# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi : 590018



A

Project report on

# “SHOPPING-CART”

Submitted in partial fulfillment of the requirement for the award of the degree of

### Bachelor of Engineering in

**INFORMATION SCIENCE AND ENGINEERING**

Submitted by

### Ankita Priyadarshini

### (1SP22IS009)

Under the guidance of

**Asst Prof. Sumitha Udaikumar**

Dept of ISE



**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

### S.E.A. COLLEGE OF ENGINEERING & TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belgaum)

## 2024-2025

**S.E.A. COLLEGE OF ENGINEERING & TECHNOLOGY**

Ekta Nagar, Near by Ayyappanagar Circle, Virgonagar Post, K.R Puram, Bengaluru-560049

(Affiliated to Visvesvaraya Technological University, Belagavi)

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

## Certificate

Certified that the Seminar entitled **“Shopping-Cart”** is a bonafide work carried out by **Ankita Priyadarshini(1SP22IS009)** in partial fulfillment for the award of degree of **Bachelor of Engineering in Information Science And Engineering of Visvesvaraya Technological University**, Belagavi during the year **2024-2025.** It is certified that all corrections/ suggestions indicated for internal assessments have been incorporated in the Report deposited in the departmental library. The project phase 1 (subject code) report has been approved as it satisfies the academic requirements in respect of Seminar prescribed for the Bachelor of Engineering Degree.

## Signature of Guide Signature of HOD Signature of Principal

## Asst Prof Sumitha Udaikumar Dr. Nijaguna G S Dr.B Venkata Narayana

## Dept of ISE Professor & HOD

## DECLARATION

I **Ankita Priyadarshini,** **(1SP22IS009)** student of B.E, Information Science And Engineering,S.E.A College of Engineering & Technology , Bengaluru-49, hereby declare that the Seminar entitled **"Shopping-Cart"** is an authentic record of my own work carried out under the supervision and guidance of Asst. Prof. Sumitha Udaikumar, Department of Information Science And Engineering, S.E.A College of Engineering & Technology, Bengaluru. I have not submitted the matter embodied to any other University or Institution for the award of any other degree.

Date:28/05/2025

Place: Bengaluru

Ankita Priyadarshini(1SP22IS009)

## ACKNOWLEDGEMENT

I express my gratitude to our institution and management for providing us with good infrastructure, laboratory, facilities and inspiring staff, and whose gratitude was of immense help in completion of this report successfully.

I am deeply indebted to **Dr. B. Venkata Narayana** Principal, S.E.A.College of Engineering & Technology, Bangalore, who has been a constant source of enthusiastic inspiration to steer us forward.

I heartily thank **Dr. Nijaguna G S**, Head of the Department, Department of Information Science And Engineering, S.E.A.College of Engineering & Technology, Bangalore, for his valuable support and for rendering us resources for this seminar.

I specially thank Asst.Professor Sumitha Udaikumar, Department of Information Science And Engineering, S.E.A.College of Engineering & Technology who guided us with valuable suggestions in completing this Project work.

I would like to express our sincere thanks and heartfelt gratitude to our beloved Parents, Respected Professors, Classmates, Friends, and juniors for their indispensable help at all times.

Last but not the least our respectful thanks to the Almighty.

Ankita Priyadarshini(1SP22IS009)

**CONTENT**

|  |  |  |
| --- | --- | --- |
| ****SL.No**** | ****Title Name**** | ****Page Number**** |
| **1.** | **Abstract** | **1** |
| **2.** | Introduction | **2** |
| **3.** | Technology Stack and System Architecture | **3** |
| **5.** | Backend Module | **4** |
| **6.** | Frontend Module | **5** |
| **7** | Database Design and Features Implemented | **6** |
| **8** | Source code | **6** |
| **9** | Conclusion and Future Scope | **15** |

## ****ABSTRACT****

This project report presents a full-stack web application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The application allows users to create, read, update, and delete to-do tasks. The system features a responsive frontend interface and a robust backend connected to a MongoDB database. This project demonstrates the effective use of RESTful APIs and React components for task management.

