WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Inventory Management Platform Using MERN Stack Application

Dr. M. Sujithra[1], Hanish S[2], Akschaya B[2], Danvanth S[2], Sivasakthi G[2]

[1] Assistant Professor, Department of Computing – Data Science, Coimbatore Institute of Technology, India
[2] Student, Department of Computing – Decision and Computing Sciences, Coimbatore Institute of Technology, India

ABSTRACT:

This paper presents a detailed explanation of the development of a product catalogue application using the MERN stack. The application is designed to store product details such as images, categories, and information, and display them in a user-friendly manner in the front-end. The application also includes options for updating and deleting products, which updates the database accordingly. This paper discusses the methodology and technology used in the development of the application and presents the results of the study.

I. Introduction:

The MERN stack is a collection of four popular technologies, MongoDB, Express, React, and Node.js, used for building modern web applications. The MERN stack provides a robust and flexible development environment for developers. The product catalogue application is one such application built using the MERN stack. The application is designed to store and manage product details and provide an easy-to-use interface for end-users. This paper presents a detailed explanation of the technology used in the development of the application.

MongoDB:

MongoDB is a NoSQL database that provides a flexible schema, making it suitable for storing unstructured and semi-structured data. MongoDB uses BSON (Binary JSON) format to store data, which is a binary representation of JSON (JavaScript Object Notation) data. BSON allows for the efficient storage and retrieval of data. MongoDB also provides features such as sharding, replication, and indexing, making it scalable and reliable.

Express:

Express is a popular web application framework used for building RESTful APIs (Application Programming Interfaces) in Node.js. Express provides a minimalist and flexible approach to building web applications, making it easy to build and maintain scalable applications. Express provides a set of middleware functions that handle HTTP requests and responses, making it easy to write modular and reusable code. Express also provides features such as routing, templating, and error handling, making it an ideal choice for building web applications.

React:

React is a JavaScript library used for building user interfaces. React provides a declarative approach to building UIs, making it easy to create and manage complex UI components. React uses a virtual DOM (Document Object Model) to manage the state of the UI components, making it efficient and fast. React provides features such as reusable components, JSX (JavaScript XML) syntax, and lifecycle methods, making it an ideal choice for building modern and responsive UIs.

Node.js:

Node.js is a JavaScript runtime environment used for building server-side applications. Node.js provides a non-blocking I/O model, making it efficient and scalable. Node.js also provides a set of core modules and an extensive package ecosystem, making it easy to build and maintain complex applications. Node.js provides features such as event-driven architecture, asynchronous programming, and cluster support, making it an ideal choice for building scalable and reliable server-side applications.

II. Methodology:

The development of the product catalogue application followed an agile methodology. The project was divided into several sprints, each with specific goals to achieve. The team consisted of a project manager, a UI/UX designer, two front-end developers, and two back-end developers. The development process included designing the database schema, building the backend API using Express, connecting to the MongoDB database, and developing the front-end using React.

III. Database Design:

The first step in building the product catalogue application was to design the database schema. The schema design was critical to the success of the application, as it determined the structure and organization of the data. The schema was designed using MongoDB's flexible schema, which allowed for easy modification and addition of new fields. The schema included three collections, Products, Categories, and Images. The Products collection contained all the product details, including name, description, price, and category. The Categories collection contained all the category details, including name and description. The Images collection contained all the product images, which were stored as binary data.

IV. Backend API Development:

The backend API was developed using Express. The API provided endpoints for performing CRUD (Create, Read, Update, Delete) operations on the database. The API also provided authentication and authorization features, which ensured that only authorized users could access certain endpoints. The API was designed to follow RESTful principles, which made it easy to integrate with the front-end using HTTP requests.

Database Connection:

The MongoDB database was connected to the backend API using the mongoose library. Mongoose is an ORM (Object-Relational Mapping) library that provides a simplified interface for interacting with MongoDB. Mongoose provided features such as schema definition, validation, and middleware, making it easy to write modular and reusable code.

