product.

 MANAGE ORDERS

Description of feature

The administrator can view orders and delete orders.

##### Functional requirements

-The system must identify the login of the admin.

-Admin account should be secured so that only owner of the shop can access that account

**NON-FUNCTIONAL REQUIREMENT**

* EFFICIENCY REQUIREMENT

When an online shopping cart android application implemented customer can purchase product in an efficient manner.

* RELIABILITY REQUIREMENT

The system should provide a reliable environment to both customers and owner. All orders should be reaching at the admin without any errors.

* USABILITY REQUIREMENT

The android application is designed for user friendly environment and ease of use.

* IMPLEMENTATION REQUIREMENT

Implementation of the system using css and html in front end with jsp as back end and it will be used for database connectivity. And the database part is developed by mysql. Responsive web designing is used for making the website compatible for any type of screen.

* DELIVERY REQUIREMENT

The whole system is expected to be delivered in four months of time with a weekly evaluation by the project guide.

**SOFTWARE REQUIREMENT**

Software requirements for a Car Rental Management System built using Node.js for the backend, MongoDB for the database, and React for the frontend involve specifying the tools, libraries, and frameworks necessary for development, testing, and deployment. Here are key software requirements for each component:

**Backend (Node.js):**

**1. Node.js:**

- Version:18.0.1 The specific version of Node.js to be used for development.

- Installation: for installing first download it form web browser a package must be in msi mode After that when you download it and install It

Ones its installed create a folder name it as you like

For stablishing a node server I have used a command npm init

After that our node js setup has been stablished

**2. Express.js:**

- Version: 6.0 The version of the Express.js framework for building the backend.

For stablish express js I have used npm I express

**3. MongoDB:**

- Version: 7.0he version of MongoDB for data storage.

- Connection: I have used a MongoDB atlas for storage in which I have establish a connection through link provided by Mongo Alas

**4. Mongoose:**

- Version: 3.0of the Mongoose library for MongoDB schema validation and interaction. - Schema: I mark the schema of users cars and for login details the data models and schema for the MongoDB collections.

**Frontend (React):**

**1. React**:

- Version: 10.0 is the version of React for building the user interface.

- Create React App: I have use a npx create-react-app public for create and install a libraries Of react

**2. React Router:**

- Version: 6.0 of the React Router library for handling navigation within the React app. To intall any dependency just use npm I and dependency name

I have used this dependencies version in my car rental system



Backend dependencies



Frontend dependencies

**Data base MongoDB:**

**MongoDB Compass:**

-I have connected MongoDB compass as a GUI tool and for connection I have used link provided by atlas

**Code Editor:**

To write a code and edit it I have use Visual studio code and for running a Command I have used a common terminal

