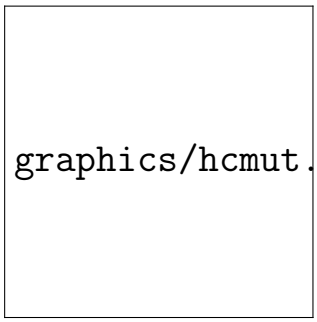
VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



REPORT
SPECIALIZED PROJECT

BUILDING A SYSTEM FOR ENTERPRISES TO INTRODUCE SERVICES AND QUOTATION IN THE FIELD OF TRANSPORTATION AND FREIGHT FORWARDING

Major: Computer Science

COUNCIL: COMPUTER SCIENCE - CLC 04
SUPERVISOR(s): Dr. Nguyễn Thị Ái Thảo

M.Sc. Trần Thị Quế Nguyệt graphics/signatu

—o0o—

STUDENT 1: Hoàng Đại Nam - 2153594
STUDENT 2: Vũ Minh Quân - 2152926

HO CHI MINH CITY, APRIL 2025

Declaration

We guarantee that this research is our own, conducted under the supervision and guidance of Dr. Nguyen Thi Ai Thao and M.Sc. Tran Thi Que Nguyet The result of our research is legitimate and has not been published in any forms prior to this. All materials used within this researched are collected by ourselves, by various sources and are appropriately listed in the references section. In addition, within this research, we also used the results of several other authors and organizations. They have all been aptly referenced. In any case of plagiarism, we stand by our actions and are to be responsible for it. Ho Chi Minh City University of Technology therefore are not responsible for any copyright infringements conducted within our research.

Acknowledgements

First and foremost, we would like to express our sincere gratitude to Dr. Nguyen Thi Ai Thao and M.Sc. Tran Thi Que Nguyet. Their invaluable support, guidance, and constructive feedback throughout both the planning and implementation phases have been instrumental in improving our system design and ensuring the success of this Specialized Project.

We are also deeply indebted to the lecturers at the Faculty of Computer Science and Engineering, Vietnam National University - Ho Chi Minh City University of Technology. The strong foundation of knowledge they have provided has enabled us to complete this thesis and contribute meaningfully to Vietnam's computer science field.

Throughout this four-month journey, we have faced numerous challenges, particularly during the initial stages. However, through careful planning, persistence, and dedication, we successfully maintained our progress and completed the project within the designated timeframe.

Finally, we extend our heartfelt appreciation to our families and friends for their unwavering emotional support throughout our academic journey. The exemplary work ethic demonstrated by our parents has served as our greatest source of inspiration and motivation to excel in our endeavors.

Abstract

The transportation and freight forwarding sector faces challenges in efficiently managing customer orders and providing accurate, real-time quotations. Companies in this field often struggle with complex logistics and communication between multiple parties, such as transport units, customs, and warehouses.

To address these challenges, FreightFlex is a software system designed to assist businesses in the transportation and freight forwarding sector with managing customer orders. The system is intended for companies that serve as intermediaries between transport providers, customs, warehouses, and customers, helping them manage orders and plan import-export logistics. The system includes features for creating landing pages to introduce services and provide quotations to end users, as well as a chatbot to facilitate customer communication. The outcome will be a practical, user-friendly solution aimed at improving efficiency and customer service for businesses in the transportation and freight forwarding sector.

Chapter Summary

Chapter 1: Introduction to the Topic
Introduction to the overview, significance, objectives, and limitations of the thesis topic.

Chapter 2: Foundation Knowledge
Introduction to fundamental theoretical knowledge, new technologies applied including programming languages, frameworks, libraries, third-party services,.... that used in building and deploying the system.

Chapter 3: System Analysis
This chapter presents system survey information, specifications, and analysis of potential features. It defines the actors and their roles, and presents use case diagrams illustrating the system's interactions.

Chapter 4: System Design
This chapter covers the database design, overall system architecture, and system interfaces. It describes the design methodology and approaches used in developing the system.

Chapter 5: System Implementation
This chapter introduces the libraries used in the system, implementation methods, and execution flow through the description of the system interface.

Chapter 6: Deployment, Testing and System Evaluation
This chapter details the deployment process, testing methodologies, and evaluation of system performance. It explores ways to improve system efficiency and reliability.

Chapter 7: Summary, Evaluation and Development Direction
This chapter summarizes the achievements throughout the project implementation, evaluates the system through its strengths and weaknesses, and provides necessary development directions for the system in the future.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Freight forwarding is a critical component of global trade, serving as the backbone of international commerce by facilitating the movement of goods across borders. A freight forwarder functions as an intermediary that organizes shipments for individuals or corporations, negotiating with various transportation providers to move goods from manufacturers or producers to market, or from vendors to final customers[38].

In today's complex global supply chains, freight forwarders coordinate multiple aspects of shipment logistics including documentation, cargo insurance, customs clearance, storage, and inventory management. They provide expertise in international shipping regulations, transportation methods, and customs requirements that many businesses lack internally. The industry handles a significant portion of global trade volume, making it an essential service for companies engaged in international commerce.

However, the freight forwarding industry faces significant challenges in the digital era:

- Manual quotation processes that often require substantial time to generate estimates[9]
- Lack of transparency in pricing and service options.
- Communication inefficiencies between forwarders, clients, and transport providers.
- Difficulty in providing real-time tracking and status updates.
- Complex documentation requirements that vary by region and transport mode.

The FreightFlex system addresses these challenges by providing a comprehensive platform that enables freight forwarding companies to create customized landing pages and offer instant quotations. The system aims to streamline operations, improve client experience, and provide

forwarders with modern digital tools to remain competitive in an increasingly technology-driven industry.

1.2 Research Objectives

The primary objective of this research is to design and develop a comprehensive web-based system that allows freight forwarding enterprises to create customized landing pages for introducing their services and providing automated quotations to clients. This system will enhance communication between forwarders and clients while streamlining operational processes.