**Introduction**

Task management is a critical function in everyday productivity tools. This project aims to create a minimal yet functional task management system using modern JavaScript technologies. The To-Do List application helps users maintain a list of tasks, mark them as complete, update them, and remove them as needed. This project emphasizes real-world application development practices using the MERN stack.

The backend is developed using Node.js and Express.js, providing a RESTful API to handle all data operations. MongoDB serves as the database to store and manage all task-related information, with Mongoose used to define the schema and simplify database interactions.

This system demonstrates key concepts of full-stack development, including client- server communication, REST API integration, and CRUD operations. It also adheres to modern web design principles with an intuitive user experience and clean interface styling using CSS.

**Technology Stack**

* **MongoDB:** NoSQL database to store to-do items.
* **Express.js:** Web framework for Node.js to build backend services and APIs.
* **React.js:** Frontend JavaScript library to create dynamic user interfaces.
* **Node.js:** Server-side JavaScript runtime environment.

Additional tools:

* **Mongoose:** ODM for MongoDB.
* **Axios/Fetch:** For making HTTP requests from React frontend.
* **CORS & Body-Parser:** Middleware used in the backend.

**System Architecture:**

* **Frontend:** React.js Single Page Application (SPA).
* **Backend:** Node.js with Express.js to define routes and controllers.
* **Database:** MongoDB to store and manage task records.

Architecture Flow:

1. User interacts with the React UI.
2. UI sends API requests to the Express server.
3. Server interacts with MongoDB and returns responses.

**Backend Module**

**Entry Point: index.js**

* Sets up Express server
* Connects to MongoDB at mongodb://localhost:27017/todo-app
* Routes mounted on /api/v1

**Database Connection: db.js**

* Uses Mongoose to connect to MongoDB

**Model:** todoModel.js

* Fields: title (String), isCompleted (String) (suggested: Boolean)
* Uses timestamps

**Controller:** todoController.js

* getAllTodos: Returns all tasks
* addNewTodo: Creates a new task
* getSingleData: Fetches a task by ID
* updateTodo: Updates a task
* deleteTodo: Deletes a task

**Routes:** todoRoutes.js

* RESTful routes linked to above controller methods

**Frontend Module**

**Entry File:** index.js

* Bootstraps React app and renders App.js

Main Component: App.js

* Renders routes to Home and Edit components

**Pages:**

* **Home.js:**
  + Lists all Cart items
  + Allows user to delete or navigate to edit page
* **Edit.js:**
  + Fetches single todo by ID
  + Allows updating and submitting changes

**UI Interactions:**

* Axios or fetch is used to communicate with backend
* State management using React Hooks (useState, useEffect)

**Database Design**

**Collection:** Shopping-Cart(Items)

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| title | String | Title of the to-do |
| isCompleted | String | Status of the task |
| createdAt | Date | Timestamp (auto) |
| updatedAt | Date | Timestamp (auto) |

**Features Implemented**

* Add new items to cart
* View all the cart items
* Edit/update existing to-dos
* Delete items from the cart
* REST API integration between frontend and backend
* Clean and responsive user interface

**Backend Code**

**index.js**

const express = require("express");

const cors = require("cors");

const connectDb = require("./config/db");

const todoRoutes = require("./routes/todoRoutes");

const app = express();

app.use(cors());

app.use(express.json());

connectDb();

app.use("/api/v1", todoRoutes);

const PORT = 4000;

app.listen(PORT, () => console.log(`Server is running on ${PORT}`));

**config/db.js**

const mongoose = require("mongoose");

const connectDb = async () => {

try {

await mongoose.connect("mongodb://localhost:27017/todo-app");

console.log("MongoDB connected");

} catch (err) {

console.error(err);

}

};

module.exports = connectDb;