**HARDWARE REQUIREMENT**

Processor: Pentium-III (or) Higher

Ram: 64MB (or) Higher

Hard disk: 80GB (or) Higher

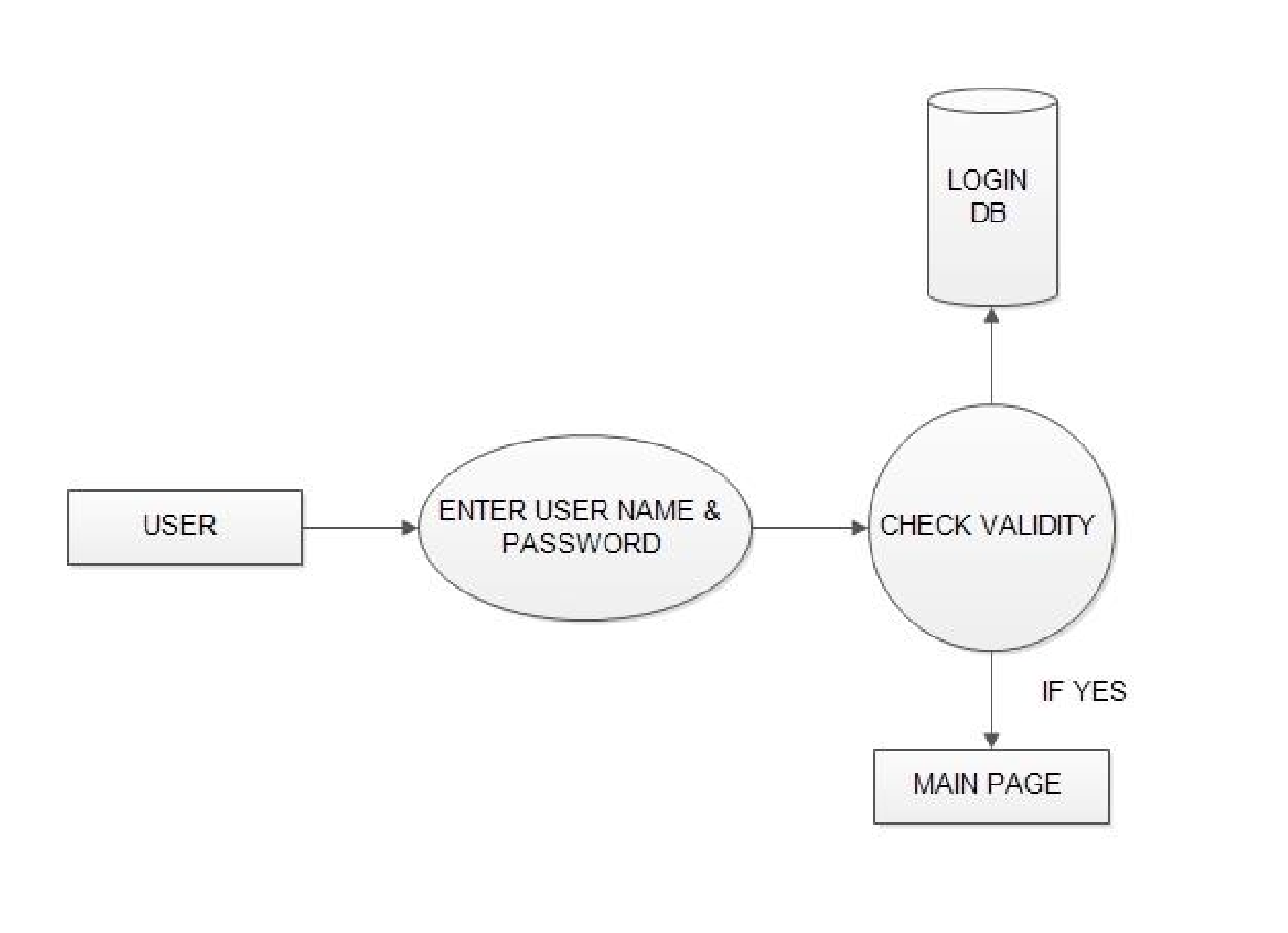
Network:

Bandwidth:

Adequate network bandwidth to handle incoming and outgoing traffic.

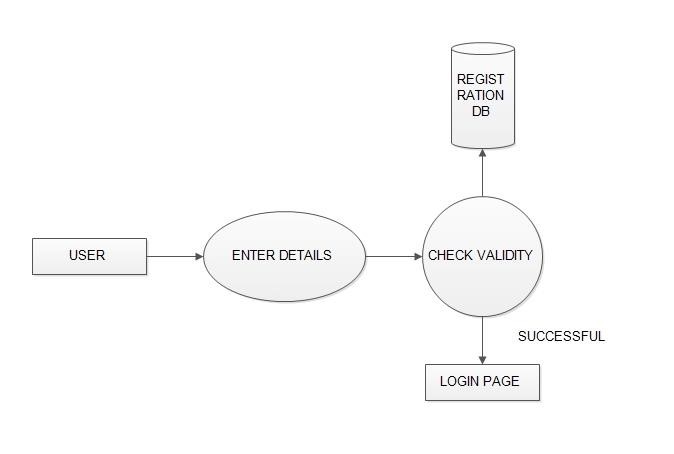
Consideration for increased bandwidth in case of high user load.