Specifically, the research aims to:

1. Analyze the operations and requirements of transportation and freight forwarding management systems to identify key challenges and opportunities for digitalization
2. Design and implement a flexible, multi-tenant architecture that supports multiple freight forwarding companies while maintaining data isolation
3. Develop an intuitive interface for freight forwarders to create customized landing pages that showcase their services
4. Implement an automated quotation system that handles the complexities of freight pricing across different transportation modes and routes
5. Create a prototype demonstrating the core features of the system
6. Evaluate the system's performance, usability, and effectiveness in meeting industry needs

1.3 Significance of the Research

1.3.1 Practical Significance

This research addresses pressing operational challenges in the freight forwarding industry and offers practical solutions with significant business impact:

- **Operational Efficiency**: By automating the quotation process, freight forwarders can reduce the time required to generate quotes from days to minutes, allowing staff to focus on value-added activities rather than manual calculations.
- **Enhanced Customer Experience**: The system provides clients with instant access to pricing information, service options, and shipment tracking, increasing transparency and satisfaction.

- **Competitive Advantage**: In an industry where many companies still rely on manual processes, digitalization through FreightFlex offers forwarders a significant competitive edge by providing faster service and a more professional appearance.
- **Cost Reduction**: Digital automation reduces operational costs associated with manual quotation preparation, document handling, and client communication, with potential savings in administrative overhead.
- **Resource Optimization**: The system helps companies better allocate human resources by automating routine tasks, allowing specialized staff to focus on complex logistics challenges.

1.3.2 Scientific Significance

This research contributes to the academic and technical advancement in several areas:

- **Multi-tenant Architecture**: The implementation explores efficient approaches to data isolation and resource sharing in cloud-based logistics applications, contributing to the body of knowledge on secure multi-tenant systems.
- **Industry-Specific UI/UX Design**: The research investigates effective user interface patterns for complex logistics operations, providing insights into domain-specific design principles.
- **Integration Methodologies**: The system demonstrates novel approaches to integrating modern web technologies with traditional logistics processes, establishing patterns that can be applied to other sectors.
- **Logistics Process Digitalization**: The research documents the transformation of manual logistics workflows into digital processes, contributing to the broader field of business process automation in specialized industries.

1.4 Limitations

While the system aims to provide comprehensive solutions for freight forwarders, certain limitations exist in the current scope:

- The system primarily focuses on standard shipping modes (road freight) and may not fully accommodate specialized shipping types such as hazardous materials, livestock transport, or project cargo.

1.4. Limitations

- The initial implementation targets small to medium-sized freight forwarding companies and may require adaptation for enterprise-scale operations.
- The current version emphasizes freight forwarding operations rather than full supply chain management, with limited inventory management capabilities.
- Integration capabilities exist for major shipping lines and carriers but may not cover all regional or specialized transport providers.
- While the system includes basic reporting functionality, complex business intelligence and predictive analytics features are targeted for future development.

These limitations represent opportunities for future enhancements as the system evolves to meet broader industry needs and incorporate emerging technologies.

Chapter 2

Foundation Knowledge

2.1 Theoretical Basis

2.1.1 Software Architecture Patterns

2.1.1.1 Client-Server Architecture

Client-Server architecture is a distributed application structure that partitions tasks between providers of resources or services, called servers, and service requesters, called clients[8]. The client-server model has become one of the central concepts of network computing, enabling multiple clients to connect to a server to access resources without requiring direct client-to-client communication.

In the context of freight forwarding systems, this architecture is particularly valuable as it allows:

- Centralized data management for shipment tracking and documentation
- Concurrent access from multiple stakeholders (staff, clients, administrators)
- Clear separation of user interface from business logic and data storage

Our system leverages this architecture to provide a responsive web application that communicates with a robust backend server, ensuring data integrity and system security while maintaining performance across multiple user sessions.

2.1.1.2 Monolithic Architecture

Monolithic architecture[21] represents a traditional unified approach where all components of an application are interconnected and function as a single unit. As described in our current implementation, this architecture offers several advantages:

- Simplified development process and code management
- Reduced cognitive load during initial implementation phases
- Streamlined debugging and deployment workflows

For our freight forwarding system, the monolithic approach provides an efficient development path given our resource constraints while maintaining the ability to transition to microservices in future iterations if scaling demands increase.

Figure 2.1.1: Monolithic Architecture

2.1.1.3 Multi-tenant Architecture

Multi-tenancy refers to an architecture where a single instance of software serves multiple customers (tenants)[1]. Each tenant's data is isolated from other tenants, though they share the application and database infrastructure.

In our implementation, we utilize schema-based multi-tenancy within PostgreSQL, where:

- Each freight forwarding company occupies a separate database schema
- Common application code serves all tenants
- Configuration data is tenant-specific
- Authentication mechanisms ensure proper data isolation

This approach enables our system to serve multiple freight forwarding companies efficiently while maintaining strict data privacy and separation.

2.1.2 Web Application Technologies

2.1.2.1 Application Programming Interfaces (APIs)

APIs serve as intermediary protocols that enable different software applications to communicate with each other. In modern web development, RESTful APIs have become the standard for building interoperable systems [26].

For our freight forwarding system, we implement RESTful APIs characterized by:

- Statelessness: Each request contains all information needed for processing
- Resource-based routing: Clear endpoint organization based on business entities
- Standard HTTP methods: Using GET, POST, PUT, DELETE for CRUD operations
- JSON data format: Lightweight data interchange for client-server communication

This API design facilitates seamless communication between our frontend and backend components while providing the foundation for potential third-party integrations in future system iterations.

2.1.2.2 Single Page Applications (SPA)

Single Page Applications represent a modern web development approach where the entire application loads a single HTML page and dynamically updates content as users interact with the application [27]. This paradigm:

- Reduces server load by transferring rendering responsibilities to the client
- Provides a more responsive user experience with minimized page reloads
- Enables rich, desktop-like application behaviors in web browsers

Our freight forwarding system utilizes the SPA approach through React.js, allowing for a seamless user experience when navigating between different system modules like quotation management, shipment tracking, and landing page customization.

2.1.3 Data Management Concepts

2.1.3.1 Object-Relational Mapping (ORM)

ORM is a programming technique that connects object-oriented programming languages with relational databases by creating a "virtual object database". This approach:

- Abstracts database interactions through programming language objects
- Reduces boilerplate SQL code and potential for SQL injection
- Facilitates database schema evolution alongside application development

In our implementation, we utilize JPA/Hibernate for the backend, enabling efficient database operations while maintaining a clean, object-oriented code structure.