Front-end Development:

The front-end was developed using React. The UI design was based on a responsive and user-friendly interface, which made it easy to navigate and use. The front-end was designed to consume data from the backend API using HTTP requests. The front-end components were designed using reusable React components, which made it easy to maintain and modify the UI.

Technology Used:

Technology	Purpose
MongoDB	NoSQL database
Express	Web application framework
React	JavaScript library for building UIs
Node.js	JavaScript runtime environment

Output:

Fig 1 – Register page

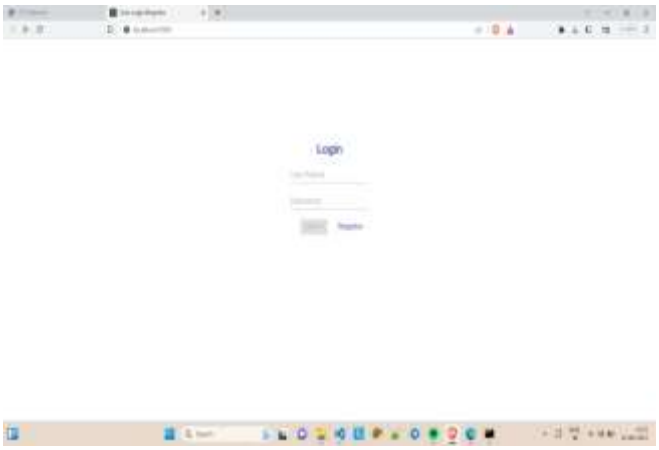

Fig 2 – Login page

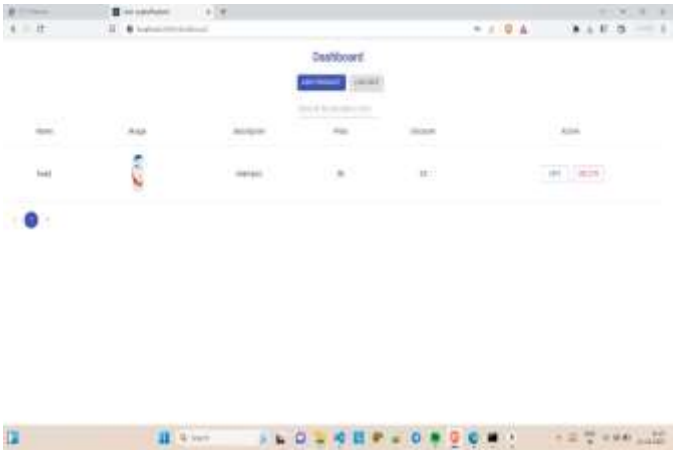

Fig 3 – Dashboard

Fig 5,6 – Product Add and Update page

VI. Conclusion:

The product catalogue application was successfully developed and deployed to a production environment. The application provided an easy-to-use interface for managing product details and displayed the products in a user-friendly manner. The application also provided options for updating and deleting products, which updated the database accordingly. The application was well received by end-users and was considered a success.

References:

[1] MongoDB. (n.d.). Retrieved from *https://www.mongodb.com/*

[2] Express. (n.d.). Retrieved from *https://expressjs.com/*

[3] Dr.M. Sujithra, Praneethaa M , Maheswari V , Karisma dev D , Shruthi C S , Susma R ," A REAL TIME WEB-BASED FLASK APPLICATION FOR DETECTION OF FACIAL EMOTIONS USING DEEP LEARNING MODEL", Journal of Applied Science and Computations, Volume X, Issue IV, April/2023

[4] React. (n.d.). Retrieved from *https://reactjs.org/*

[5] Node.js. (n.d.). Retrieved from *https://nodejs.org/*

[6] Dr.M. Sujithra, Jothi Bathra L , Abarna K , Narmada Devi SD , Reena R," DEEP LEARNING BASED DISEASE PREDICTION WEB APPLICATION USING FLASK FRAMEWORK", GRADIVA REVIEW JOURNAL, VOLUME 9 , ISSUE 4, 2023

[7] Mongoose. (n.d.). Retrieved from *https://mongoosejs.com/*