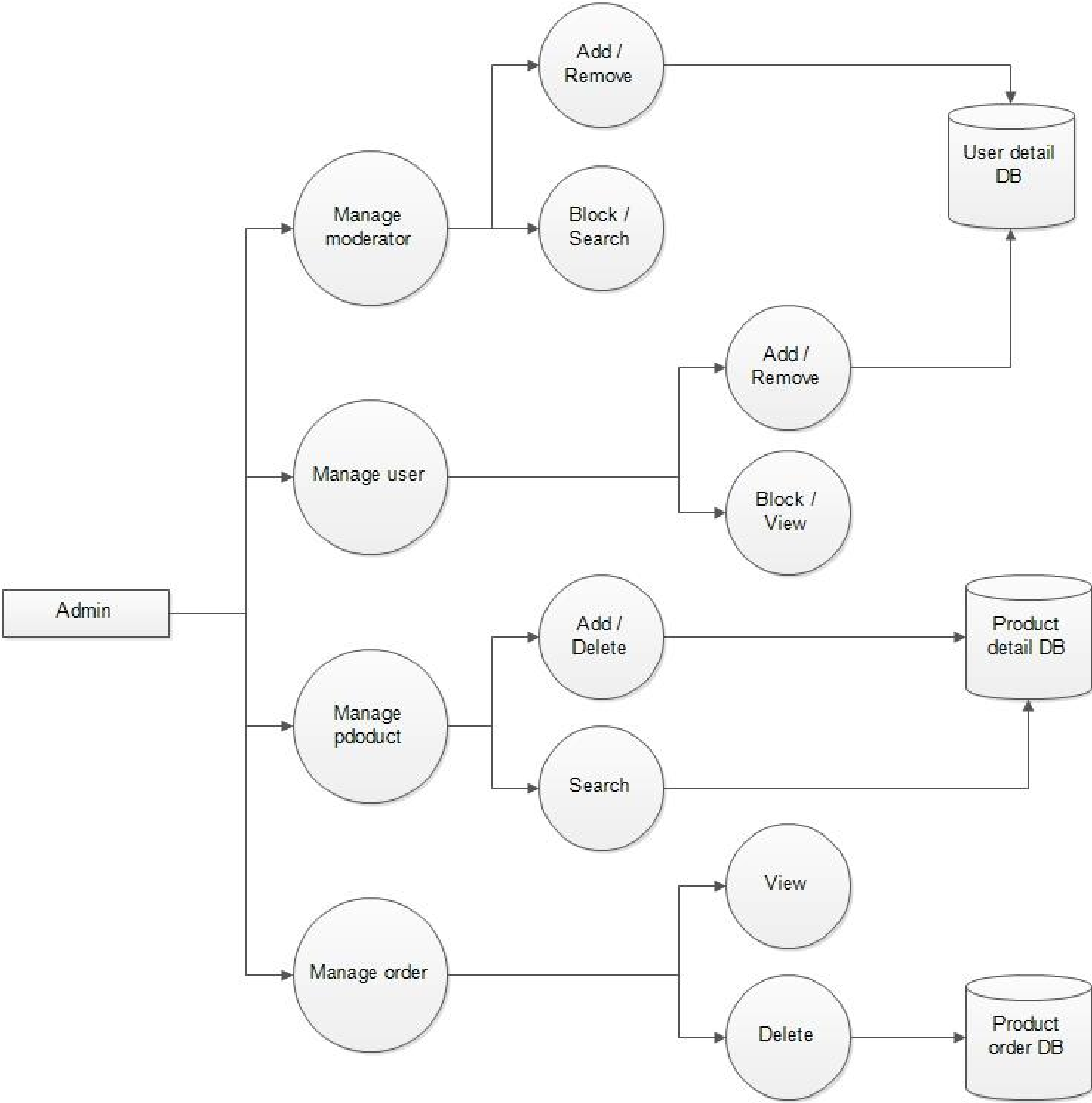
**DFD DIAGRAM**

* LOGIN DFD

Login DFD

* RAGISTRASTION DFD

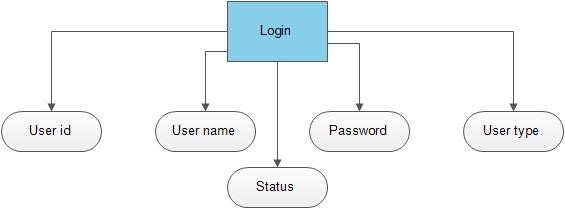
Registration DFD

* ADMIN FDF

Admin DFD

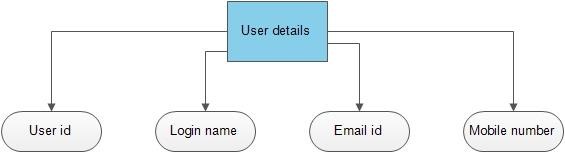
**ER DIAGRAM**

* LOG-IN



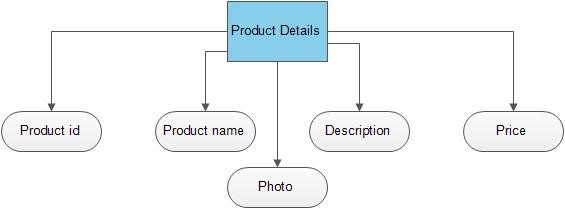
Login ER

* USER DETAILS



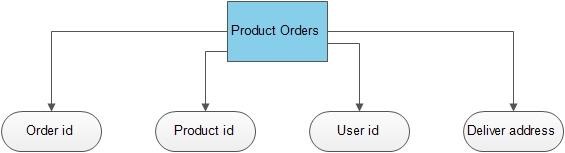
User Details ER

* PRODUCT DETIALS



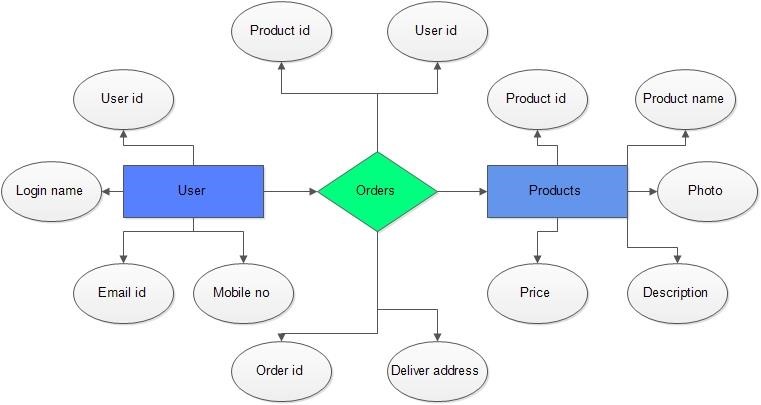