**models/CartModel.js**

**const mongoose = require("mongoose");**

**const cartSchema = mongoose.Schema(**

**{**

**cartItems: [**

**{**

**product: {**

**id: { type: Number, required: true },**

**title: { type: String, required: true },**

**price: Number,**

**description: { type: String, required: true },**

**rating: { rate: { type: Number }, count: { type: Number } },**

**category: { type: String, required: true },**

**image: { type: String, required: true },**

**},**

**quantity: { type: Number, default: 1 },**

**},**

**],**

**user: {**

**type: mongoose.Types.ObjectId,**

**ref: "user",**

**required: [true, "user id is required"],**

**},**

**totalPrice: Number,**

**},**

**{ timestamp: true }**

**);**

**const cartModel = mongoose.model("cart", cartSchema);**

**module.exports = cartModel;**

**controllers/Controller.js**

const asyncHandler = require("../middleware/asyncHandler");

const userModel = require("../model/userModel");

const cartModel = require("../model/cartModel");

const bcrypt = require("bcrypt");

const sendToken = require("../utils/jwt");

// /api/user/login

const login = asyncHandler(async (req, res) => {

const { email, password } = req.body;

if (!email || !password) {

res.status(401).json({ success: false, message: "Invalid Credential" });

} else {

const isExist = await userModel.findOne({ email });

if (!isExist) {

res.status(401).json({ success: false, message: "Invalid Credential" });

}

const checkPassword = isExist.verifyPassword;

if (checkPassword) {

sendToken(isExist, res);

} else {

res.status(400).json({ success: false, message: "Invalid Credential" });

}

}

});

const register = asyncHandler(async (req, res) => {

const { email, password, name } = req.body;

if (!email || !password || !name) {

res.status(401).json({ success: false, message: "Invalid credential" });

}

const isExist = await userModel.findOne({ email });

if (!isExist) {

const newUser = await userModel.create(req.body);

sendToken(newUser, res);

} else {

res.status(400).json({ success: false, message: "Mail already exists" });

}

});

const logout = asyncHandler(async (req, res) => {

res.cookie("token", "", { httpOnly: true, expires: new Date(0) });

res.status(200).json({ message: "Logout successfully" });

});

const addCart = asyncHandler(async (req, res) => {

const { userId } = req;

const { id } = req.body.product;

const cart = await cartModel.findOne({ user: userId });

if (cart) {

const productExist = cart.cartItems.findIndex(

(item) => item.product.id == id

);

if (productExist !== -1) {

await cartModel.findOneAndUpdate(

{ user: userId, "cartItems.product.id": id },

{ $inc: { "cartItems.$.quantity": 1 } },

{ new: true }

);

const updatedProduct = await cartModel.findOne({ user: userId });

res.status(200).json({

message: "Added Cart",

total: updatedProduct?.cartItems?.length,

});

} else {

const addNewProduct = await cartModel.findOneAndUpdate(

{ user: userId },

{ $push: { cartItems: { product: { ...req.body.product } } } },

{ new: true }

);

res.status(200).json({

message: "Added Cart",

total: addNewProduct?.cartItems?.length,

});

}

} else {

const product = await cartModel.create({

user: userId,

cartItems: [{ product: { ...req.body.product } }],

});

res.status(200).json({

message: "Added Cart",

total: product?.cartItems?.length,

});

}

});

const removeCart = asyncHandler(async (req, res) => {

const { userId } = req;

const { id } = req.body;

const cart = await cartModel.findOne({

user: userId,

"cartItems.product.id": id,

});

if (cart) {

const response = await cartModel.findOneAndUpdate(

{

user: userId,

"cartItems.product.id": id,

},

{ $pull: { cartItems: { "product.id": id } } },

{ new: true }

);

res.status(200).json({

message: "Item removed from cart",

total: response?.cartItems?.length,

});

} else {

res.status(404);

throw new Error("item not found");

}

});

const addCartQty = asyncHandler(async (req, res) => {

const { userId } = req;

const { productId } = req.body;

const cart = await cartModel.findOne({

user: userId,

"cartItems.product.id": productId,

});

if (cart) {

await cartModel.findOneAndUpdate(

{ "cartItems.product.id": productId },

{ $inc: { "cartItems.$.quantity": 1 } },

{ new: true }

);

res.status(200).json({ message: "Added qty" });

