Project Report

ShopeZ E-commerce Application.

Team Lead: JOTHILAKSHMANAN N

Team Member:

ELAMARAN M
KARTHIKEYAN K
ARCHUNA PANDIYAN A

Project Introduction:

In today's digital age, e-commerce platforms are crucial for businesses to reach a wider audience and facilitate online transactions. The Shopez E-Commerce Application is designed to provide a seamless online shopping experience for users, with features such as product browsing, secure transactions, and user-friendly navigation. The application leverages the MERN stack, a robust framework combining MongoDB, Express.js, React, and Node.js, to build a dynamic, scalable, and high-performance e-commerce solution.

Technology used:

ShopeZ E-commerce Application.

"Experience seamless shopping with Shopez, an innovative e-commerce platform powered by the robust MERN stack. Fast, responsive, and user-friendly, Shopez offers a dynamic online shopping experience with real-time updates, secure transactions, and personalized recommendations—all driven by MongoDB, Express.js, React, and Node.js for unmatched performance and scalability."

Development for ShopeZ E-commerce Application:

"Build your dream online store with the Shopez E-Commerce Application, powered by the robust MERN stack! From seamless user experiences to powerful backend performance, Shopez offers a cutting-edge solution for businesses looking to sell smarter, faster, and more efficiently. Ready to scale your e-commerce game? Shopez is the future of online shopping."

Folder Structure:

client/: This folder contains all the React code for the frontend. It will include:

- components/: Reusable UI components (e.g., buttons, headers).
- pages/: React components for different pages of the application (e.g., Home, Product Detail, Cart, etc.).
- redux/: Redux-related files like actions, reducers, and the store configuration to manage global app state.
- assets/: Store static files such as images and fonts.

server/: This folder contains the backend Express server code.

- controllers/: Contains the business logic for handling requests.
- models/: Mongoose models for interacting with MongoDB (e.g., Product, Order, User).
- routes/: API routes (e.g., for product listings, user authentication).
- middleware/: Middleware for things like authentication or error handling.
- config/: Configuration files for MongoDB, JWT, etc.

- **Shared Configurations**: The root package.json handles both client and server dependencies, and the .env files store sensitive information like API keys and database URIs.

Pre Requisites:

```
shopez-ecommerce-app/
│
├── client/              # React frontend code
│   ├── public/            # Static files
│   │   └── index.html
│   ├── src/               # React source code
│   │   ├── assets/         # Images, fonts, etc.
│   │   ├── components/      # Reusable UI components (buttons, form
inputs, etc.)
│   │   ├── pages/          # React components for different pages
(home, products, cart, etc.)
│   │   ├── redux/          # Redux state management (actions, reducers,
store)
│   │   ├── utils/          # Helper functions and utilities
│   │   ├── App.js          # Main app component
│   │   └── index.js        # Entry point for React
│   ├── .env               # Environment variables for the client
│   └── package.json        # Client-side dependencies
│
├── server/              # Express backend code
```

```
│   ├── config/            # Configuration files (e.g., database, JWT,
environment variables)
│   ├── controllers/        # Route handlers (business logic)
│   ├── middleware/         # Middleware (auth, error handling, etc.)
│   ├── models/             # Mongoose models (Product, User, Order,
etc.)
│   ├── routes/             # API routes (RESTful routes for products,
users, etc.)
│   ├── utils/              # Helper functions for backend logic
│   ├── .env               # Environment variables for the server
│   ├── server.js          # Main server file (sets up Express app)
│   └── package.json        # Server-side dependencies
│
├── .gitignore             # Ignore files for Git
├── README.md               # Project description and setup instructions
└── package.json            # Root package.json to manage both client
and server dependencies
```

These remain largely the same, regardless of the tech stack you choose. However, with **Node.js** and **MongoDB**, you can leverage the flexibility of JavaScript and the scalability of NoSQL databases.

Frontend (React.js):

- **HTML, CSS, and JavaScript**: Understanding of the fundamentals of web development.
- **React.js**: Familiarity with React concepts like

- **Components** (functional and class-based)
- **State and Props**
- **Hooks** (use State, Use Effect, etc.)
- **React Router** for navigation
- **Context API** or **Redux** for state management
- **Component lifecycle** and **Event handling**