Product Details ER

* PRODUCT ORDERS



Product Orders ER

* COMPLITE DIAGRAM



Complete ER

**DATAFLOW**

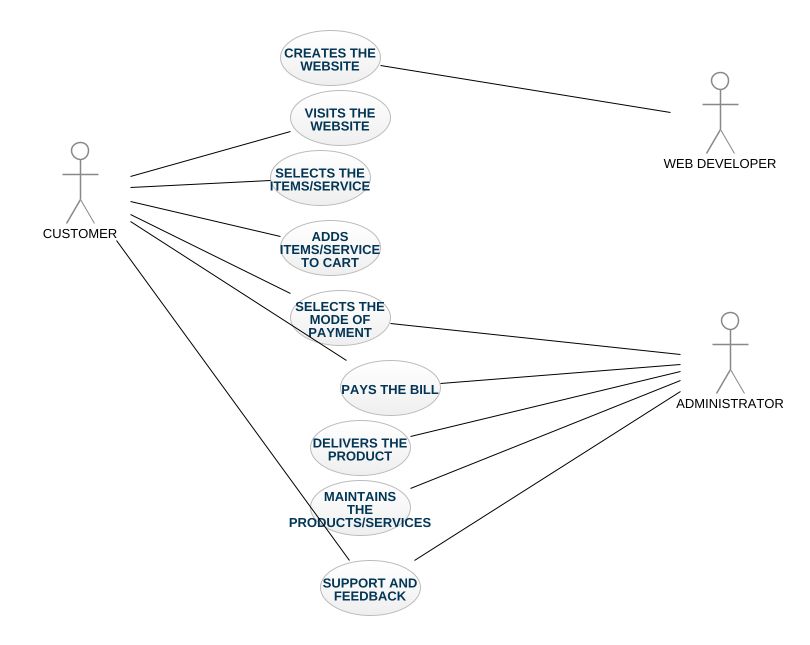
The data flow shows the flow of information from a source to its destination. Data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal or electronic. Each data flow may be referenced by the processes or data stores at its head and tail, or by a description of its contents.

