E-Commerce Application:

This is a simple e-commerce application that allows users to view products, add products to cart, and checkout. The admin can manage products, users, and orders.

The application is built using the following technologies:

1. Frontend: React, Axios, React-Router-Dom V6, Bootstrap, Redux.
2. Backend: Node.js, Express.js, MongoDB, Mongoose, JWT, Bcrypt, Multer, Nodemailer, Nodemon, Dotenv, Morgan, Cors, cookie-parser.
3. Tools: Postman, VS Code, Git, GitHub, Netlify, Render, Vite.

Steps:

Backend:

1. Create an empty directory and open it in VS Code.
2. Open the terminal and run the following command to create a package.json file:

```
npm init
```

3. create an entry point file (index.js).
4. Configure the package.json file. Add the following code:

```
"scripts": {
    "start": "node index.js"
  }
```

5. Create a readme.md file and add the project description.
6. Create an empty repository in GitHub.Com. Copy the repository URL.
7. Initialize the git repository in the project directory:

```
git init
```

8. Add the remote repository URL:

```
git remote add origin <repository-url>
```

9. Create a .gitignore file and add the following code:

```

```

```
node_modules
package-lock.json
DS_Store
.env
```

10. Rename the default branch from master to main:

```
git branch -m main
```

11. Add the changes to the staging area:

```
git add .
```

12. Commit the changes:

```
git commit -m "basic backend application setup"
```

13. Push the changes to the remote repository:

```
git push -u origin main
```

Database Setup:

1. Visit MongoDB.Com and create an account.
2. Create a new project and cluster.
3. Create a new user and password.
4. Open database access if necessary to change the user credentials and privileges.
5. Open network access to allow connections from anywhere by adding the IP address 0.0.0.0/0.
6. Create a new database and a collection.
7. Copy the connecting string from the cluster.
8. Install mongodb compass and connect to the database using the connecting string.

From the backend, connect to the database:

1. Copy the connection string from the cluster.
2. Install mongoose:

```
npm install mongoose
```

3. In the index.js file, add the following code:

```
const mongoose = require('mongoose');

mongoose.connect(connection_string);
```

4. Install dotenv:

```
npm install dotenv
```

5. Create a .env file and add the connection string:

```
MONGODB_URI=connection_string
```

6. Require the dotenv package in the index.js file:

```
require('dotenv').config();
```

7. Change the connection string in index.js inside mongoose.connect() function to

```
process.env.MONGODB_URI
```

8. Add the .env file to the .gitignore file.
9. Create a config.js file under the utils folder and add the following code:

```
require('dotenv').config();

const MONGODB_URI = process.env.MONGODB_URI;

module.exports = {
    MONGODB_URI
};
```

10. Require the config.js file in the index.js file:

```
const { MONGODB_URI } = require('./utils/config');
```

11. Update the variable process.env.MONGODB_URI to MONGODB_URI in the mongoose.connect() function.

Connect to the server using Express.js:

1. Install express:

```
npm install express
```

2. update the index.js file:

```
const express = require('express');

const app = express();

app.get('/', (req, res) => {
    res.send('Hello World');
});

app.listen(3001, () => {
    console.log('Server is running on port 3001');
});
```

This is a simple e-commerce application that allows users to view products, add products to cart, and checkout. The admin can manage products, users, and orders.

User Stories:

1. As a user, I should be able to register and login to the application.
2. As a user, I should be able to view all products.
3. As a user, I should be able to view a single product.
4. As a user, I should be able to add a product to the cart.
5. As a user, I should be able to remove a product from the cart.
6. As a user, I should be able to view the cart.
7. As a user, I should be able to checkout.
8. As an admin, I should be able to add a user.
9. As an admin, I should be able to view all users.
10. As an admin, I should be able to view a single user.
11. As an admin, I should be able to update a user.
12. As an admin, I should be able to delete a user.
13. As an admin, I should be able to add a product.
14. As an admin, I should be able to view all products.
15. As an admin, I should be able to view a single product.
16. As an admin, I should be able to update a product.

17. As an admin, I should be able to delete a product.
18. As an admin, I should be able to view all orders.
19. As an admin, I should be able to view a single order.
20. As an admin, I should be able to update an order.
21. As an admin, I should be able to delete an order.

Tasks:

Backend:

1. Setup the project with Node.js and Express.js. (Done)
2. Setup the environment variables (Dotenv). (Done)
3. Connect to the MongoDB database. (Done)
4. Run the server and test the connection. (Done)
5. Setup the architecture of the project (Models, Routes, Controllers, Middlewares). (Done)
6. Basic housekeeping (Error handling, Logging, Parsing). (Done)
7. Setup the authentication system (Register, Login, Logout). (Not Done)
8. Setup the authorization system (Roles, Permissions). (Not Done)
9. Setup the product system (CRUD). (Not Done)
10. Setup the user system (CRUD). (Not Done)
11. Setup the order system (CRUD). (Not Done)
12. Setup the cart system (CRUD). (Not Done)
13. Setup the checkout system (Payment, Shipping). (Not Done)
14. Setup the email system (Nodemailer). (Not Done)
15. Setup the image system (Multer). (Not Done)
16. Setup the deployment system (Netlify, Render). (Not Done)

Frontend:

1. Setup the project with React. (Not Done)
2. Setup the architecture of the project (Components, Pages, Routes, Redux). (Not Done)
3. Setup the authentication system (Register, Login, Logout). (Not Done) a. Create the Register component. b. Create the Login component. c. Create the Logout component. d. Create the Auth component.
4. Setup the authorization system (Roles, Permissions). (Not Done)
5. Setup the product system (CRUD). (Not Done)
6. Setup the user system (CRUD). (Not Done)
7. Setup the order system (CRUD). (Not Done)
8. Setup the cart system (CRUD). (Not Done)
9. Setup the checkout system (Payment, Shipping). (Not Done)
10. Setup the email system (Nodemailer). (Not Done)
11. Setup the image system (Multer). (Not Done)
12. Setup the deployment system (Netlify, Render). (Not Done)

Models:

1. User Model:

```
{
    name: String,
    email: String,
    password: String,
    role: String,
    cart: [
        {
            product: {
                type: mongoose.Schema.Types.ObjectId,
                ref: 'Product'
            },
            quantity: Number
        }
    ]
}
```

2. Product Model:

```
{
    name: String,
    description: String,
    price: Number,
    image: String
    stock: Number
}
```

3. Order Model:

```
{
    user: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User'
    },
    products: [
        {
            product: {
                type: mongoose.Schema.Types.ObjectId,
                ref: 'Product'
            },
            quantity: Number
        }
    ],
    total: Number,
    status: String
}
```

4. Cart Model:

```
{
    user: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User'
    },
    products: [
        {
            product: {
                type: mongoose.Schema.Types.ObjectId,
                ref: 'Product'
            },
            quantity: Number
        }
    ]
}
```

Routes:

User Routes:

1. POST /api/users/register
2. POST /api/users/login
3. POST /api/users/logout
4. GET /api/users/profile
5. PUT /api/users/profile
6. DELETE /api/users/profile

Product Routes:

1. POST /api/products
2. GET /api/products
3. GET /api/products/:id
4. PUT /api/products/:id
5. DELETE /api/products/:id

Order Routes:

1. POST /api/orders
2. GET /api/orders
3. GET /api/orders/:id
4. PUT /api/orders/:id
5. DELETE /api/orders/:id

Cart Routes:

1. POST /api/carts
2. GET /api/carts
3. GET /api/carts/:id
4. PUT /api/carts/:id
5. DELETE /api/carts/:id