CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND
COMMUNICATION TECHNOLOGY


PROJECT - SPECIALIZED REPORT
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

# BUILD SPORT EQUIREMENT ECOMMERCE WEBSITE WITH MEVN STACK

**Student:** Nguyen Thanh Phat
**Student ID:** B2005853
**Cohort: K46**

**Cantho, 09/2023**

CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND
COMMUNICATION TECHNOLOGY


PROJECT - SPECIALIZED REPORT
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

# BUILD SPORT EQUIREMENT ECOMMERCE WEBSITE WITH MEVN STACK

**Advisor:**
Dr. Lam Van Khang

**Student:**
Nguyen Thanh Phat
**Student ID:** B2005853
**Cohort: K46**

**Cantho, 09/2023**

# Contents

# List of Figures

# List of Tables

**Abstract**

This project is aimed at creating an online marketplace for sports equipment, offering users a seamless and intuitive shopping experience. The website will be developed using the MEVN stack (MongoDB, Express.js, Vue.js, Node.js), supplemented with Bootstrap for responsive design, and TypeScript for static typing, enhancing code quality and understandability.

The development process will adhere to the principles of the Minimum Viable Product (MVP) approach, resulting in a compact, single-tiered website. This approach ensures the system is scalable, maintainable, and provides a rich, user-friendly interface.

The project's scope is concentrated on fulfilling the essential features of an e-commerce website, including product listing, shopping cart functionality, order processing, user authentication, wishlist management, and product reviews.

The anticipated outcomes of this project include a deeper comprehension of full-stack web development using the MEVN stack, understanding the MVP approach, and gaining insights into the operational aspects of an e-commerce website. This project serves as a practical application of these technologies and concepts, demonstrating their effectiveness in a real-world scenario.

**Tóm tắt nội dung**

Dự án này nhằm mục đích tạo ra một thị trường trực tuyến cho thiết bị thể thao, mang đến cho người dùng trải nghiệm mua sắm liền mạch và trực quan. Trang web sẽ được phát triển bằng MEVN stack (MongoDB, Express.js, Vue.js, Node.js), được bổ sung Bootstrap cho thiết kế đáp ứng và TypeScript cho kiểu tĩnh, nâng cao chất lượng và khả năng hiểu của mã.

Quy trình phát triển sẽ tuân theo các nguyên tắc của phương pháp Sản phẩm tối thiểu khả thi (MVP), dẫn đến một trang web nhỏ gọn, một tầng. Phương pháp này đảm bảo hệ thống có khả năng mở rộng, dễ bảo trì và cung cấp giao diện người dùng phong phú, thân thiện.

Phạm vi của dự án tập trung vào việc đáp ứng các tính năng thiết yếu của một trang web thương mại điện tử, bao gồm liệt kê sản phẩm, chức năng giỏ hàng, xử lý đơn hàng, xác thực người dùng, quản lý danh sách mong muốn và đánh giá sản phẩm.

Các kết quả dự kiến của dự án bao gồm sự hiểu biết sâu sắc hơn về phát triển web full-stack bằng MEVN stack, hiểu được phương pháp MVP và thu được thông tin chi tiết về các khía cạnh hoạt động của trang web thương mại điện tử. Dự án này đóng vai trò là ứng dụng thực tế của các công nghệ và khái niệm này, thể hiện hiệu quả của chúng trong một tình huống thực tế.

# Chapter 1

# Introduction

## 1.1 Background

The rise of information technology has significantly impacted business operations, with software applications becoming integral to sales operations. However, choosing the right software and platform can be challenging, especially for small and medium-sized enterprises (SMEs).

Traditionally, many businesses have used monolithic architecture, a unified and self-contained model. While this model is convenient in the early stages of a project, it can become complex and hard to manage as the application grows.

To address these challenges, businesses are now turning to the MEVN stack (MongoDB, Express.js, Vue.js, Node.js). This stack allows for a scalable and maintainable system, providing a rich, user-friendly interface.

For instance, a sports equipment shop website can be developed using the MEVN stack. By breaking down the application into smaller parts, it's easier to scale services based on demand and deploy updates more frequently without disrupting the entire system.

In the case of our sports equipment shop website, we will be using the MEVN stack to manage products, orders, and payments. Inventory management wil not be included in the initial version of the system. Instead, it's planned to be incorporated in a future update. This approach allows for the system to be flexible and scalable, accommodating new features as needed while ensuring system reliability.

## 1.2 Problem statement

SMEs face difficulties in selecting suitable software and platforms for their evolving digital operations. Traditional monolithic architecture, while initially convenient, becomes cumbersome and challenging to manage as applications grow, hindering scalability and user experience.