**USECASE DIAGRAM**

Use case diagram consists of use cases and actors and shows the interaction between them. The key point is:

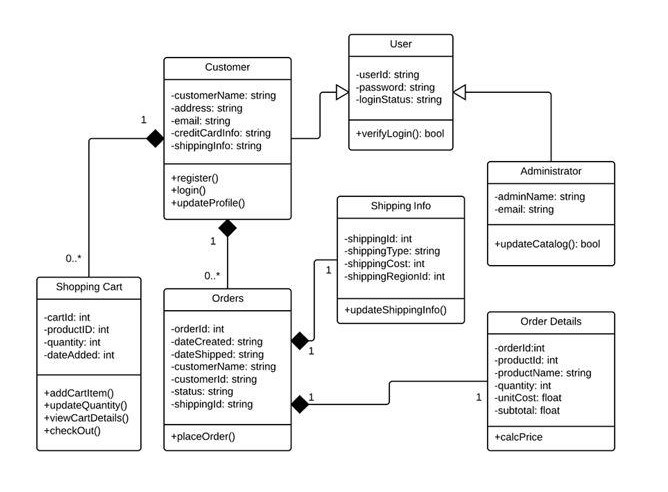
The main purpose is to show the interaction between the use cases and the actor.

To represent the system requirement from user’s perspective.



USECASE DIAGRAM

**CLASS DIAGRAM**



CLASS DIAGRAM

**DATABASE TABLE**

have use MongoDB in my E-COMMERCE SHOPPING WEBSITE (the database design will differ from a traditional relational database. MongoDB is a NoSQL database that stores data in BSON (Binary JSON) format, and it uses collections instead of tables. Below is a simplified outline of how your MongoDB database structure might look for a CRMS:

1. **USER COLLECTION**

{

"\_id":

{

"$oid":"658bb4f5b73d1522170604bb"

},

"firstName":"first name",

"lastName":"last name",

"password":"hashed\_password",

"email":"gmail@gmail.com",

"role":"customer",

"address":[],

"paymentInformation":[],

"ratings":[],

"reviews":[],

"createdAt":

{

"$date":{"$numberLong":"1703516571300"}},"\_\_v":{"$numberInt":"0"

}

}

1. **Cart Details**

{

"\_id":

{

"$oid":"658bb61fb73d1522170604bc"

},

"user":

{

"$oid":"658979d7f7236beff8130dc9"

},

"cartItems":[],

"totalPrice":

{

"$numberInt":"0"

},

"totalItem":

{

"$numberInt":"0"

},

"totalDiscountPrice":

{

"$numberInt":"0"

},

"discounte":

{

"$numberInt":"0"

},

"\_\_v":

{

"$numberInt":"0"

}}

Products Details

{

    {

    "\_id": "569549sd44fg54894fg85",

    "firstName":"first name",

    "lastName": "last name",

    "password": "hashed\_password",

    "email": "gmail@gmail.com",

    "role":[],

    "address"[],

    "paymentInformation": [],

    "ratings"[],

    "reviews":[],

    "createdAt"[]

    }

    "orderItems":{

        "\_id":"456456fgf4564hngv5ygd",

        "title":"woman Printed Pure Cotton Straigth Kurta",

        "discription": "tradition kurta for womans",

        "price": 1999,

        "discountedPrice": 599,

        "discountedPeresent":40,

    }

}

**TESTING**

Testing is an important aspect of software development that ensures the quality, reliability, and performance of your E-commerce Website site.

Below are the different types of tests to consider for both React frontends and Node.js backends:

Frontend Tests (React):

1. Unit Tests: - Libraries: Use testing libraries such as Jest or React Testing Library.

- Components: Test individual React components to ensure they render correctly and behave as expected.

2. Integration Testing: - Component Interaction: Verify that the various components work together as intended.

- API Call: Test the integration of the component with the backend API.

3. Snapshot Testing: - UI Changes: Use snapshot testing to capture a visual representation of a component and detect unintended UI changes.

4. End-to-end testing: - User flow: Test the entire user flow by simulating user interactions.

- Browser Automation: Tools like Cypress and Selenium can be used for end-to-end testing.

5. Accessibility Test: - Screen Reader: Test your screen reader to make sure you can access your website.

- AX or Lighthouse: Use tools like AX or Lighthouse for automated accessibility testing.

