VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

FINAL PROJECT

WEB APPLICATION USING NODEJS

POINT OF SALE WEBSITE

Instructor: MR. **MAI VĂN MẠNH**

Implementer: **NGUYỄN THÀNH TÀI – 521H0481**

NGUYỄN VIẾT MINH DUY – 521H0445

ĐỖ ĐỨC GIA KHANG – 520H0540

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

FINAL PROJECT

WEB APPLICATION USING NODEJS

POINT OF SALE WEBSITE

Instructor: MR. **MAI VĂN MẠNH**

Implementer: **NGUYỄN THÀNH TÀI – 521H0481**

NGUYỄN VIẾT MINH DUY – 521H0445

ĐỖ ĐỨC GIA KHANG – 520H0540

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

ACKNOWLEDGEMENT

To complete this final project, we would like to express our gratitude to Ton Duc Thang University for providing the necessary facilities and to the teachers who have supported us throughout our studies at the university. Moreover, we love to give thanks to our parents who gave us chances to approach knowledge at TDTU and motivated, supported us a lot.

Especially, we would like to thank Mr. Mai Văn Mạnh for teaching us with great dedication and detail, so that we have enough knowledge to use for this essay. Due to our limited experience and knowledge, we are sure that there are some mistakes in our work. We sincerely hope to receive feedbacks and constructive criticism from our teacher so that we can complete this essay more effectively.

We would like to express my heartfelt thanks and wish teacher good health.

TP. Hồ Chí Minh, ngày tháng năm 2023

Tác giả

(ký và ghi rõ họ tên)

Nguyễn Thành Tài

Nguyễn Viết Minh Duy

Đỗ Đức Gia Khang

THE REPORT IS COMPLETED
AT TON DUC THANG UNIVERSITY

We assure you that this is our essay product and under the guidance of Mr. Mai Văn Mạnh. The research contents and results on this topic are honest and have not been published in any previous forms. The figures in the tables serving the analysis, comments, and assessments were collected by the author from different sources specified in the references.

If any fraud is found, We are fully responsible for the content of my report. Ton Duc Thang University is unrelated to copyright and copyright infringement caused by us during the implementation process (if any).

TP. Hồ Chí Minh, ngày tháng năm 2023

Tác giả

(ký và ghi rõ họ tên)

Nguyễn Thành Tài

Nguyễn Viết Minh Duy

Đỗ Đức Gia Khang

PART CERTIFICATION AND ASSESSMENT OF THE LECTURERS

The certification part of the instructor

_____

_____

_____

_____

_____

_____

Ho Chi Minh City, Date: / /2023.

(sign and state full name)

The evaluation part of the teacher rated

_____

_____

_____

_____

_____

_____

_____

Ho Chi Minh City, Date: / /2023.

(sign and state full name)

LIST OF PICTURES

SUMMARY

The Point of Sale system is a digital platform used in retail stores for processing sales transactions. It typically involves managing products, handling customer data, and conducting sales. The project in the report involves building a web-based POINT OF SALE application for phone and accessory sales. This application includes features like account management, product catalog, customer database, and transaction processing. The aim is to streamline sales processes in retail environments, making them more efficient and organized.

TABLE OF CONTENTS

ACKNOWLEDGEMENT i
THE REPORT IS COMPLETED ii
PART CERTIFICATION AND ASSESSMENT OF THE LECTURERS iii
LIST OF PICTURES iv
SUMMARY v
CHAPTER 1: INTRODUCTION 1
1.1 Implementation goals and achieved results 1
CHAPTER 2: OVERVIEW 2
2.1 Overview of system architecture 2
2.2 Overview of System Features: 3
CHAPTER 3: TECHNOLOGIES, LIBRARIES, SERVICES 4
3.1 Overview technologies, libraries, services used to develop website 4
3.2 Comparison table for Point of Sale system libraries, technologies, services: 4
CHAPTER 4: ANALYZE AND EVALUATE 11
4.1 Problem and solution: 11
4.2 Analyze and evaluate: 11
CONCLUSION 14
APPENDIX 16
REFERENCES 17