For online sports equipment shops, managing products, orders, and payments is vital. However, due to constraints, the initial system version will be include inventory

management, but it is planned for a future update. This strategy ensures flexibility, scalability, and system reliability, addressing key challenges

The core problem lies in developing a user-friendly, scalable, and reliable online sports equipment shop using a technology stack that overcomes the limitations of traditional monolithic architecture and effectively manages the fundamental requirements of an e-commerce website.

## 1.3 Research Objectives

The primary aim of this project is to design and implement an e-commerce application for a sports equipment shop, along with a front-end application using Vue.js. The specific objectives are as follows:

- **Literature Review:** Conduct a comprehensive review of existing literature on the MEVN stack, its benefits, challenges, and best practices. This will provide a theoretical foundation for the project.
- **Design:** Design a MEVN-based e-commerce application model. The design should consider factors such as scalability, fault tolerance, and ease of adding new features.
- **Development:** Develop a minimum viable product (MVP) of an e-commerce application for a sports equipment shop website using the proposed model. The MVP will include key features such as product management, order processing, and payment processing. Inventory management will be considered for future updates.
- **Documentation:** DDocument the entire process, including the design decisions made, challenges encountered, solutions implemented, and lessons learned. This will serve as a valuable resource for future projects of similar nature.

## 1.4 Research Scope

The focus of this research is on the design and implementation of a MEVN-based e-commerce application for a sport equipment shop website. The specific areas covered in this research include:

- **MEVN Stack:** The research will delve into the principles and practices of the MEVN stack. It will explore how to design and implement an e-commerce application using this stack.
- **Front-end Compatibility:** The research will explore the compatibility of the developed back-end API with various front-end frameworks. While the API is designed to be standalone and can work with any front-end framework, the effectiveness of this design will be evaluated
- **Front-end Development with Vue.js:**The research will delve into the integration of the developed back-end API with Vue.js. While the API is designed

to be standalone and compatible with any front-end framework, Vue.js has been chosen for its approachability and popularity.

- **Key Features:** The research will focus on implementing key features for an e-commerce application, such as product management, inventory management, order processing, and payment processing.
- **Evaluation:** The research will include an evaluation of the implemented system in terms of functionality, performance, scalability, and reliability.

Please note that while the research aims to cover these areas, it is limited by the project's short time. Therefore, the e-shop application developed as part of this project will be a minimum viable product (MVP) with basic features.

## 1.5 Solution approach

The approach to solving the problem involves several steps, each designed to ensure the successful implementation of an e-commerce application for a sports equipment shop, along with a front-end application using Vue.js. The steps are as follows:

- **Literature Review:** The first step involves conducting a comprehensive review of existing literature on the MEVN stack. This will provide a theoretical foundation for the project and inform the design and implementation stages.
- **Design:** The next step is to design the MEVN-based e-commerce application model. The design will take into account factors such as scalability, fault tolerance, and ease of adding new features.
- **Development:** Once the design is complete, the development of the minimum viable product (MVP) begins. This involves coding the back-end services and front-end application, and setting up the necessary databases and interfaces. Inventory management will be considered for future updates.
- **Testing: :** After the MVP is developed, it will be thoroughly tested to ensure it functions as expected. This includes unit testing, integration testing, and system testing.
- **Evaluation:** The MVP will then be evaluated in terms of its functionality, performance, scalability, and reliability. Feedback from this evaluation will be used to identify areas for improvement.

## 1.6 Report structure

**Chapter 1. Introduction:** This chapter provides a general overview of the topic, its potential, and practical applications in the future. It introduces the concept of a MEVN-based e-commerce application for a sport equipment shop website.

**Chapter 2. Literature review** This section provides a summary of relevant research on the topic and the foundational knowledge necessary to develop an e-commerce application using the MEVN stack.

**Chapter 3. System design and implementation** This section presents the approach to the problem. It discusses the details of the implementation approach, including the tools used.

**Chapter 4. Testing and evaluation.** This chapter presents the testing plan and management, testing scenarios for the main functions of the system.

**Chapter 5. Conclusion** This section presents the results achieved, the remaining limitations, and the system's further development.

# Chapter 2

# Literature reiview

The aim of this literature review is to provide a comprehensive understanding of the key concepts and technologies that underpin this project. These include the MEVN stack and front-end development using Vue.js. This review will serve as the theoretical foundation for the design and implementation of a MEVN-based e-commerce application for a sport equipment shop website.

In the next section, we will delve deeper into the MEVN stack, starting with its definition and core principles.

