SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur, Chengalpattu District - 603203, Tamil Nadu.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

REPORT

on

“1904716 - INTERNSHIP”

at

NTECH PVT.LTD.

142221104161 – TAMIZHARASAN M

SEVENTH SEMESTER

2025-2026

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur - 603203.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **TAMIZHARASAN M** (Reg.No.**142221104161**)-VII Semester, B.E (Computer Science and Engineering) has completed the Internship in **"FULL STACK DEVELOPMENT"** during the period 9th June 2025 to 23rd June 2025 at **"NTECH PVT.LTD."** and has been submitted to Anna University.

SIGNATURE OF MENTOR
Mrs. Dr. A. VIDHYA, M.E.,
Ph.D., ASSISTANT PROFESSOR,
Department of CSE,
SRM Valliammai Engineering College,
Kattankulathur – 603 203.

SIGNATURE OF HOD
Dr. B. VANATHI, M.E., Ph.D.,
PROFESSOR & HEAD,
Department of CSE,
SRM Valliammai Engineering College,
Kattankulathur – 603 203.

Submitted for the university examination held on at
SRM Valliammai Engineering College, Kattankulattur.

INDEX

SI.NO	TITLE	PAGE NO
	ACKNOWLEDGEMENT	iv
	VISION & MISSION	v
	ABSTRACT	vi
1	INTRODUCTION	1
2	COMPANY PROFILE	2
3	INTERNSHIP DESCRIPTION	3
4	TECHNOLOGIES USED	4
5	INTERNSHIP PROJECT	7
6	SOURCE CODE	10
7	OUTPUT	21
8	CONCLUSION	24
9	CERTIFICATE	25

ACKNOWLEDGEMENT

I wish to express my heartfelt gratitude to the eminent personalities who served as the backbone of my internship's success, beyond my own efforts. I am sincerely thankful to "**NTECH PVT.LTD.**" for providing me with the opportunity to undertake this internship.

I extend my sincere thanks to **Mr.MOHAMAD ISMAN** , Tutor of "**NTECH PVT.LTD.**" for his kind patronage and for all the facilities provided that contributed to the successful completion of this internship.

We sincerely express our gratitude in depth to our esteemed Founder Chairman & Chancellor **Dr. T. R. Paarivendhar, Thiru. Ravi Pachamoothoo** Chairman SRM Group & Pro-Chancellor (Admin). **Mrs. R. Harini,** Correspondent, SRM VEC and authorities of **SRM VALLIAMMAI ENGINEERING COLLEGE** for the patronage on our welfare rooted in the academic year.

I am highly indebted to **Director Dr. B. Chidhambararajan, M.E., Ph.D.,** for his constant encouragement, which has been motivated to strive towards excellence. We thank our sincere **Principal Dr. M. Murugan, M.E., Ph.D., and Vice-Principal Dr. Visalakshi. S, M.E., Ph.D.,** for the support and facilities they provided to accomplish this internship.

I would like to thank my Head of the Department, **Dr. B. Vanathi, M.E., Ph.D.,** for her constructive criticism throughout my internship.

I am grateful to **Mrs. Dr. A. VIDYA, M.E., .Ph.D.,** Assistant Professor, my mentor and Internship Coordinator, Department of Computer Science, for her support and advice to get complete internship in above said organization.

I am extremely grateful to the staff members and friends in my department who helped me in the successful completion of this internship.

VISION AND MISSION

VISION

To devise captivating, fascinating, and unique practices of teaching that discovers the trained talent and inherent competences of young minds to evolve as humane professional Computer Science Engineers.

MISSION

- To provide students with challenging ventures, contributing to the betterment of their selfdom to compete with international talents.

- To act as a motivational hub to exhibit practical knowledge with the latest technological updates and research publications.

- To render ample knowledge to exhibit their ubiquitous talents for the social prosperity and promote industry-institute harmony to upgrade the standards for the international reputation.

ABSTRACT

Full-stack web development involves managing both the front-end and back-end, along with the database components of a web application. This approach allows developers to work on the entire architecture of a project — from designing user interfaces to implementing server-side logic and managing databases.

During this internship, the focus was on the **MERN stack**, which includes **MongoDB, Express.js, React.js, and Node.js**. These modern technologies collectively support the development of efficient, scalable, and dynamic web applications.

The **front-end** was developed using **React.js**, which is known for building responsive and interactive user interfaces through component-based architecture. On the **back-end**, **Node.js** and **Express.js** were used to handle the server-side logic, create RESTful APIs, and manage HTTP requests efficiently. **MongoDB**, a NoSQL database, was used to store and retrieve application data, offering flexibility and scalability.