Backend testing (Node.js):

1. Unit Tests: - Test Frameworks: Test individual functions and modules using test frameworks such as Mocha and Jest.

- Mocks: Isolate your tests by mocking external dependencies such as databases and external APIs.

2. Integration Testing: - API Endpoints: Test the integration of your API routes to ensure that requests and responses are handled correctly.

- Database interaction: Validates interaction with the database.

3. Security Testing: - Input Validation: Ensure input validation is in place to avoid security vulnerabilities.

- Authentication and Authorization: Test security mechanisms for user authentication and authorization.

4. Performance Testing: - Load Testing: Use tools like Apache JMeter and Artillery to simulate high loads and identify performance bottlenecks.

- Stress Test: Evaluates how a system handles extreme stress or resource depletion.

5. Database Test: - Data Integrity: Verify that data is correctly saved to and retrieved from the MongoDB database.

- Query Performance: Evaluates the performance of database queries.

6. Testing your API documentation: - Swagger/Open API: If you have API documentation, make sure it accurately represents your API endpoints and their behavior.

- Postman or Insomnia: Manually test your API endpoints using tools such as Postman or Insomnia.

7. Error Handling Test: - Error Response: Test how the backend handles errors and returns an appropriate response.

- Edge cases: Ensure that the backend properly handles edge cases and unexpected scenarios

8. Continuous integration (CI) and continuous deployment (CD) pipelines: - Automated builds: CI/CD pipelines to automate testing and deployment processes Set up.

- Code quality checks: Integrate tools like ESLint and Prettier to maintain code quality.

Don't forget to run tests regularly, ideally as part of your development workflow and CI/CD pipeline, to catch issues early and ensure the continued stability of your E-commerce Website site.

**LIMITATIONS**

Building a E-commerce Website site using React on the frontend, Node.js on the backend, and MongoDB as the database is a powerful and scalable solution.

However, like any technology stack, there are limitations and considerations to keep in mind.

1. Performance: - Client-side rendering (CSR): React primarily uses CSR.

, this can affect the

initial load time of the page.

Consider implementing server-side rendering (SSR) or using a framework like Next.js to improve performance.

- API Requests: Too many API requests can degrade performance.

Implement effective data retrieval strategies and caching mechanisms to optimize user experience.

2. Security: - Authentication and Authorization: Ensure secure user authentication and authorization mechanisms.

It uses a secure method for authentication, such as JWT (JSON Web Token), and handles authorization on the server side.

- Input Validation: Always validate and sanitize user input to prevent common vulnerabilities such as SQL injection and cross-site scripting (XSS).

3. Scalability: - Horizontal Scaling: Node.js allows horizontal scaling by adding instances to handle increased load.

However, consider possible bottlenecks within your architecture.

- Database Scaling: MongoDB can also be scaled horizontally, but keep in mind that there may be limitations with large data sets or high transaction volumes.

4. Data Integrity: - Transactions: MongoDB does not support multi-document transactions in all situations.

Make sure your application can handle scenarios where data consistency is important.

- Data Modeling: Carefully design your MongoDB schema to avoid common pitfalls such as excessive use of embedded documents.

5. SEO (Search Engine Optimization): - SSR for SEO: If SEO is a priority, make sure search engines properly recognize your Can Consider using server-side rendering (SSR).

Index website.

6. Testing and Maintenance: - Unit Testing: Create comprehensive unit tests for both React components and Node.js backends to diagnose problems identified early.

To do.

- Dependency Management: Update dependencies regularly to address security vulnerabilities and benefit from performance improvements.

7. User Experience: - Responsive Design: Make sure your E-commerce Website is responsive and provides a great user experience on a variety of devices.

- Accessibility: Implement accessibility best practices to make our website available to everyone, including people with disabilities.

8. Payments and Transactions: - Security Compliance: Follow industry standards for security (such as PCI DSS compliance) when processing payments.

- Transaction Processing: Implement robust transaction processing to ensure the integrity of financial transactions.

9. Legal Compliance: - Privacy: Comply with data protection regulations, especially when dealing with sensitive user information.

10. Documentation: - Documentation: Maintain thorough documentation of front-end and back-end code to facilitate future development and bug fixes.