## 2.1 Microservices Architecture

### 2.1.1 Overview of Microservices Architecture

Microservices architecture is a design approach where an application is built as a collection of small services. Each of these services runs in its own process and communicates with others using lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and are independently deployable by fully automated deployment machinery.

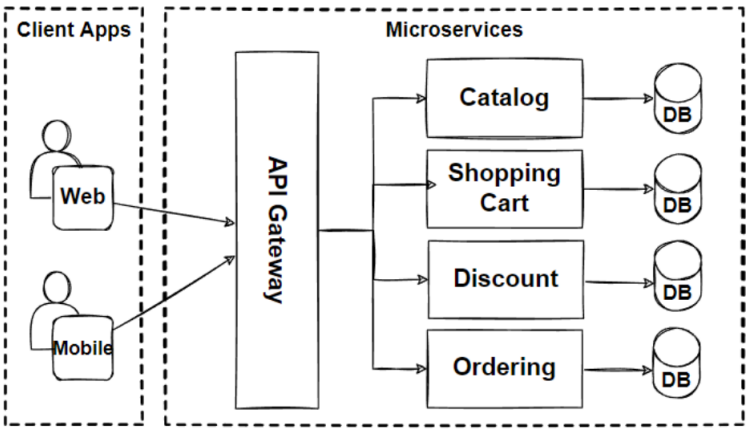


Figure 2.1: Microservices

For instance, consider an e-commerce application. In a monolithic architecture, all functionalities like user management, product catalog, order processing, and payment system would be in a single codebase. In contrast, in a microservices architecture, each of these functionalities would be a separate service with its own database and business logic.

### 2.1.2 Benefits of Microservices:

The benefits of using microservices architecture in e-commerce applications are numerous. They include improved scalability as each service can be scaled independently based on demand, increased resilience as the failure of a single service does not affect the entire system, and enhanced agility and speed in bringing new features to market.

**Amazon** is a prime example of the benefits of microservices. Originally, Amazon had a monolithic architecture, but as their services expanded, they moved to a microservices architecture. This allowed them to scale individual services based on demand. For example, during a sale, the order processing service could be scaled up without affecting other services.

### 2.1.3 Challenges of Microservices:

Despite its benefits, microservices architecture presents several challenges:

- **Complexity:** Managing multiple services can be complex, requiring sophisticated coordination and orchestration.
- **Distributed Data Management:** Handling distributed data across services can be difficult and requires careful consistency management.
- **Communication Overhead:** Implementing communication between services can introduce additional latency and resource usage.

**Netflix** faced challenges with microservices when they first transitioned from a monolithic architecture. They had to deal with complexities in managing multiple services and ensuring data consistency across services. They also faced challenges in implementing communication between services.

## 2.2 Microservices in ASP.NET

In this project, a new souvenir e-shop will be implemented using a microservices architecture from the ground up with ASP.NET. Each feature such as product management, inventory management, order processing, and payment processing will be implemented as a separate microservice running in its own Docker container.

### 2.2.1 ASP.NET and Microservices

ASP.NET is a robust framework for .NET that simplifies the creation of APIs, which form the backbone of microservices. It provides built-in support for Docker containers,

which are lightweight and can be easily managed, making them ideal for deploying microservices.

ASP.NET offers high performance, which is critical in a microservices architecture where each service should be highly responsive. The high throughput of .NET surpasses many popular frameworks, making it an excellent choice for building efficient microservices.

### 2.2.2 Communication Between Services

In a microservices architecture built with ASP.NET, secure communication between services is crucial. Technologies like HTTPS provide secure communication over a computer network, while RabbitMQ with MassTransit in ASP.NET offers a reliable and efficient message broker system, which is essential for communication between microservices.

### 2.2.3 Docker and .NET Tools

Docker has become a standard in the container industry due to its ability to encapsulate microservices in containers, enhancing their isolation, portability, and scalability. .NET is built to work seamlessly with Docker, allowing developers to focus on building their microservices without worrying about compatibility issues.

The Visual Studio family of products offers comprehensive tools for developing ASP.NET applications with Docker support on Windows. These tools simplify the process of configuring applications for Docker and allow developers to step through their code line-by-line as it runs in a Docker container.

## 2.3 Introduction to Flutter and Clean Architecture

### 2.3.1 Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase. Key features of Flutter include its fast development cycle, expressive and flexible UI, and native performance.

### 2.3.2 Clean Architecture

Clean Architecture is a software design philosophy that separates software into layers with ‘stable dependencies’. This architecture promotes the separation of concerns, making the system easier to manage and maintain. It is important because it makes the system more flexible, maintainable, and scalable.