} else {

res.status(404);

throw new Error("Item not found");

}

});

const decreaseCartQty = asyncHandler(async (req, res) => {

const { userId } = req;

const { productId } = req.body;

const cart = await cartModel.findOne({

user: userId,

"cartItems.product.id": productId,

});

if (cart) {

await cartModel.findOneAndUpdate(

{ "cartItems.product.id": productId },

{ $inc: { "cartItems.$.quantity": -1 } },

{ new: true }

);

res.status(200).json({ message: "Decreased qty" });

} else {

res.status(404);

throw new Error("Item not found");

}

});

const getUserCart = asyncHandler(async (req, res) => {

const { userId } = req;

const cart = await cartModel.findOne({ user: userId });

if (!cart) {

res.status(404);

throw new Error("Item not found");

} else {

res.status(200).json(cart);

}

});

const getCartNumber = asyncHandler(async (req, res) => {

const { userId } = req;

// console.log(userId);

const resPonse = await cartModel.findOne({ user: userId });

if (resPonse) {

res.status(200).json(resPonse?.cartItems.length);

} else res.status(200).json(0);

});

module.exports = {

login,

register,

logout,

addCart,

removeCart,

decreaseCartQty,

addCartQty,

getUserCart,

getCartNumber,

};

**🎨 Frontend Code**

**src/index.js**

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/svg+xml" href="/vite.svg" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Vite + React</title>

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/main.jsx"></script>

</body>

</html>

import \* as YUP from "yup";

export const loginSchema = YUP.object({

email: YUP.string().email("invalid mail").required("enter email"),

password: YUP.string().required("enter password"),

});

export const registerSchema = YUP.object({

name: YUP.string().min(3, "minimum 3 character").required("enter name"),

email: YUP.string().email("invalid mail").required("enter email"),

password: YUP.string().required("enter password"),

confirmPassword: YUP.string()

.oneOf([YUP.ref("password"), null], "password must be match")

.required("enter password"),

});

**src/App.js**

import React from "react";

import { RouterProvider } from "react-router-dom";

import routes from "./routes/Router";

const App = () => {

return <RouterProvider router={routes} />;

};

export default App;

**src/pages/Cart.jsx**

import React, { useEffect } from "react";

import Layout from "./../layout/Layout";

import CartCard from "../components/CartCard";

import { useAppStore } from "../../store/appStore";

import { useNavigate } from "react-router-dom";

import { PiSmileySadFill } from "react-icons/pi";

const Cart = () => {

const navigate = useNavigate();

const { cart, getUserCart, user } = useAppStore();

const displaycart = cart?.map((itemes, id) => {

return <CartCard item={itemes} key={id} />;

});

useEffect(() => {

if (!user) {

navigate("/login");

}

getUserCart();

}, []);

let initialValue = 0;

let total = Math.ceil(

cart?.reduce(

(acc, item) => acc + item.product.price \* item.quantity,

initialValue

) \* 81

);

return (

<Layout>

<div className="flex min-h-screen w-full flex-col py-5 gap-5 md:flex-row lg:p-[60px]">

<div className="w-full text-slate-600 md:w-[70%] shadow-md border-[1px] bg-white p-2 rounded-lg">