CHAPTER 1: INTRODUCTION

1.1 Implementation goals and achieved results

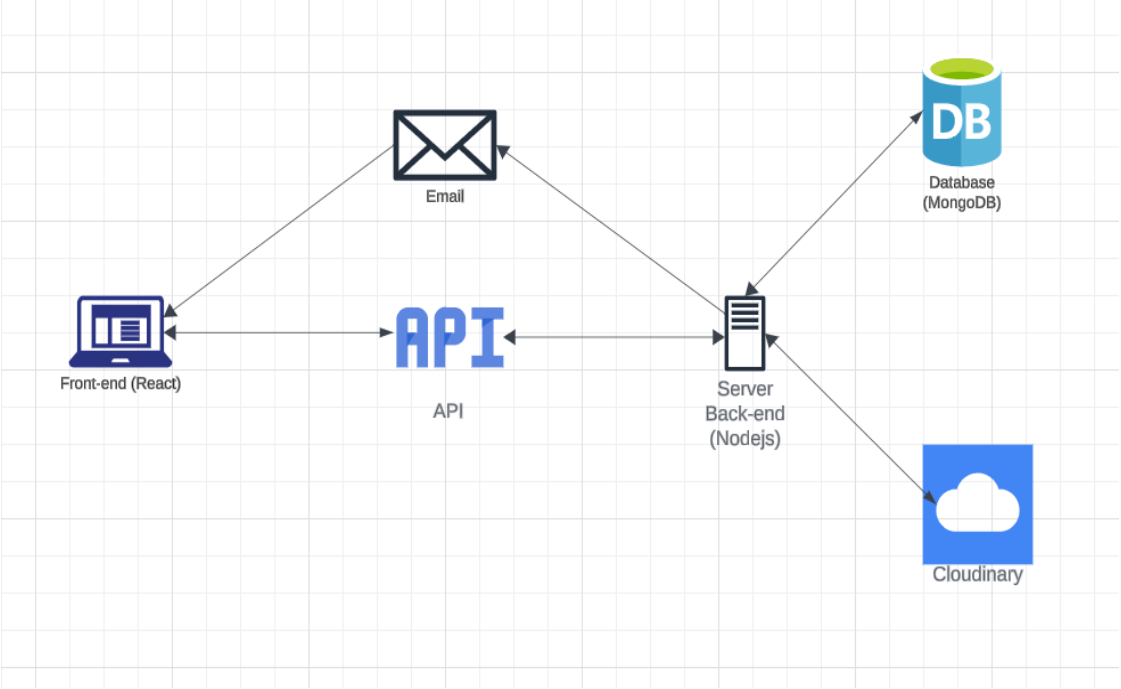
- **Implementation goals**: the web-based Point of Sale system was created to update and streamline retail operations, particularly in a store specializing in phones and accessories. The system was developed in response to the demand for efficient digital solutions in retail settings with intricate transactions and inventory management.

- **Achieved results**: this Point of Sale system distinguishes itself with its user-friendly interface and robust functionality, encompassing account management, product catalog management, customer management, transaction processing, and reporting & analytics. These features enhance operational efficiency, reduce errors, and provide valuable insights into sales trends and customer behavior, effectively addressing significant challenges in retail management.

  Utilizing modern web technologies and Node.js, the Point of Sale system delivers a seamless, integrated solution that simplifies retail management, ultimately enhancing the overall shopping experience for both staff and customers.

CHAPTER 2: OVERVIEW

2.1 Overview of system architecture



In this project, we developed a Node.js-based web application designed for Point of Sale functionalities in a retail store for phones and accessories. Utilizing a range of front-end and back-end technologies, our system streamlines sales transactions, product management, and employee management.

Front-end Development: We employed a modern stack including React for building user interfaces, TailwindCSS for styling, Zustand for state management, and various other libraries like Axios, date-fns, and framer-motion for enhancing functionality and user experience.

Back-end Development: Our server-side logic is powered by Nodejs (Express), with Mongoose for database interactions. We integrated cloud services like Cloudinary for media management and used JWT for secure authentication.

2.2 Overview of System Features:

1. Account Management: This includes built-in administrator accounts and the creation of salesperson accounts by the admin. It involves sending automatic emails for account creation, login via email links, and mandatory password changes on first login.