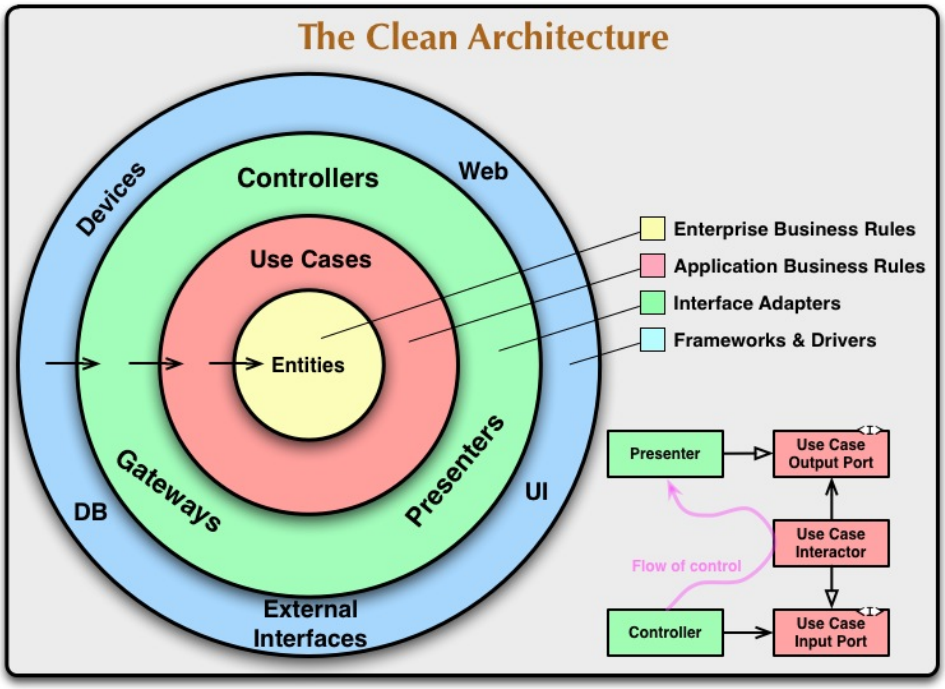


Figure 2.2: Clean architecture by Robert C. Martin [2]

### 2.3.3 Applying Clean Architecture in Flutter

Clean Architecture can be applied in Flutter by separating the application into three layers: Presentation, Domain, and Data. The Presentation layer handles the UI and user interactions. The Domain layer contains business logic and use cases. The Data layer manages data from various sources like network requests and databases.

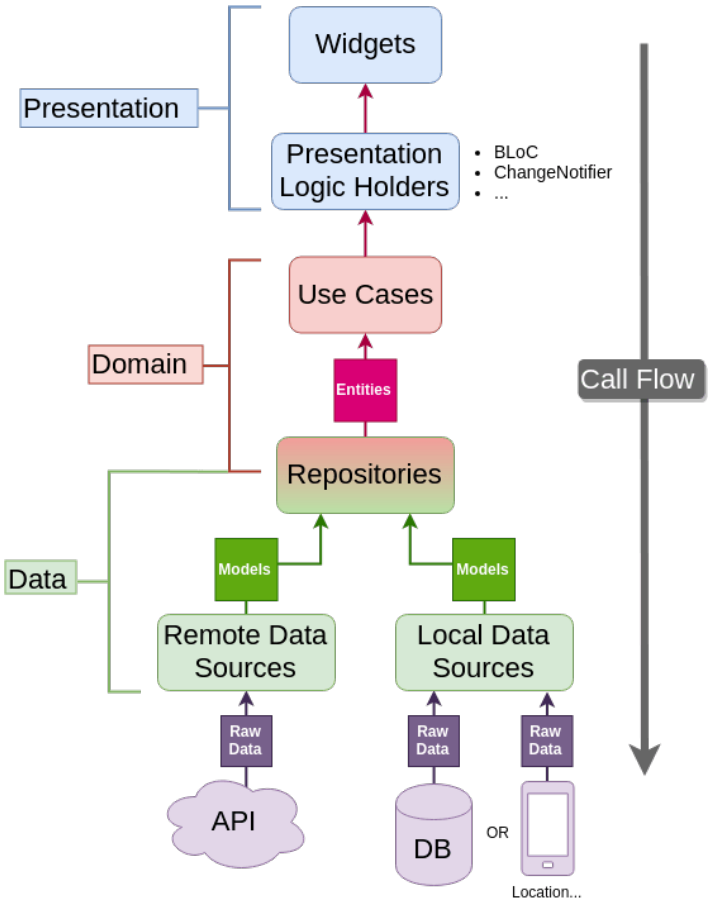


Figure 2.3: Clean Architecture Diagram by Reso Coder[1]