2.1.3.2 Database Design for Multi-tenancy

Implementing multi-tenancy at the database level requires careful consideration of data isolation, performance, and maintenance. Our system employs PostgreSQL schema-based separation, which offers:

- Logical separation of tenant data without requiring separate database instances
- Efficient resource utilization compared to database-per-tenant approaches
- Simplified backup and restoration procedures
- Reduced operational overhead while maintaining security boundaries

This approach aligns with our requirements for data isolation while providing cost-effective scaling as our user base grows.

2.1.4 User Experience Concepts

2.1.4.1 Responsive Design

Responsive design ensures that web applications render effectively across various devices and screen sizes[25]. For a freight forwarding system accessed by users on diverse devices, this approach:

- Adapts layout and content based on viewport dimensions
- Ensures usability from desktop workstations to mobile devices
- Maintains consistent branding and user experience across platforms

Our implementation leverages the responsive capabilities of modern CSS frameworks to deliver an adaptive experience optimized for various usage contexts.

2.1.4.2 Progressive Enhancement

Progressive enhancement is a design philosophy that emphasizes core content and functionality first, then progressively adds more complex layers of presentation and features based on browser capabilities[29]. This approach:

- Ensures basic functionality remains accessible even in limited environments
- Delivers enhanced experiences to users with modern browsers
- Supports the diverse technological landscape of global freight operations

By adopting progressive enhancement principles, our system remains functional across the varied technical environments common in the logistics industry while delivering optimal experiences where possible.

2.2 Foundational Technologies

2.2.1 Programming Languages and Web Technologies

2.2.1.1 Java

Introduction: Java[16] is a class-based, object-oriented programming language designed to have minimal implementation dependencies. It follows the "write once, run anywhere" (WORA) principle, allowing compiled Java code to run on all platforms that support Java without recompilation. **Purpose of use:** In our project, Java serves as the primary backend programming language, providing a robust foundation for implementing business logic, data processing, and integration with external systems. Its enterprise-grade capabilities support the complex operations required in freight management.

Advantages:

- Platform independence through the Java Virtual Machine

- Strong typing system that reduces runtime errors
- Comprehensive standard libraries and frameworks
- Robust security features for handling sensitive logistics data
- Excellent documentation and large developer community

Disadvantages:

- Relatively verbose syntax compared to more modern languages
- Higher memory consumption than some alternatives
- Longer startup times that can impact development speed
- Steeper learning curve for beginners

Ability to meet project requirements:

- Provides the reliability needed for critical shipping operations
- Supports complex business rules implementation for freight forwarding
- Enables secure handling of sensitive customer and shipment data
- Integrates well with enterprise systems common in logistics

2.2.1.2 HTML/CSS/JavaScript

Introduction: HTML (HyperText Markup Language)[14], CSS (Cascading Style Sheets)[15], and JavaScript[17] form the core technologies for web development. HTML provides structure, CSS handles presentation, and JavaScript enables interactivity and dynamic content.

Purpose of use: For our freight forwarding system, these technologies create the foundation of our user interface, enabling us to build an accessible, responsive web application that works across various devices used in logistics operations.

Advantages:

- Universal browser support ensures wide accessibility
- Separation of concerns improves maintainability
- Responsive design capabilities adapt to different screen sizes
- Rich interactive capabilities through JavaScript

Disadvantages:

- Browser compatibility issues can increase development complexity
- CSS management becomes challenging as applications scale
- JavaScript without type checking can lead to runtime errors
- Performance optimization requires careful implementation

Ability to meet project requirements:

- Enables creation of interfaces for various logistics workflows
- Supports responsive design for field and office use cases
- Provides the interactivity needed for shipment tracking and status updates
- Allows progressive enhancement for varying connectivity scenarios

2.2.1.3 TypeScript

Introduction: TypeScript[46] is a strongly typed programming language that builds on JavaScript by adding static type definitions. It compiles to plain JavaScript but provides better tooling and error catching during development.

Purpose of use: In our system, TypeScript enhances JavaScript development by providing type safety and better code organization, which is critical for the complex data structures involved in freight management.

Advantages:

- Static typing catches errors during development
- Enhanced IDE support with better autocompletion
- Improved code documentation through type definitions
- Easier refactoring and maintenance for large codebases

Disadvantages:

- Additional compilation step compared to plain JavaScript
- Learning curve for developers unfamiliar with static typing
- Type definitions can add verbosity to code

- Integration with some third-party libraries may require additional type definitions

Ability to meet project requirements:

- Ensures data integrity in complex shipping and rate calculations
- Improves maintainability for our logistics workflow implementations
- Enables safer refactoring as requirements evolve
- Provides better documentation for the development team

2.2.2 Development Frameworks

2.2.2.1 Spring Boot

Introduction: Spring Boot[45] is an extension of the Spring framework that simplifies Java application development through convention-over-configuration. It provides a set of pre-configured templates and tools to accelerate development.

Purpose of use: For our freight forwarding system, Spring Boot provides the backend framework that handles API endpoints, business logic, and database operations with minimal boilerplate code.

Advantages:

- Rapid development through auto-configuration
- Built-in security features for protecting sensitive freight data
- Comprehensive testing support for business logic validation
- Production-ready monitoring and health checks

Disadvantages:

- Learning curve for developers new to the Spring ecosystem
- "Magic" configuration can be difficult to troubleshoot
- Potential overhead for simpler applications
- Occasional version compatibility issues with dependencies

Ability to meet project requirements:

- Provides secure, robust API endpoints for freight operations

- Supports complex business logic implementation for shipping processes
- Enables efficient database operations for logistics data
- Facilitates integration with external shipping and customs services

2.2.2.2 Node.js

Introduction: Node.js[12] is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser, using the V8 JavaScript engine.

Purpose of use: In our project, Node.js supports the frontend development environment and certain backend services, providing a JavaScript runtime for server-side execution and build tooling.

Advantages:

- Event-driven, non-blocking I/O model for efficiency
- Unified language across development stack reduces context switching
- Extensive package ecosystem through NPM
- Strong community support and frequent updates

Disadvantages:

- Single-threaded architecture can limit CPU-intensive tasks
- Callback patterns can lead to complex code structure
- Asynchronous programming model has a learning curve
- Evolving ecosystem can lead to dependency management challenges

Ability to meet project requirements:

- Provides efficient development tools for frontend implementation
- Supports asynchronous operations needed for shipping status updates
- Enables rapid API prototyping during development
- Integrates well with modern web development workflows

2.2.2.3 React.js

Introduction: React.js[42] is a JavaScript library for building user interfaces, focusing on component-based architecture and efficient rendering through a virtual DOM implementation.