This full-stack approach ensures seamless integration between client-side and server-side components. By working with the MERN stack, I gained hands-on experience in developing real-world web applications, handling routing, implementing state management, and ensuring performance and security. The internship helped me understand how to address common web development challenges such as cross-platform compatibility, data handling, and optimization, resulting in robust and user-friendly web solutions.

1. INTRODUCTION

1.1 INTERNSHIP OVERVIEW

This report highlights my **15-day internship experience in MERN Full Stack Web Development** at **NTECH Computer Education, Tambaram, Chennai**, a reputed institution dedicated to advanced IT training. The primary objective of this internship was to bridge the gap between theoretical knowledge and its real-world application in web development using modern technologies.

Throughout this focused and enriching training, I was exposed to the complete web development lifecycle. I gained **hands-on experience in building and deploying full-stack web applications** using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**. The internship offered valuable insight into both **front-end development**, involving user interface creation with React.js, and **back-end operations**, including server logic and database handling with Node.js, Express.js, and MongoDB.

This immersive learning experience significantly enhanced my understanding of full-stack development and equipped me with the technical skills needed to contribute effectively to real-time web development projects.

1.2 OBJECTIVES

1. **Skill Acquisition**: Develop hands-on proficiency in full-stack development using **MongoDB, Express.js, React.js, and Node.js**, focusing on both front-end and back-end technologies.
2. **Project Implementation**: Participate in building responsive, dynamic, and secure web applications by applying **MERN stack principles** and modern development workflows.
3. **Collaboration**: Engage in collaborative tasks and team-based activities to strengthen communication, problem-solving, and code integration skills in a development environment.
4. **Problem-Solving**: Utilize logical and analytical thinking to solve real-time issues in both client-side and server-side operations throughout the web application lifecycle.
5. **Industry Insight**: Gain exposure to **current industry practices**, full-stack architecture, version control, and deployment strategies used in professional MERN-based web development projects.

3. INTERNSHIP DESCRIPTION

DOMAIN: MERN DEVELOPMENT

3.1 ROLE AND RESPONSIBILITIES

My responsibilities were diverse and aimed at providing comprehensive exposure to the field of MERN stack development.

The key responsibilities included:

1. **Front-End Development**: Developed dynamic and responsive user interfaces using **React.js**, incorporating modern JavaScript features to ensure a seamless user experience.
2. **Back-End Development**: Implemented server-side functionality using **Node.js** and **Express.js**, and managed data operations using **MongoDB**, supporting robust application logic and database interactions.
3. **Project Collaboration**: Engaged in team-based tasks, collaborating with peers and instructors to design, develop, and deploy MERN stack applications, strengthening communication and coordination skills.
4. **Code Integration and Testing**: Integrated front-end and back-end components, performed **unit and functional testing**, and ensured that applications met performance, security, and usability standards.
5. **Troubleshooting and Debugging**: Identified bugs and resolved development issues during the build and deployment stages, ensuring the delivery of smooth and error-free web applications.
6. **Learning and Adaptation**: Continuously explored new tools and techniques within the MERN ecosystem, applying best practices and keeping up with current trends in full-stack development.

4. TECHNOLOGIES USED

4.1 HTML:

Hypertext Markup Language (HTML) is the standard markup language used to create web pages. It forms the foundation of web development by providing the structure for web content. HTML elements are the building blocks of web pages, allowing for the embedding of images, text, and interactive forms. HTML tags, defined using angle brackets, are used to create headings, paragraphs, lists, links, and other elements, facilitating the development of well-structured documents. For this project, HTML was used to create the skeleton of the web application's user interface, ensuring a clear and logical layout for users to interact with the web app.

4.2 CSS:

Cascading Style Sheets (CSS) is a style sheet language employed to describe the presentation of HTML documents. CSS allows for the separation of content and presentation, enabling developers to control the layout, colors, fonts, and overall aesthetics of a web page. By using CSS, the web application was given a visually appealing and consistent design, ensuring that it was both user-friendly and responsive across various devices. CSS was essential for defining the look and feel of the system, ensuring a seamless user experience.

4.3 JAVASCRIPT:

JavaScript is a versatile scripting language used to create dynamic and interactive content on web pages. It allows developers to implement complex features such as interactive forms, animations, and real-time updates. In this project, JavaScript was used to enhance the user experience by providing client-side validation, dynamic content updates, and other interactive features.

4.4 REACT.JS:

React.js is a powerful front-end JavaScript library used to build dynamic and responsive user interfaces. It follows a component-based architecture that promotes reusability and clean code structure. React enables real-time UI updates with efficient rendering. Features like hooks and props help manage state and data flow. It is ideal for building scalable and interactive web applications.