Stay up to date on the latest best practices and security measures for the technologies you use.

Regularly monitor and analyze performance to identify and address potential bottlenecks and improvements to your E-commerce Website site.

**FUTURE SCOPE**

The future scope of E-commerce Website sites built with React, Node.js, and MongoDB will change dynamically as several trends and opportunities emerge.

Here are some areas with potential for growth and development:

1. Progressive Web Applications (PWA): Improve your shopping experience by converting your website to a PWA.

PWAs offer users improved performance, offline functionality, and a more app-like experience.

2. Microservices Architecture: Consider breaking your application into microservices to improve scalability, maintainability, and flexibility.

Each microservice can handle a specific functionality such as user management, inventory, or order processing

3. Artificial Intelligence (AI) Integration: - Personalization: Uses AI algorithms to analyze user behavior and preferences to provide personalized product recommendations and provide you with a customized shopping experience.

- Chatbot: Implement an AI-powered customer support chatbot that helps users find products, track orders, and answer frequently asked questions.

4. Blockchain technology: - Supply chain transparency: Discover blockchain to improve supply chain transparency, ensure authenticity, and improve product life.

Prevents counterfeiting in cycles.

– Secure Transactions: Consider blockchain for secure and transparent financial transactions, especially when dealing with cryptocurrencies.

5. Augmented reality (AR) and virtual reality (VR): - Virtual try-on: Add AR to virtual try-on to help users visualize products such as clothing Implement it.

, it looks like you've viewed the furniture in a real environment before purchasing it.

- Virtual Shopping Environment: Create an immersive VR shopping experience for your users, giving them a unique and engaging way to browse and purchase products.

6. Voice Commerce: - Voice Search: Enabling the voice search feature improves the user experience and allows you to use voice commands to search for products, you will be able to order.

7. Cross-platform development: - React Native: Develop a mobile app version of your E-commerce Website using a framework like React Native to expand your reach and Wider audience.

8. Data Analytics and Business Intelligence: - Data-Driven Decision Making: Leverage data analysis tools to gain insights into user behavior, preferences, and market trends.

Enable informed business decisions.

- Predictive Analytics: Use predictive analytics to predict demand, optimize inventory, and personalize marketing strategies.

9. Subscription models and loyalty programs: - Subscription services: Subscription-based models for products or services to increase customer loyalty and generate recurring revenue Please consider adopting.

- Loyalty Programs: Implement and optimize loyalty programs to reward repeat customers and encourage brand loyalty.

10. Environmental and Social Responsibility: - Sustainability Practices: We embrace sustainability and ethical business practices.

Communicate your commitment to environmental and social responsibility as conscious consumption continues to rise

11. Integration with new technologies: - 5G technology: How will the introduction of 5G technology impact application performance, especially in terms of faster load times and improved user experience?

Check what you give.

Remember that flexibility and adaptability to new technologies and consumer trends are essential to the long-term success of online shopping platforms.

Stay ahead of the curve by regularly evaluating market trends, user feedback, and technological advances.

**CONCLUSION**

The project entitled Online shopping system was completed successfully.

The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. The purpose of this project was to develop a web application and an android application for purchasing items from a shop.

This project helped us in gaining valuable information and practical knowledge on several topics like designing web pages using html & css, usage of responsive templates, designing of android applications, and management of database using MongoDB. The entire system is secured. Also, the project helped us understanding about the development phases of a project and software development life cycle. We learned how to test different features of a project.

This project has given us great satisfaction in having designed an application which can be implemented to any nearby shops or branded shops selling various kinds of products by simple modifications.

There is a scope for further development in our project to a great extent. A number of features can be added to this system in future like providing moderator more control over products so that each moderator can maintain their own products. Another feature we wished to implement was providing classes for customers so that different offers can be given to each class. System may keep track of history of purchases of each customer and provide suggestions based on their history. These features could have implemented unless the time did not limited us.

**REFERENCES**

[**https://react.dev**](https://react.dev)

**react-redux.js.org**

[**https://tailwindcss.com/**](https://tailwindcss.com/)

**https://tailwindcomponents.com/**

**mongodb.com/docs.manual.reference/database-references/**

[**https://nodejs.org**](https://nodejs.org)