Purpose of use: In our freight forwarding system, React provides the foundation for our user interface, enabling us to create reusable components that match logistics workflows and efficiently update based on data changes.

Advantages:

- Component-based architecture promotes reusability
- Virtual DOM ensures efficient UI updates
- Unidirectional data flow simplifies state management
- Rich ecosystem with many supporting libraries

Disadvantages:

- Focuses only on UI layer, requiring additional libraries for routing and state management
- Learning curve for concepts like JSX and component lifecycle
- Regular API changes can require adaptation
- Bundle size management requires attention

Ability to meet project requirements:

- Enables creation of complex, interactive shipping interfaces
- Supports efficient updates for real-time tracking features
- Allows modular development matching freight forwarding workflows
- Provides performance optimizations for data-heavy logistics screens

2.2.2.4 Next.js

Introduction: Next.js[30] is a React framework that provides server-side rendering, static site generation, and additional features like routing and API endpoints to enhance React applications.

Purpose of use: For our system, Next.js extends React's capabilities with improved performance, simplified routing, and server-side rendering for better initial page loads.

Advantages:

- Server-side rendering improves initial load performance and SEO
- Built-in routing system simplifies navigation implementation
- Automatic code splitting reduces bundle sizes
- API routes enable backend functionality within the same project

Disadvantages:

- More complex deployment than pure client-side React
- Additional build step can slow development iteration
- Some React libraries require adaptation for Next.js
- Server-side rendering adds complexity to component design

Ability to meet project requirements:

- Improves load performance for logistics dashboards with large datasets
- Simplifies navigation between different freight management sections
- Enhances SEO for public-facing freight service pages
- Provides consistent structure for large-scale application development

2.2.2.5 Ant Design

Introduction: Ant Design[18] is a design system and React UI library that provides a comprehensive set of high-quality components for building enterprise applications.

Purpose of use: In our freight forwarding system, Ant Design provides pre-built UI components that follow enterprise design patterns, accelerating development and ensuring consistency.

Advantages:

- Enterprise-focused components suitable for logistics applications
- Comprehensive component set reduces development time
- Consistent design language throughout the application
- Responsive layouts adapt to different device sizes

Disadvantages:

- Distinctive visual style may require customization
- Large component library can increase initial bundle size
- Some advanced components have steep learning curves
- Customization can become complex for highly specific needs

Ability to meet project requirements:

- Provides data tables needed for shipping manifests and rate sheets
- Offers form components for complex logistics data entry
- Includes visualization tools for shipment tracking and analytics
- Supports professional presentation required for enterprise logistics

2.2.3 Database and Storage Technologies

2.2.3.1 PostgreSQL

Introduction: PostgreSQL[39] is an advanced, open-source object-relational database system with a strong reputation for reliability, feature robustness, and performance.

Purpose of use: As our primary database, PostgreSQL stores and manages all freight forwarding data including shipments, clients, rates, and documentation.

Advantages:

- Strong support for complex data types and relationships
- Advanced indexing and query optimization
- Robust transaction support ensures data integrity
- Schema-based multitenancy capabilities

Disadvantages:

- More complex setup and administration than some alternatives
- Higher resource requirements than lightweight databases
- Horizontal scaling requires additional configuration
- Learning curve for advanced features

Ability to meet project requirements:

- Handles complex relationships between shipments, routes, and documentation
- Provides transaction support for critical freight operations
- Supports geographic data types for shipping routes and locations
- Offers schema separation for multi-tenant freight forwarding companies

2.2.3.2 Redis

Introduction: Redis[40] is an open-source, in-memory data structure store that can be used as a database, cache, and message broker. It supports various data structures and offers optional persistence.

Purpose of use: In our system, Redis provides high-performance caching for frequently accessed data, session management, and temporary storage needs.

Advantages:

- Exceptional performance for data retrieval operations
- Support for various data structures beyond simple key-value
- Built-in expiration policies for cache management
- Pub/sub capabilities for real-time features

Disadvantages:

- In-memory nature limits storage capacity
- Potential data loss during crashes if persistence isn't configured
- More complex clustering compared to some alternatives
- Requires careful memory management

Ability to meet project requirements:

- Accelerates access to frequently used shipping rates and location data
- Provides session storage for user authentication
- Supports real-time notifications for shipment status changes
- Enables fast search suggestions for logistics entities

2.2.4 Security and Authentication

2.2.4.1 JSON Web Tokens (JWT)

Introduction: JWT[24] is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. The information can be verified and trusted because it is digitally signed.

Purpose of use: Our system uses JWT for authentication and authorization, securing API endpoints and maintaining user sessions without server-side state.

Advantages:

- Stateless authentication reduces server storage requirements
- Compact format for efficient transmission
- Contains user claims and roles for authorization
- Works well across services and domains

Disadvantages:

- Tokens cannot be revoked before expiration
- Security depends on proper key management
- Token size increases with more claims
- Requires secure storage on client side

Ability to meet project requirements:

- Provides secure access to freight management features
- Supports role-based permissions for staff, admin, and client accounts
- Enables stateless authentication for scalable deployment
- Facilitates secure API access for third-party integrations

2.2.5 Infrastructure and Deployment

2.2.5.1 Docker

Introduction: Docker[6] is a platform that uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries, and configuration files.

Purpose of use: For our freight forwarding system, Docker provides consistent development and deployment environments, simplifying the transition between stages and ensuring reliable operation.

Advantages:

- Environment consistency eliminates configuration discrepancies
- Isolation improves security and simplifies dependency management
- Resource efficiency compared to traditional virtualization
- Simplified deployment and scaling procedures

Disadvantages:

- Learning curve for developers new to containerization
- Potential performance overhead in some scenarios
- Storage management for images and volumes
- Additional complexity in networking configuration

Ability to meet project requirements:

- Ensures consistent behavior across development and production
- Simplifies deployment of complex freight forwarding components
- Provides isolation for multi-tenant configurations
- Enables efficient resource utilization for cost-effective hosting

2.2.6 Development and Collaboration Tools

2.2.6.1 GitHub

Introduction: GitHub[23] is a web-based platform for version control and collaboration that uses Git as its core technology. It provides hosting for software development and offers distributed version control functionality.