<div className="my-[40px] flex flex-col rounded-[4px] bg-slate-100 md:p-[15px]">

<h2 className="text-[18px] font-[700]">Your Cart</h2>

<h3>Total Itemes : {cart?.length} </h3>

</div>

<div className="flex flex-col gap-3 lg:ml-[120px]">{displaycart}</div>

{!cart?.length && (

<div className="flex flex-col justify-center items-center">

<p className="text-[24px] text-black font-bold">cart is empty</p>{" "}

<br />

<PiSmileySadFill size={35} color="black" />

</div>

)}

</div>

{cart?.length ? (

<div className="my-[40px] flex w-full flex-col md:w-[30%] md:p-[8px]">

<div className="rounded-[4px] shadow-lg">

<div className="flex flex-col rounded-[4px] bg-slate-200 p-[15px]">

<h2 className="text-[18px] font-[700]">You Pay</h2>

<h3>Check Our order Policy</h3>

</div>

<div className="rounded-[4px] bg-slate-600 p-[15px] text-white">

You Can Pay .... on This Order

</div>

<div className="flex flex-col gap-[8px] rounded-[4px] bg-slate-50 p-[15px]">

<h2 className="border-b-[1px] py-[4px] text-[18px] font-[700]">

Total Price {"₹"} {total}

</h2>

<h3 className="border-b-[1px] py-[4px]">

Distributor Price -20

</h3>

<h3 className="border-b-[1px] py-[4px]">Discount 40</h3>

</div>

</div>

</div>

) : null}

</div>

</Layout>

);

};

export default Cart;

**src/pages/Login.jsx**

import React, { useEffect } from "react";

import Layout from "../layout/Layout";

import { useFormik } from "formik";

import { loginSchema } from "../schema";

import InputField from "../components/inputForm/InputField";

import { Navigate, useNavigate } from "react-router-dom";

import { useAppStore } from "../../store/appStore";

const initialValues = {

email: "",

password: "",

};

const Login = () => {

const { loginUser, user } = useAppStore();

const navigate = useNavigate();

const { handleBlur, handleChange, handleSubmit, values, errors, touched } =

useFormik({

initialValues,

validationSchema: loginSchema,

onSubmit: (values, action) => {

loginUser(values, navigate);

},

});

useEffect(() => {

if (user) {

navigate("/");

}

}, [user]);

return (

<Layout>

<section className="w-full h-screen flex items-center justify-center">

<form

onSubmit={handleSubmit}

className="w-[550px] shadow-md p-8 sm:p-10 rounded-2xl bg-white"

>

<h2 className="font-extrabold text-[26px] tracking-widest">Login</h2>

<p onClick={() => navigate("/register")} className="pt-6">

Don't have an account?

<span className="text-blue-500 underline cursor-pointer">

Get Started

</span>

</p>

<InputField

handleChange={handleChange}

name={"email"}

title={"Email "}

type={"email"}

value={values.email}

errors={errors}

touched={touched}

handleBlur={handleBlur}

/>

<InputField

handleChange={handleChange}

name={"password"}

title={"Password "}

type={"password"}

value={values.password}

errors={errors}

touched={touched}

handleBlur={handleBlur}

/>

<h2 className="text-blue-500 py-3 underline text-right cursor-pointer">

forgot-password

</h2>

<button

type="submit"

className="text-[18px] hover:bg-black hover:text-white my-4 w-full py-4 rounded-[8px] border-[1px] border-black font-semibold"