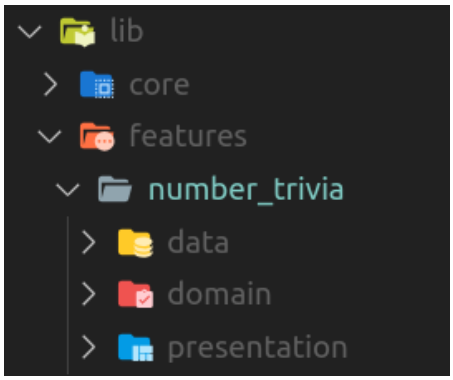


Figure 2.4: Example of structure in Flutter project

## 2.4 Applications of Flutter and Clean Architecture

### 2.4.1 Accelerated MVP Development with Flutter

Flutter's ability to deliver high-performance applications on Android and iOS from a single codebase makes it an excellent choice for rapid MVP development. Its rich set of widgets and reactive framework allow for a fast development cycle, enabling developers to quickly implement key features and bring the product to market.

### 2.4.2 Case Studies of Flutter and Clean Architecture

There are several case studies of applications developed using both Flutter and Clean Architecture. These applications demonstrate how Clean Architecture can enhance the maintainability and scalability of Flutter applications, even when under the constraints of rapid MVP development.

Flutter has been used in various real-world applications due to its ability to deliver high-performance applications on Android and iOS from a single codebase. Some examples include the Alibaba e-commerce app, the Toyota: Improving Infotainment Systems app,

and Google Ads.

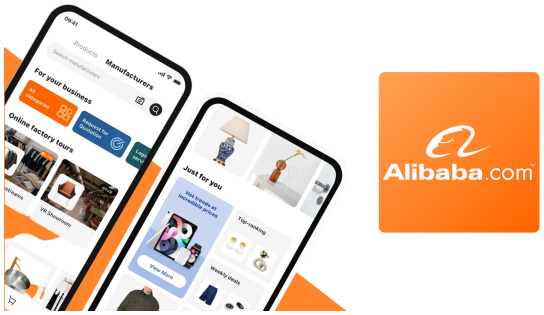


Figure 2.5: Alibaba Group: Ecommerce

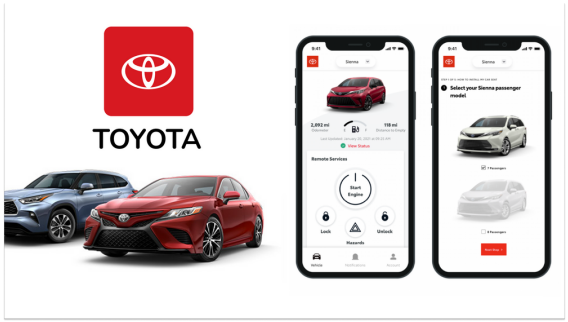


Figure 2.6: Toyota: Improving Infotainment Systems

### 2.4.3 Impact of Flutter and Clean Architecture

The use of Flutter along with Clean Architecture has been shown to improve the development process by making it easier to manage codebases, implement new features, and fix bugs. It also enhances the performance of applications by ensuring they run smoothly on different platforms.

## 2.5 Microservices, Flutter, and E-commerce

### 2.5.1 Application of Microservices and Flutter in E-commerce

Microservices architecture and Flutter can be effectively applied in the context of e-commerce. Microservices allow for the separation of functionalities into independent services such as user authentication, product catalog, order management, and payment processing1. Flutter, with its fast development cycle and expressive UI, can be used to build the frontend of the e-commerce application.

### 2.5.2 Benefits and Challenges

The benefits of using these technologies in e-commerce include improved scalability, faster deployment times, better fault isolation, and more flexibility in implementing new features. However, challenges may include increased dependency on technology, large costs involved especially for small businesses, risk of job cuts due to automation, security risks related to data and fraud, and dealing with issues like shopping cart abandonment and maintaining customer loyalty.

### 2.5.3 Case Studies

Several e-commerce giants like eBay, Etsy, Gilt and Zalando have transformed their infrastructures into microservice architectures to create flexible, global systems. The combination of these technologies holds promise for efficient and scalable e-commerce application development.

# Chapter 3

# System design and implementation

## 3.1 Section 1

# Chapter 4

# Testing and evaluation.

## 4.1 Section 1

# Chapter 5

# Conclusion

## 5.1 Section 1

# Bibliography

[1] Reso Coder. Flutter tdd clean architecture - explanation and project structure, 2019.

[2] Robert C. Martin. The clean architecture, 2012. Accessed: 2023-10-24.