Purpose of use: For our project, GitHub manages source code, facilitates collaboration through code reviews, and tracks issues throughout the development lifecycle.

Advantages:

- Comprehensive version control with branching and merging
- Pull request workflow supports code quality
- Issue tracking integrates with development activities
- Documentation capabilities through wikis and README files

Disadvantages:

- Learning curve for team members new to Git
- Potential for merge conflicts in active development
- Limited access control options in free tier
- Requires internet connectivity for most operations

Ability to meet project requirements:

- Supports collaborative development across team members
- Provides version history for complex freight system components
- Enables code review processes to maintain quality
- Tracks features and bugs throughout the development lifecycle

2.2.6.2 Jira

Introduction: Jira[44] is a proprietary issue tracking product developed by Atlassian that allows bug tracking, agile project management, and custom workflow creation.

Purpose of use: In our development process, Jira organizes tasks, tracks progress, and facilitates agile project management for our freight forwarding system.

Advantages:

- Flexible workflow customization for different development processes
- Comprehensive reporting and visualization tools
- Integration with development tools for traceability
- Supports agile methodologies like Scrum and Kanban

Disadvantages:

- Complex interface can be overwhelming for new users
- Requires consistent maintenance to remain effective
- Configuration complexity for custom workflows
- Potential for over-formalization of processes

Ability to meet project requirements:

- Provides structured tracking of freight system features
- Enables clear visualization of development progress
- Supports prioritization of critical logistics functionality
- Facilitates efficient task assignment and tracking

Chapter 3

System Analysis

3.1 Research on existing solutions

3.1.1 Freightify

Freightify[13] is a SaaS platform designed for freight forwarders to streamline their operations through rapid quote generation, digital storefront creation, and real-time vessel tracking capabilities, enabling forwarders to digitize their services and enhance customer experience through automated processes.

Key features of Freightify include:

- **Rate Procurement:** Centralized management of carrier rates from multiple sources
- **Instant Quote Generation:** Automated calculation of shipping costs based on cargo details
- **Track & Trace:** Real-time shipment tracking capabilities
- **Customer Portal:** Self-service options for clients to request quotes and track shipments
- **API Integration:** Connectivity with existing systems and third-party platforms

3.1.2 Phuong Nam Logistics

Phuong Nam Logistics Company is a comprehensive local logistics service provider that delivers a diverse portfolio of services, encompassing sea freight, air freight, and land transportation, complemented by value-added services including packaging, handling, customs clearance, and warehousing solutions.

The company maintains established strategic partnerships with multiple shipping lines and airlines. Leveraging its experienced professional team, Phuong Nam Logistics consistently demonstrates commitment to understanding client and partner requirements, delivering optimized solutions aligned with their operational principle: "Efficient - Dedicated - Optimal."

Key operational features of Phuong Nam Logistics' digital platform include:

- **Global Freight Transportation:** Strategic partnerships with reputable carriers ensure secure and timely cargo delivery across international networks
- **Warehousing Solutions:** Provision of modern storage facilities and inventory management systems tailored to meet diverse client requirements
- **Shipment Visibility:** Implementation of real-time online tracking systems enabling comprehensive shipment status monitoring and transparency
- **Documentation Management:** Handling of customs clearance, packaging, and other paperwork requirements
- **Multi-modal Transport Options:** Offering sea freight, air freight, and land transportation services

3.1.3 Cargologik

Cargologik is a SaaS solution that enables freight forwarders to streamline communication workflows and optimize operational efficiency through automated task management, centralized documentation, and simplified booking processes. The platform facilitates enhanced service delivery by allowing forwarders to shift focus from routine email management to core customer service requirements, while providing clients with streamlined access to critical shipment information and documentation.

Key features include:

- **Track and Trace:** With automated exception management and task queues
- **Communication Management:** Continuous, seamless communication across multiple shipments
- **Document Repository:** Critical document storage and management
- **Simplified Booking:** Streamlined quotation and booking processes

3.1.4 FreightFlex: Addressing Limitations of Existing Solutions

After analyzing the existing solutions in the freight forwarding management space, we identified several opportunities for enhancement that FreightFlex addresses:

- **Enhanced User Interface and Experience:** While existing solutions offer functional interfaces, FreightFlex prioritizes intuitive design with modern UI/UX principles, reducing the learning curve for users across different technical backgrounds.
- **Customizable Landing Page Builder:** Unlike most existing solutions that offer limited branding options, FreightFlex provides a comprehensive landing page builder that enables freight forwarders to create professional, branded online presence without requiring technical expertise or additional web development costs.
- **Advanced Multi-tenant Architecture:** FreightFlex implements a robust schema-based multi-tenant architecture that ensures superior data isolation between companies while maintaining cost efficiency, allowing freight forwarding businesses of all sizes to benefit from enterprise-grade security.
- **Granular Permission System:** FreightFlex extends beyond basic role-based access control with a fine-grained permission system allowing administrators to precisely define staff roles and access levels, creating custom permission sets tailored to their specific organizational structure.

By addressing these limitations, FreightFlex aims to provide a comprehensive solution that enhances operational efficiency, improves customer experience, and enables freight forwarding companies to maintain competitiveness in an increasingly digital marketplace.

3.2 Introduction to the System

FreightFlex is a web-based system that helps freight forwarding companies manage their business online. The system uses a multi-tenant approach, which means multiple companies can use the same platform while keeping their data completely separate from each other.

When a freight forwarding company subscribes to FreightFlex, they receive their own workspace with a unique web address. This workspace acts as their digital office where they can:

- Create their company website landing page
- Display their shipping services

- Provide automatic price quotes to customers
- Perform all essential operations typical of a freight forwarding application

3.3 Actors and Roles

3.3.1 Actors of the system

- **Freight Forwarder:** The logistics company that arranges international cargo transportation. They manage shipping documentation, coordinate with carriers, and ensure smooth delivery from origin to destination.
- **Clients:** Businesses or individuals who need to ship cargo internationally. They use the freight forwarder's services to transport their goods and track shipment progress.
- **Service Providers:** Companies that provide transportation and related services, including:
  - Shipping lines
  - Airlines
  - Trucking companies
  - Warehouse operators
- **System Administrator:** Technical staff who maintain the platform, manage user access, and provide support to ensure the system runs efficiently.
- **Regulatory Authorities:** Government bodies that oversee international trade and transportation, setting rules for customs, documentation, and safety standards.