4.5 NODE.JS:

Node.js is a server-side runtime environment that allows JavaScript to be executed outside the browser. It supports asynchronous, event-driven programming, which enhances performance for web applications. Node.js was used to handle backend logic and interact with databases. It enables the creation of fast and scalable server applications. Its compatibility with JavaScript across the stack simplifies development.

4.6 EXPRESS.JS:

Express.js is a lightweight Node.js framework used for building web applications and APIs. It simplifies routing, middleware integration, and server configuration. Express was used to handle HTTP requests and manage backend logic efficiently. It connects the front-end to the database securely. Its minimalist approach provides flexibility and rapid development.

5.INTERNSHIP PROJECT

PROJECT TITLE:MERN Stack Based E-Commerce Web Application (ShopNow)

The project titled **"ShopNow"** is a feature-rich and user-friendly **E-Commerce Web Application** developed using the **MERN stack (MongoDB, Express.js, React.js, and Node.js)**. It is designed to provide customers with a smooth online shopping experience, allowing them to browse products, add items to cart, place orders, and manage their profiles. Administrators can manage products, view orders, and monitor user activity through a dedicated dashboard. This full-stack web application offers dynamic content rendering, secure authentication, and a responsive design to ensure accessibility across all devices.

5.1 OBJECTIVE:

The objective of this project is to build a responsive and user-friendly e-commerce website using the MERN stack. It allows users to browse products, manage their cart, place orders, and securely log in. Admins can manage products and view orders through a separate dashboard. The project also aims to enhance full-stack development skills and implement secure, real-time web features.

5.2 KEY FEATURES:

5.2.1 User Authentication: Secure login and registration system using JWT (JSON Web Tokens), enabling users to create accounts, log in, and manage sessions securely.

5.2.2 Product Listing and Search:
Users can browse a wide variety of products, with functionality to search, filter by category, and sort results for a more efficient shopping experience.

5.2.3 Shopping Cart:
An interactive cart system allows users to add, update, or remove products before placing an order, ensuring a flexible and user-friendly checkout experience.

5.2.4 Order Management:
Users can place orders and view their previous purchase history. Each order is securely stored in the database for future reference and tracking.

5.3 DESIGNED WEBPAGES:

- `index.html` – Main landing page showcasing featured products or offers.
- `style.css` – CSS file for styling all frontend components and layout.
- `contact.php` – Contact form page for user inquiries or support.
- `script.js` – JavaScript file for handling interactivity and dynamic behavior on the client side.

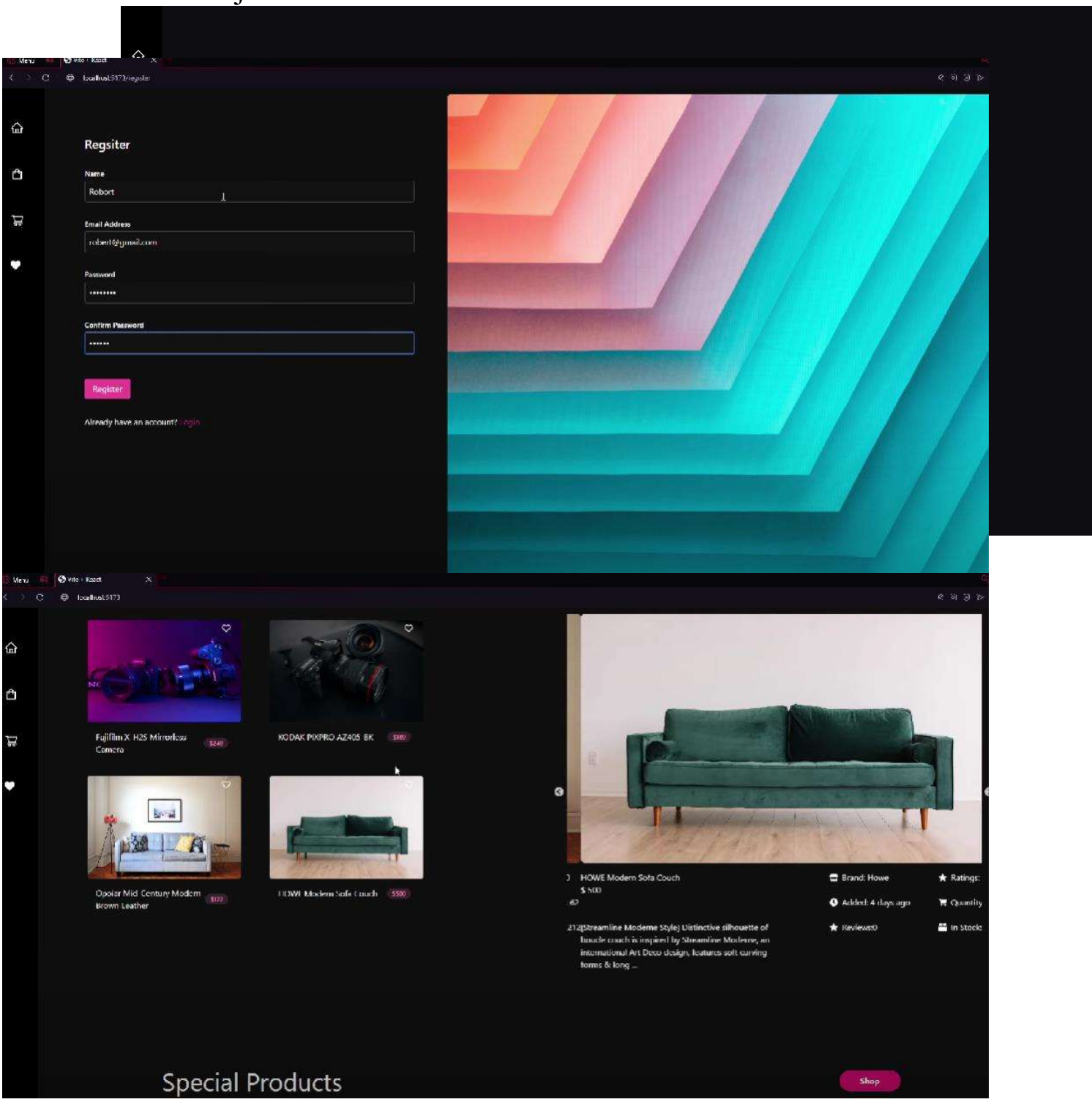
5.4 SOFTWARES REQUIRED:

- **MongoDB**
- **Node.js**
- **Express.js**
- **React.js**

5.5 IMPLEMENTATION

5.5.1 EXECUTING THE PROJECT:

MERN Project Photos



6.SOURCE CODE

6.1 INDEX.HTML:

```
// packages
import path from "path";
import express from "express";
import dotenv from "dotenv";
import cookieParser from "cookie-parser";
import cors from "cors"; // ☑ IMPORT CORS

// Utiles
import connectDB from "./config/db.js";
import userRoutes from "./routes/userRoutes.js";
import categoryRoutes from "./routes/categoryRoutes.js";
import productRoutes from "./routes/productRoutes.js";
import uploadRoutes from "./routes/uploadRoutes.js";
import orderRoutes from "./routes/orderRoutes.js";

dotenv.config();
const port = process.env.PORT || 5000;

connectDB();

const app = express();


app.use(cors({
  origin: "http://localhost:5173",
  credentials: true,
}));

app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cookieParser());

// API routes
app.use("/api/users", userRoutes);
app.use("/api/category", categoryRoutes);
app.use("/api/products", productRoutes);
app.use("/api/upload", uploadRoutes);
app.use("/api/orders", orderRoutes);

// Paypal route
app.get("/api/config/paypal", (req, res) => {
  res.send({ clientId: process.env.PAYPAL_CLIENT_ID });
```

```
    process.exit();
  } catch (err) {
    console.error('❌ Error creating admin users:', err);
    process.exit(1);
  }
};

seedAdmins();
```