>

Login

</button>

</form>

</section>

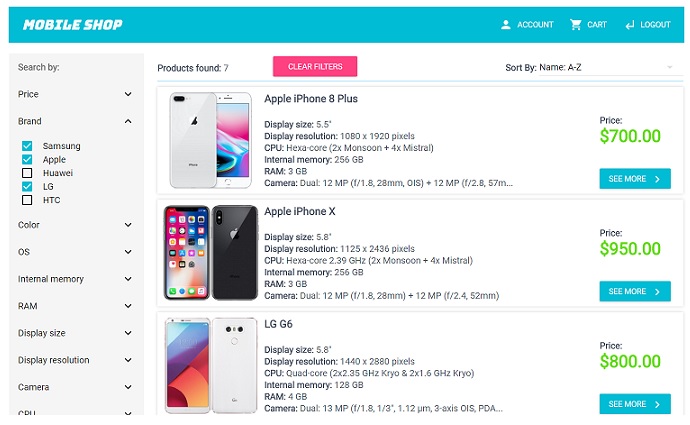
</Layout>

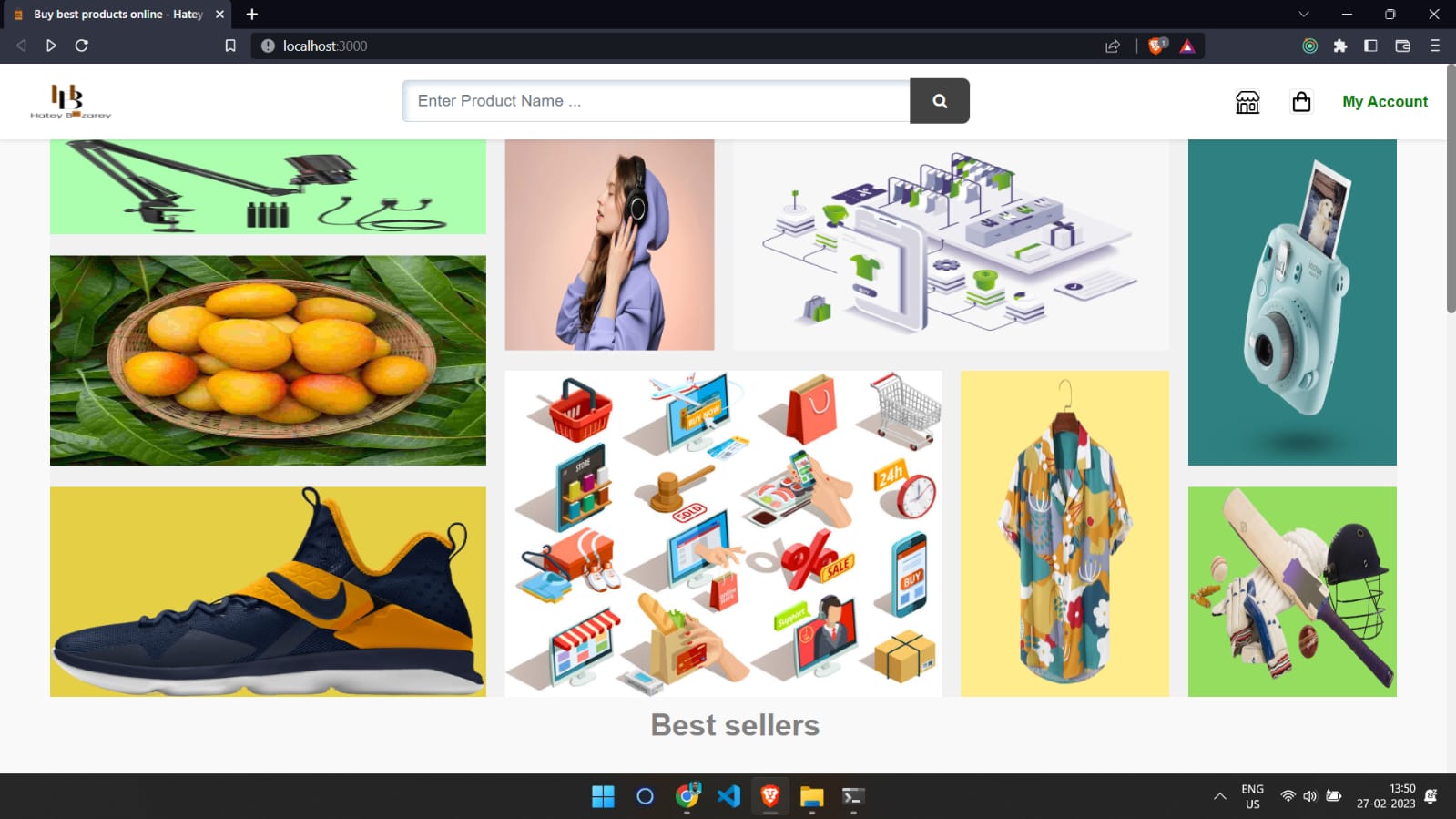
);

};

export default Login;

Snapshots





**Conclusion and Future Scope**

This MERN stack application successfully demonstrates CRUD operations using modern JavaScript technologies. It provides a functional base for developing more advanced productivity tools.

**Future Enhancements:**

* Add user authentication (JWT)
* Improve UI using frameworks like Material-UI or Tailwind CSS
* Add task prioritization and due dates
* Host backend on cloud (e.g., Render, Heroku)
* Host frontend on Vercel/Netlify