2. Product Catalog Management: Administrators can view, add, update, and delete products. Sales staff can view product lists but can't modify them.

3. Customer Management: Sales staff enter customer details at checkout, and the system automatically recognizes returning customers. Employees can view customer personal information and purchase history.

4. Transaction Processing: Sales staff can add products to a purchase through search or barcode entry. The system updates the total amount automatically, and the checkout process includes customer information entry and invoice generation.

5. Reporting and Analytics: Both admin and sales staff can view sales results across different timelines, with detailed information on orders, total amounts, and products.

CHAPTER 3: TECHNOLOGIES, LIBRARIES, SERVICES

3.1 Overview technologies, libraries, services used to develop website

- **Technologies:**

  Front-end: React (NextJS), TypeScript

  Back-end: Node.js (ExpressJS), JavaScript

- **Libraries:**

  Front-end: shadcn/ui, react/table, axios, cookies-next, date-fns, framer-motion, lucide-react, next-cloudinary, uuid, zod, zustand, recharts, react-hot-toast, react-hook-form, tailwindcss

  Back-end: multer, mongoose, cors, jsonwebtoken, gravatar, multer-storage-cloudinary, nodemailer, bcrypt

- **Services:**

  Cloudinary (used for media management both in front-end and back-end)

3.2 Comparison table for Point of Sale system libraries, technologies, services:

Library / Technology / Service	Function	Advantages	Disadvantages	Alternatives (if any)	Why choosen?
Front-end					
React (NextJS)	UI framework	Cross-platform, component-based, large community	Steep learning curve, can be heavy for small projects	Vue.js, Angular	Robust and flexible for dynamic UIs
TypeScript	Type safety	Improves code stability, reduces errors, enhances	Learning curve, stricter syntax compared to JavaScript	None	Improves code quality and maintainability

		developer experience			
shadcn/ui	Provides customizable Point of Sale UI components	Streamlines development, offers pre-built components for common Point of Sale tasks	Less customizable than pure React components, may have limited feature set	React + other UI libraries (requires more coding)	Faster development and consistency for common Point of Sale functionalities
react/table	Enables efficient rendering of large datasets	Improves performance for displaying product tables and other data-rich interfaces	Requires additional configuration and styling, may not be suitable for small datasets	Hand-built React tables (more effort but potentially more flexible)	High performance and efficient for large data sets
axios	Makes HTTP requests to the back-end	Simplifies API calls and data fetching, offers clean syntax and promise-based approach	Can be verbose for simple requests, requires additional libraries for handling complex responses	Fetch API (built-in browser API, but less feature-rich)	Simplifies API calls and data fetching
cookies-next	Manages user sessions and authentication securely	Provides secure storage for session data, simplifies user login and authorization	Requires configuration and integration with back-end authentication system, can be complex for beginners	JWT libraries (manage tokens directly, but require more coding)	Secure and convenient for user authentication
date-fns	Provides utilities for working with dates and times	Simplifies date manipulation tasks, offers formatting	Can be feature-rich for simple needs, may have a slight learning curve	Built-in JavaScript Date object (less powerful but simpler)	Flexible and lightweight for basic date needs

		and parsing options			
framer-motion	Adds animations and transitions	Enhances user experience, improves visual appeal and interactivity	Can be complex to implement for advanced animations, may impact performance on slower devices	CSS animations (less powerful but lighter weight)	Improves user experience without compromising performance
lucide-react	Provides scalable and customizable icons	Offers consistent and flexible icons for the Point of Sale interface, improves visual design	Limited icon library compared to other options, requires additional configuration for customization	Material UI icons (larger library but potentially heavier)	Consistent and flexible icons with a larger library
next-cloudinary	Integrates Cloudinary for image management	Simplifies image uploading, delivery, and optimization within the front-end	Requires Cloudinary account, adds dependency on a third-party service	Direct image upload and CDN integration (more complex but potentially more flexible)	Integrates seamlessly with Cloudinary for optimized images
uuid	Generates unique identifiers for items and transactions	Ensures data integrity and avoids conflicts, improves data tracking	Can be replaced with auto-incrementing IDs for simple needs, may not be necessary for all data	None	Efficient for unique IDs and tracking transactions
zod	Validates user input and data integrity	Improves data accuracy, prevents invalid inputs	Adds another layer of complexity, requires additional	Built-in form validation libraries (simpler but less robust)	Enhances data quality and prevents errors