UserControl.js:

```
import User from "../models/userModel.js";
import asyncHandler from "../middlewares/asyncHandler.js";
import bcrypt from "bcryptjs";
import createToken from "../utils/createToken.js";

const createUser = asyncHandler(async (req, res) => {
  const { username, email, password } = req.body;

  if (!username || !email || !password) {
    throw new Error("Please fill all the inputs.");
  }

  const userExists = await User.findOne({ email });
if (userExists) {
  res.status(400).send("User already exists");
  return; // <-- ADD THIS
}



  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(password, salt);
  const newUser = new User({ username, email, password: hashedPassword });

  try {
    await newUser.save();
    createToken(res, newUser._id);

    res.status(201).json({
      _id: newUser._id,
      username: newUser.username,
      email: newUser.email,
      isAdmin: newUser.isAdmin,
    });
  } catch (error) {
```

```
    res.status(400);
    throw new Error("Invalid user data");
  }
});

const loginUser = asyncHandler(async (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    res.status(400);
    throw new Error("Please provide both email and password");
  }

  const existingUser = await User.findOne({ email });

  if (!existingUser) {
    res.status(401);
    throw new Error("Invalid email or password");
  }

  const isPasswordValid = await bcrypt.compare(password, existingUser.password);

  if (!isPasswordValid) {
    res.status(401);
    throw new Error("Invalid email or password");
  }

  // If valid login
  createToken(res, existingUser._id);

  res.status(200).json({
    _id: existingUser._id,
    username: existingUser.username,
    email: existingUser.email,
    isAdmin: existingUser.isAdmin,
  });
});


const logoutCurrentUser = asyncHandler(async (req, res) => {
  res.cookie("jwt", "", {
    httpOnly: true,
    expires: new Date(0),
  });
```

```
  res.status(200).json({ message: "Logged out successfully" });
});

const getAllUsers = asyncHandler(async (req, res) => {
  const users = await User.find({});
  res.json(users);
});

const getCurrentUserProfile = asyncHandler(async (req, res) => {
  const user = await User.findById(req.user._id);

  if (user) {
    res.json({
      _id: user._id,
      username: user.username,
      email: user.email,
    });
  } else {
    res.status(404);
    throw new Error("User not found.");
  }
});

const updateCurrentUserProfile = asyncHandler(async (req, res) => {
  const user = await User.findById(req.user._id);

  if (user) {
    user.username = req.body.username || user.username;
    user.email = req.body.email || user.email;

    if (req.body.password) {
      const salt = await bcrypt.genSalt(10);
      const hashedPassword = await bcrypt.hash(req.body.password, salt);
      user.password = hashedPassword;
    }

    const updatedUser = await user.save();

    res.json({
      _id: updatedUser._id,
      username: updatedUser.username,
      email: updatedUser.email,
      isAdmin: updatedUser.isAdmin,
    });
  } else {
```

```
    res.status(404);
    throw new Error("User not found");
  }
});

const deleteUserById = asyncHandler(async (req, res) => {
  const user = await User.findById(req.params.id);

  if (user) {
    if (user.isAdmin) {
      res.status(400);
      throw new Error("Cannot delete admin user");
    }

    await User.deleteOne({ _id: user._id });
    res.json({ message: "User removed" });
  } else {
    res.status(404);
    throw new Error("User not found.");
  }
});

const getUserById = asyncHandler(async (req, res) => {
  const user = await User.findById(req.params.id).select("-password");

  if (user) {
    res.json(user);
  } else {
    res.status(404);
    throw new Error("User not found");
  }
});

const updateUserById = asyncHandler(async (req, res) => {
  const user = await User.findById(req.params.id);

  if (user) {
    user.username = req.body.username || user.username;
    user.email = req.body.email || user.email;
    user.isAdmin = Boolean(req.body.isAdmin);

    const updatedUser = await user.save();

    res.json({
      _id: updatedUser._id,
```

```
      username: updatedUser.username,
      email: updatedUser.email,
      isAdmin: updatedUser.isAdmin,
    });
  } else {
    res.status(404);
    throw new Error("User not found");
  }
});

export {
  createUser,
  loginUser,
  logoutCurrentUser,
  getAllUsers,
  getCurrentUserProfile,
  updateCurrentUserProfile,
  deleteUserById,
  getUserById,
  updateUserById,
};
```

UserModel.js:

```
import mongoose from "mongoose";

const userSchema = mongoose.Schema(
  {
    username: {
      type: String,
      required: true,
    },

    email: {
      type: String,
      required: true,
      unique: true,
    },

    password: {
      type: String,
      required: true,
    },

    isAdmin: {
```

```
      type: Boolean,
      required: true,
      default: false,
    },
  },
  { timestamps: true }
);

const User = mongoose.model("User", userSchema);

export default User;
```

ProductModel.js:

```
import mongoose from "mongoose";
const { ObjectId } = mongoose.Schema;

const reviewSchema = mongoose.Schema(
  {
    name: { type: String, required: true },
    rating: { type: Number, required: true },
    comment: { type: String, required: true },
    user: {
      type: mongoose.Schema.Types.ObjectId,
      required: true,
      ref: "User",
    },
  },
  { timestamps: true }
);

const productSchema = mongoose.Schema(
  {
    name: { type: String, required: true },
    image: { type: String, required: true },
    brand: { type: String, required: true },
    quantity: { type: Number, required: true },
    category: { type: ObjectId, ref: "Category", required: true },
    description: { type: String, required: true },
    reviews: [reviewSchema],
    rating: { type: Number, required: true, default: 0 },
    numReviews: { type: Number, required: true, default: 0 },
    price: { type: Number, required: true, default: 0 },
    countInStock: { type: Number, required: true, default: 0 },
  },
```

```
  { timestamps: true }
);

const Product = mongoose.model("Product", productSchema);
export default Product;
```