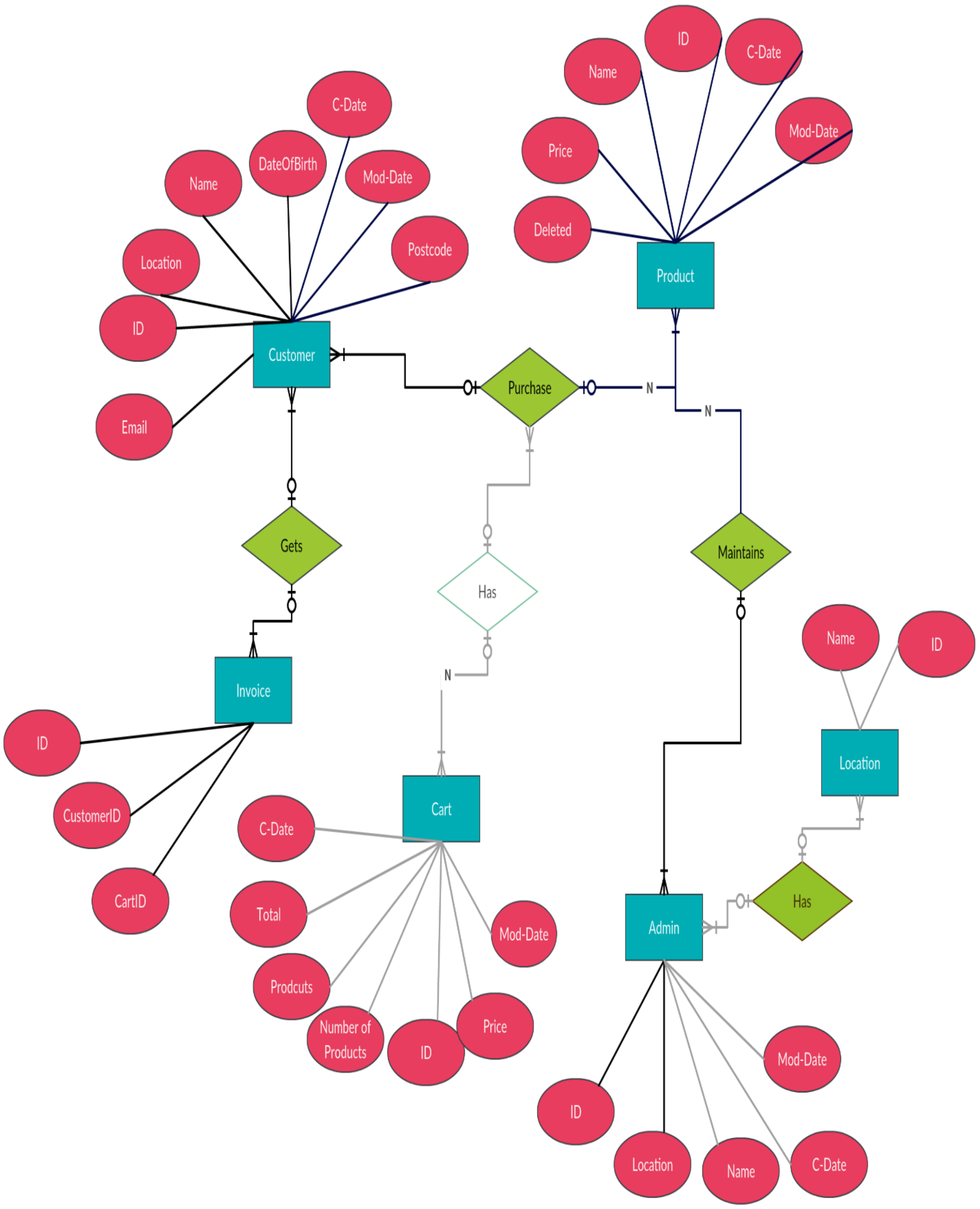
Backend (Node.js & Express.js):

- **Node.js**: Knowledge of JavaScript on the server side.
- **Express.js**: Familiarity with this framework to handle HTTP requests and manage middleware.
- **RESTful API principles**: Understanding of how to design and structure REST APIs (GET, POST, PUT, DELETE).
- **Authentication**: Familiarity with user authentication using **JWT (JSON Web Tokens)**, **sessions**, or **OAuth** for secure logins.
- **Middleware**: Understanding of how to write middleware functions (e.g., for authentication, logging, error handling).

Project Structure:

- User Authentication: JWT-based login and registration (with authMiddleware.js).
- Product Listings: Add, edit, and display products (via productController.js and productRoutes.js).
- Shopping Cart: Cart management using React Context API (or Redux if you need more complex state management).
- Order Management: Users can place orders, which are saved in the Order.js model.

ER Diagram :



Customer
Name
DateOfBirth
C-Date
Mod-Date
Postcode
Location
ID
Email
Product
Name
ID
C-Date
Mod-Date
Price
Deleted
Purchase
N
N
Gets
Has
Maintains
Invoice
ID
CustomerID
CartID
N
Cart
C-Date
Total
Prodcuts
Number of Products
ID
Price
Mod-Date
Location
Name
ID
Has
Admin
ID
Location
Name
C-Date
Mod-Date

Conclusion:

The **Shopez E-Commerce Application**, powered by the **MERN stack** (MongoDB, Express, React, and Node.js), seamlessly blends cutting-edge technologies to create an unparalleled online shopping experience. With its responsive and intuitive user interface, dynamic product management, and secure payment integration, Shopez revolutionizes the way users engage with e-commerce platforms. By leveraging the scalability and flexibility of the MERN stack, Shopez ensures lightning-fast performance, real-time updates, and robust data handling, all while maintaining a user-friendly design. Whether you're a seller or a shopper, Shopez offers a truly seamless and sophisticated digital marketplace, setting new standards in the e-commerce industry.

Team Contribution for Project:

Team Lead Jothi Oversaw the entire project, ensuring that tasks were prioritized and deadlines were met. System Design & Architecture Played a key role in designing the overall architecture of the application, ensuring a scalable and secure backend structure using the MERN stack.

Team Members:

Elamaran M made significant contributions to frontend development. Elamaran's expertise in React.js played a crucial role in creating a dynamic and user-friendly interface. By focusing on a seamless shopping experience, Elamaran ensured smooth navigation, fast loading times, and responsive design, making the application visually appealing and accessible across devices.

Karthikeyan K. made significant contributions to backend development. His work focused on designing and implementing the server-side logic, ensuring smooth communication between the database and frontend. By using **Node.js** and **Express.js**, Karthikeyan optimized API endpoints, integrated user authentication, and implemented efficient data processing for features such as product management, order processing, and customer services.

Archuna Pandiyan A played a vital role in ER diagram error recovery. Their efforts ensured the database structure was optimized and error-free, which streamlined data management and supported a smooth and efficient backend. Archuna’s contributions were essential in creating a reliable foundation for the application's data flow