		from reaching the back-end	configuration for custom validation rules		
zustand	Manages global state and data across components	Simplifies state management in complex applications, avoids prop drilling	Adds another dependency, requires careful implementation to avoid performance issues	Redux (mature state management library, but steeper learning curve)	Lightweight and efficient for managing global state
recharts	Creates interactive data visualizations	Enables insightful sales reports and data analysis within the Point of Sale system	Adds complexity to the front-end, requires data formatting and manipulation	Chart.js (popular alternative, but less feature-rich)	Creates interactive visualizations with a simpler setup
react-hot-toast	Displays user feedback notifications	Provides a convenient way to notify users about actions and errors	Can be intrusive if used excessively, requires careful styling to match the UI	Built-in browser notifications (less customizable but simpler)	Effective user feedback without cluttering the UI
react-hook-form	Facilitates form building and validation	Simplifies form creation and validation process, improves user experience	Adds dependency and complexity, requires additional configuration for custom validation rules	Built-in HTML forms (less powerful but simpler)	Improves user experience and form completion
tailwindcss	Provides a utility-first CSS framework	Speeds up UI development, offers responsive and	Can be verbose for complex layouts, requires familiarity with	Bootstrap (popular alternative, but less utility-first)	Efficient and consistent styling with a shorter learning curve

		customizable styles	the Tailwind syntax		
Back-end					
Node.js (ExpressJS)	Server framework	Lightweight, efficient, high performance	Requires familiarity with JavaScript and server-side development	Python/Django	Ideal for Point of Sale systems due to performance and scalability
multer	File uploads	Handles file uploads for product images and other media	Can be complex to configure for advanced use cases	Built-in file upload libraries	Simplifies file upload for images and media
mongoose	Object-document mapper	Flexible and powerful for interacting with the database	Can be heavy for small projects	Sequelize	Efficient and well-suited for relational databases
cors	Cross-origin resource sharing	Enables secure access to data from the back-end	Requires configuration for specific access control needs	None	Secure access to data between front-end and back-end
jsonwebtoken	User authentication and token generation	Secure access control, simplifies user authentication	Requires additional setup and configuration	JWT libraries	Robust and widely used for secure user authentication
gravatar	User avatars	Provides personalized avatars from Gravatar service	Adds dependency on external service, limited customization options	Built-in user avatar generation	Easy integration for basic user avatars
multer-storage-cloudinary	Image storage and optimization	Stores uploaded images in Cloudinary, offers image optimization	Requires Cloudinary account, adds dependency on third-party service, can be	Local Storage	Integrates seamlessly with Cloudinary for optimized

		and transformation capabilities	expensive for large volumes of images		images and scalability
nodemailer	Email notifications	Sends email notifications for salesperson	Can be complex to configure initially, requires SMTP server setup, may not be necessary for simple notifications	SendGrid (managed email service, easier to set up but potentially more expensive), built-in email APIs of some cloud providers	Simplifies email sending and integrates with Point of Sale backend, offers templates and scheduling options
bcrypt	Password hashing	Securely stores user passwords, protects against brute-force attacks	Adds another layer of complexity, requires additional configuration and libraries	None	Essential for secure user authentication and data protection
Third-party					
Cloudinary	Secure and scalable cloud storage	Protects data from loss, scales to accommodate large image volumes, offers redundancy and disaster recovery.	Requires third-party service, incurs additional storage and bandwidth costs compared to local storage.	Local storage (suitable for small-scale applications, less secure).	Protects data, scales with business growth.
	Image optimization and transformation	Improves page loading times, reduces bandwidth usage, enhances visual quality.	Requires configuration and optimization settings, may not be suitable for all image	Server-side image processing (requires more development effort, less flexibility).	Improves page loading and user experience.

			formats or use cases.		
	Easy image management and delivery	Provides APIs and SDKs for integration with Point of Sale applications, simplifies image upload, delivery, and manipulation.	Adds another dependency, may require additional configuration for security and access control.	Direct image upload and CDN integration (more complex to manage, requires server-side scripting).	Simplifies upload, delivery, and manipulation.