3.3.2 Roles of the system

- **Super Admin:** Each company will be provided with a Super Admin account, which has the following capabilities:
  - Manage Users: Create or edit Client and Staff accounts, and assign permissions to them.
  - Create Admin Accounts: Create Admin accounts, which have the ability to create or edit Client and Staff accounts.
  - Exclusive Deletion Rights: Only the Super Admin account has the authority to delete Client, Staff, or Admin accounts.

- **Staff:** Staff account can be created by Super Admin or Admin. Their capabilities depend on the assigned roles and include:
  - Sales Staff: Request customers to confirm orders, place external orders with freight forwarders or brokers, and coordinate with carriers for returns and management of electronic shipping files.
  - Pricing Staff: Contact suppliers such as shipping lines or trucking companies to receive quotes for the sales team.
  - Logistics Operations Staff: Responsible for managing and coordinating issues related to the transportation of goods by road, such as trucks, container trucks, loading areas, loading/unloading goods, etc., or maritime issues, such as ship docking and departures.
  - Import/Export Documentation Manager: Fully responsible for all documents and paperwork related to the import/export activities of businesses specializing in this field. For example: arrival notices, packing lists, contracts, invoices, bills of lading, etc. Main areas of work include: export/import sea freight documentation, customs declaration documents, customs procedures, freight – logistics documentation, and international payment documents.
  - Accounting Staff: Manage costs and revenues, and prepare company reports.
- **Client:** Client account can be created by Super Admin or Admin, which has the following capabilities:
  - Create Quote Requests
  - View Rates
  - Track Shipment status in real-time

3.4 Usecase Diagram

3.4.1 Usecase Diagram for Admin

Figure 3.4.1: Usecase Diagram for Admin

3.4.1.1 Landing Page Settings

Usecase name	Custom Landing Page
Actor	Admin
Description	Custom Landing Page
Preconditions	1. Authorized with Super Admin Account. 2. Actor is in Landing Page.
Postconditions	Notification success or failed.
Normal Flow	1. Actor turns on edit mode. 2. Actor upload Company logo. 3. Actor choose color theme for each section. 4. Actor choose charts to show. 5. Actor drag and drop each section to suitable position. 6. Actor preview changes.
Exception Flow	5a. Section size is not fit to desired position
Alternative Flow	

Table 3.4.1: Custom Landing Page

3.4.1.2 System Settings

Usecase name	Setup Company Profile
Actor	Admin
Description	Setup Company Profile
Preconditions	1. Authorized with Super Admin Account.
Postconditions	Notification success or failed.
Normal Flow	1. Actor go to Settings. 2. Actor choose tab Company Profile. 3. Actor upload Company logo. 4. Actor fills required fields. 5. Actor click on save button.
Exception Flow	3a. Wrong file format. 4a. Miss required fields. 4b. Wrong field format.
Alternative Flow	4c. Actor changes or removes data.

Table 3.4.2: Company Profile Settings

Usecase name	Create role
Actor	Admin
Description	Add role with permissions
Preconditions	Authorized with Super Admin Account.
Postconditions	Show role details.
Normal Flow	1. Actor go to Settings. 2. Actor choose tab Role. 3. Actor choose add new button. 4. Actor ticks the checkbox for permission 5. Actor click on save button.
Exception Flow	
Alternative Flow	

Table 3.4.3: Create new role

3.4.1.3 Account Settings

Usecase name	Add staff account
Actor	Admin
Description	Add staff account
Preconditions	Authorized with an Admin Privilege account.
Postconditions	Add account successful
Normal Flow	1. Actor go to Adminstrantions. 2. Actor choose tab Staff. 3. Actor choose create button. 4. Actor fills required fields. 5. Actor click on save button.
Exception Flow	4a. Actor does not fill required fields. Cannot save staff.
Alternative Flow	

Table 3.4.4: Add new staff

Usecase name	Grant permissions for staff account
Actor	Admin
Description	Grant permissions for staff account
Preconditions	Authorized with an Admin Privilege account.
Postconditions	Update account successful
Normal Flow	1. Actor go to Settings. 2. Actor choose tab Role. 3. Actor choose role to edit. 4. Actor add account to role. 5. Actor click on save button.
Exception Flow	
Alternative Flow	

Table 3.4.5: Grant permissions for staff

3.4.1.4 Agent Settings

Usecase name	View agent details
Actor	Admin, Staff
Description	View agent details
Preconditions	Staff has view permission or Admin
Postconditions	Display a detailed agent including rates
Normal Flow	1. Actor go to Providers. 2. Actor choose tab Agent. 3. Actor choose Agent to view.
Exception Flow	Actor does not have permission to view agent
Alternative Flow	

Table 3.4.6: View agent details

3.4. Usecase Diagram

Usecase name	Search agent
Actor	Admin, Staff
Description	Search agent
Preconditions	Staff has view permission or Admin
Postconditions	Display a paginated list of agents related to the search information.
Normal Flow	1. Actor go to Providers. 2. Actor choose tab Agent. 3. Actor enters information to search. 4. Actor click search button.
Exception Flow	Actor does not have permission to view agent
Alternative Flow	

Table 3.4.7: Search agent

Usecase name	Import agents using excel file
Actor	Admin, Staff
Description	Import agents using excel file
Preconditions	Staff has add agent permission or Admin
Postconditions	Display agent list after imported
Normal Flow	1. Actor go to Providers. 2. Actor choose tab Agent. 3. Actor choose Upload excel file. 4. Actor preview agent list imported from excel. 5. Actor confirm to import agents.
Exception Flow	4a. Actor does not have permission to add agents 4b. Wrong data format.
Alternative Flow	5a. Actor cancel to import

Table 3.4.8: Import agent from excel

Usecase name	Add single rate
Actor	Admin, Staff
Description	Add rate to the agent
Preconditions	Staff has add rate permission or Admin
Postconditions	Display rate in agent detail
Normal Flow	1. Actor go to Rates. 2. Actor choose tab Freight or Local Charges or Other Services. 3. Actor clicks new button. 4. Actor choose a provider. 5. Actor fills required fields. 6. Actor clicks save button.
Exception Flow	3a. Actor does not have permission to edit agent 5b. Actor does not fill required fields. Cannot save.
Alternative Flow	

Table 3.4.9: Add single rate

3.4.2 Usecase Diagram for Staff

graphics/usecase/freight-flex-UC-Staff.png

Figure 3.4.2: Usecase Diagram for Staff

3.4.2.1 Rate Settings

Usecase name	Unsubscribe rate change notification
Actor	Client, Staff
Description	Unsubscribe rate change notification
Preconditions	Client, Staff has view rates permission
Postconditions	Unsubscribe successful
Normal Flow	1. Actor go to Rates 2. Actor choose existing rate. 3. Actor choose unsubscribe button. 4. Actor choose confirm button.
Exception Flow	
Alternative Flow	

Table 3.4.10: Unsubscribe rate change notification

3.4.2.2 Quote Settings

Usecase name	View Quote Request list
Actor	Staff
Description	View all quote requests.
Preconditions	Staff has view quote permission
Postconditions	Display a paginated list of quote requests
Normal Flow	1. Actor go to Quotations. 2. Actor go to Quote Request tab.
Exception Flow	
Alternative Flow	

Table 3.4.11: View Quote Request list

3.4. Usecase Diagram

Usecase name	View Quote Request detail
Actor	Staff
Description	View quote request detail.
Preconditions	Staff has view quote permission
Postconditions	Display quote request detail
Normal Flow	1. Actor go to Quotations. 2. Actor go to Quote Request tab. 3. Actor choose a quote request.
Exception Flow	
Alternative Flow	

Table 3.4.12: View Quote Request detail

Usecase name	Change status of a quote request
Actor	Staff
Description	Change status of a quote request
Preconditions	Staff has view and edit quote permission
Postconditions	Update status successful
Normal Flow	1. Actor go to Quotations. 2. Actor go to Quote Request tab. 3. Actor choose a quote request. 4. Actor choose status 5. Actor choose update
Exception Flow	5a. Status changed failed because status must be changed level one by one.
Alternative Flow	

Table 3.4.13: Change quote request status

Usecase name	Create quote from request
Actor	Staff
Description	Generate a quote from the quote request
Preconditions	Staff has view, create and edit quote permission
Postconditions	Create quote successful
Normal Flow	1. Actor go to Quotations. 2. Actor go to Quote Request tab. 3. Actor choose a quote request. 4. Actor choose create quote from request button. 5. Actor choose confirm.
Exception Flow	
Alternative Flow	

Table 3.4.14: Create quote from request

3.4.2.3 Shipment Settings

Usecase name	Add shipment
Actor	Staff
Description	Add a shipment
Preconditions	1. Staff has create shipment permission. 2. A quote is created.
Postconditions	Create shipment successful
Normal Flow	1. Actor go to Quotations. 2. Actor go to Quote tab. 3. Actor choose a quote not booked. 4. Actor choose create shipment button. 5. Actor update shipment information. 6. Actor choose create.
Exception Flow	
Alternative Flow	

Table 3.4.15: Create shipment from quote

3.4.3 Usecase Diagram for Client

Figure 3.4.3: Usecase Diagram for Client

Usecase name	View shipment status
Actor	Client
Description	View shipment status
Preconditions	1. Clien has view shipment permission.
Postconditions	Display shipment status detail
Normal Flow	1. Actor go to Shipments. 2. Actor choose a shipment.
Exception Flow	
Alternative Flow	

Table 3.4.16: View shipment

Usecase name	Seach shipment
Actor	Client
Description	View a paginated list of shipments related to the search information
Preconditions	1. Clien has view shipment permission.
Postconditions	Display a paginated list of shipments related to the search information
Normal Flow	1. Actor go to Shipments. 2. Actor clicks to Search. 3. Actor enters information to search. 4. Actor clicks search button
Exception Flow	Search keyword does not match any shipments
Alternative Flow	

Table 3.4.17: Seach shipment

Chapter 4

System Design

4.1 Database Design

4.1.1 Enhanced ER Diagram.

Figure 4.1.1: EERD

4.1.2 Entity Detail

4.1.2.1 Action

Attribute	Type	Description
id	UUID	Primary key
name	String	Action name

Table 4.1.1: Table of Action

4.1.2.2 Resource

Attribute	Type	Description
id	UUID	Primary key
name	String	Resource name
parent_id	UUID	Foreign key to Resource

Table 4.1.2: Table of Resource

4.1.2.3 Resource Action

Attribute	Type	Description
id	Integer	Primary key
resource_id	UUID	Foreign key to Resource
action_id	UUID	Foreign key to Action
description	String	Description of the resource action

Table 4.1.3: Table of Resource Action

4.1.2.4 Staff

Attribute	Type	Description
id	Integer	Primary key
code	String	Staff code
email	String	Email for login
password	String	Password for login
firstname	String	First name
lastname	String	Last name
phone	String	Phone number
last_login_time	Timestamp	Last login time
created_at	Timestamp	The time when the Staff is created
updated_at	Timestamp	The time when the Staff is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.4: Table of Staff

4.1.2.5 Staff Group

Attribute	Type	Description
id	UUID	Primary key
name	String	Group name
description	String	Group description
created_at	Timestamp	The time when the Group is created
updated_at	Timestamp	The time when the Group is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.5: Table of Staff Group

4.1.2.6 Staff Group Staff

Attribute	Type	Description
id	UUID	Primary key
staff_group_id	UUID	Foreign key to Staff Group
staff_id	Integer	Foreign key to Staff
joined_date	Timestamp	The time when the Staff joined the group
added_by	Integer	Foreign key to Staff

Table 4.1.6: Table of Staff Group Staff

4.1.2.7 Role

Attribute	Type	Description
id	UUID	Primary key
name	String	Role name
description	String	Role description
type	String	Role type
created_at	Timestamp	The time when the Role is created
updated_at	Timestamp	The time when the Role is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.7: Table of Role

4.1.2.8 Staff Group Role

Attribute	Type	Description
id	Integer	Primary key
group_id	UUID	Foreign key to Staff Group
role_id	UUID	Foreign key to Role

Table 4.1.8: Table of Staff Group Role

4.1.2.9 Staff Role

Attribute	Type	Description
id	Integer	Primary key
staff_id	Integer	Foreign key to Staff
role_id	UUID	Foreign key to Role

Table 4.1.9: Table of Staff Role

4.1.2.10 Permission

Attribute	Type	Description
id	UUID	Primary key
name	String	Permission name
description	String	Permission description
scope	Integer	Permission scope
created_at	Timestamp	The time when the Permission is created
updated_at	Timestamp	The time when the Permission is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.10: Table of Permission