CHAPTER 4: ANALYZE AND EVALUATE

4.1 Problem and solution:

- **Problem Statement:** storing images locally posed challenges, including increased server pressure and the lack of synchronization across devices. This hindered our goal of creating a seamless and efficient image management system.
- **Solution Overview:**
  - To address these challenges, we opted to transition from local storage to third-party cloud storage. This shift aims to alleviate server strain and enhance cross-device image synchronization.
  - Cloudinary is a cloud-based media management solution that offers versatile image and video storage, manipulation, and delivery services. By utilizing Cloudinary, we tap into a scalable and reliable platform designed to optimize media management in a cloud environment.

4.2 Analyze and evaluate:

Table of Comparison: saving images in Cloudinary and Local Storage:

Feature	Cloudinary	Local Storage
Scalability	Extremely scalable, handles high traffic and image volume efficiently	Limited scalability, can bottleneck with increased traffic
Availability	Global content delivery network (CDN), high availability even during outages	Server-dependent, vulnerable to downtime and outages
Security	Built-in security features, data replicated across data centers	Requires manual security implementation, vulnerable to data breaches

Image transformations	On-the-fly transformations (resizing, cropping, watermarking)	Requires server-side processing or additional libraries
Content Delivery Network (CDN)	Delivers images through global CDN for fast loading	Images served from server location, slower loading times for distant users
Ease of integration	SDKs and plugins for popular languages and frameworks, simplifies integration with your application	Requires manual integration and management
Cost	Pay-as-you-go model, can be cost-effective for high traffic	Lower cost for low traffic, but additional storage and bandwidth costs
Control	Limited control, images stored on Cloudinary's servers	Full control over storage location and security
Security considerations	Trust required in Cloudinary's security practices	Requires implementation and maintenance of your own security measures

Table of Evaluation: saving images in Cloudinary and Local Storage:

Feature	Cloudinary	Local Storage
Scalability	Good	Bad
Availability	Good	Bad
Security	Good	Bad
Image transformations	Good	Bad
Content Delivery Network (CDN)	Good	Bad

Ease of integration	Good	Bad
Cost	Pay-as-you-go	Upfront costs
Control	Limited	Full
Security considerations	Good	Bad

Conclusion: for high-traffic applications, Cloudinary is the best choice because it offers superior scalability, availability, security, and performance. For low-traffic applications, Local Storage is a good option if you need full control over your images and security.

CONCLUSION

- **Key takeaways**:
  - Front-end: React (NextJS) and TypeScript provide a powerful and flexible foundation for building the UI, while selected libraries like shadcn/ui, react/table, and axios address specific components and functionalities like data presentation, user interaction, and API communication.
  - Back-end: Node.js (ExpressJS) offers a lightweight and efficient server framework, and libraries like multer, mongoose, and jsonwebtoken handle file uploads, database interaction, and user authentication securely.
  - Overall integration: Cloudinary seamlessly manages image storage and optimization, and libraries like nodemailer and bcrypt ensure secure communication and data protection.

- **Benefits:**
  - Performance and scalability: The chosen technologies are known for their efficiency and ability to handle large datasets and workloads, making them suitable for growing businesses.
  - User experience: The front-end libraries focus on creating a smooth, intuitive, and visually appealing interface for users, while back-end functionalities ensure seamless data processing and communication.
  - Security: bcrypt and secure authentication methods safeguard user data and transactions, while secure communication protocols between front-end and back-end further enhance security.
  - Development efficiency: Many libraries offer pre-built components and functionalities, speeding up development and reducing complexity for developers.

- **Conclusion:**

  The combination of technologies and libraries chosen for this Point of Sale system prioritizes a robust, scalable, and user-friendly experience while addressing specific functionalities crucial for the application.

APPENDIX

REFERENCES

https://www.npmjs.com