4.1.2.11 Permission Resource Action

Attribute	Type	Description
id	UUID	Primary key
permission_id	UUID	Foreign key to Permission
resource_action_id	Integer	Foreign key to Resource Action

Table 4.1.11: Table of Permission Resource Action

4.1.2.12 Permission Role

Attribute	Type	Description
id	Integer	Primary key
permission_id	UUID	Foreign key to Permission
role_id	UUID	Foreign key to Role

Table 4.1.12: Table of Permission Role

4.1.2.13 Company Info

Attribute	Type	Description
id	Integer	Primary key
name	String	Company name
address	String	Company address
city	String	Company city
country	String	Company country
email	String	Company email
zip_code	String	Company zipcode
phone	String	Company phone number
tax_number	String	Company tax number
is_domestic	Boolean	Is domestic

Table 4.1.13: Table of Company Info

4.1.2.14 Contact Person Info

Attribute	Type	Description
id	Integer	Primary key
first_name	String	Contact person first name
last_name	String	Contact person last name
email	String	Contact person email
phone	String	Contact person phone number
position	String	Contact person position/title

Table 4.1.14: Table of Contact Person Info

4.1.2.15 Client

Attribute	Type	Description
id	Integer	Primary key
code	String	Client code
name	String	Client name
email	String	Client email
company_info_id	Integer	Foreign key to Company Info
contact_person_info_id	Integer	Foreign key to Contact Person Info
staff_id	Integer	Foreign key to Staff

Table 4.1.15: Table of Client

4.1.2.16 Provider

Attribute	Type	Description
id	Integer	Primary key
code	String	Provider code
name	String	Provider name
type	String	Provider type
tracking_url	String	URL for tracking shipments
company_info_id	Integer	Foreign key to Company Info
contact_person_info_id	Integer	Foreign key to Contact Person Info

Table 4.1.16: Table of Provider

4.1.2.17 UnLoCo (United Nations Location Code)

Attribute	Type	Description
id	Integer	Primary key
code	String	Location code
country_code	String	Country code
country_name	String	Country name
city_code	String	City code
city_name	String	City name

Table 4.1.17: Table of UnLoCo

4.1.2.18 Currency

Attribute	Type	Description
id	Integer	Primary key
code	String	Currency code
name	String	Currency name
symbol	String	Currency symbol
created_at	Timestamp	The time when the Currency is created
updated_at	Timestamp	The time when the Currency is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.18: Table of Currency

4.1.2.19 Charge Type

Attribute	Type	Description
id	Integer	Primary key
name	String	Charge type name
description	String	Charge type description
calculation_type	String	Type of calculation for charges
created_at	Timestamp	The time when the Charge Type is created
updated_at	Timestamp	The time when the Charge Type is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.19: Table of Charge Type

4.1.2.20 Service Charge

Attribute	Type	Description
id	Integer	Primary key
charge_type_id	Integer	Foreign key to Charge Type
price	Numeric	Price amount
currency_id	Integer	Foreign key to Currency
transport_type	Integer	Type of transport
valid_from	Date	Start date of validity
valid_to	Date	End date of validity
created_at	Timestamp	The time when the Service Charge is created
updated_at	Timestamp	The time when the Service Charge is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.20: Table of Service Charge

4.1.2.21 FCL Rate

Attribute	Type	Description
id	Integer	Primary key
provider_id	Integer	Foreign key to Provider
origin_id	Integer	Foreign key to Origin UnLoCo
destination_id	Integer	Foreign key to Destination UnLoCo
currency_id	Integer	Foreign key to Currency
kind	String	Kind of FCL rate
transport_type	String	Type of transport
commodity	String	Type of goods
transit_time	Integer	Transit time in days
remark	String	Additional remarks
valid_from	Date	Start date of validity
valid_to	Date	End date of validity
is_from_quote	Boolean	Whether this rate is from a quote
rate_20dc	Integer	Rate for 20ft dry container
rate_40dc	Integer	Rate for 40ft dry container
rate_20hc	Integer	Rate for 20ft high cube container
rate_40hc	Integer	Rate for 40ft high cube container
rate_20rf	Integer	Rate for 20ft reefer container
rate_40rf	Integer	Rate for 40ft reefer container
rate_45hc	Integer	Rate for 45ft high cube container
created_at	Timestamp	The time when the FCL Rate is created
updated_at	Timestamp	The time when the FCL Rate is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.21: Table of FCL Rate

4.1.2.22 Quote

Attribute	Type	Description
id	Integer	Primary key
client_id	Integer	Foreign key to Client
provider_id	Integer	Foreign key to Provider
origin_id	Integer	Foreign key to Origin UnLoCo
destination_id	Integer	Foreign key to Destination UnLoCo
cargo_volume_id	Integer	Foreign key to Cargo Volume
shipment_type	String	Type of shipment
shipment_mode	String	Mode of shipment
transport_type	String	Type of transport
incoterm	String	International commercial terms
is_request	Boolean	Whether this is a quote request
status	String	Status of the quote
note	String	Additional notes
valid_until	Date	Quote validity end date
created_at	Timestamp	The time when the Quote is created
updated_at	Timestamp	The time when the Quote is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.22: Table of Quote

4.1.2.23 Quote Price Detail

Attribute	Type	Description
id	Integer	Primary key
quote_id	Integer	Foreign key to Quote
provider_id	Integer	Foreign key to Provider
charge_type_id	Integer	Foreign key to Charge Type
description	String	Description of the price detail
base_price	Integer	Base price amount
quantity	Integer	Quantity
total_price	Integer	Total price amount
currency_id	Integer	Foreign key to Currency
created_at	Timestamp	The time when the Quote Price Detail is created
updated_at	Timestamp	The time when the Quote Price Detail is updated
created_by	Integer	Foreign key to Staff
updated_by	Integer	Foreign key to Staff

Table 4.1.23: Table of Quote Price Detail

4.2 System Architecture

graphics/sys-design/system-architecture.png

Figure 4.2.1: FreightFlex Overall Architecture

4.2. System Architecture

After careful analysis of constraints such as human resources, development timelines, and maintainability requirements, the FreightFlex system adopts a client-server architectural pattern combined with an n-layer architecture for its backend. This decision balances system complexity with development efficiency while ensuring scalability and maintainability.

The frontend utilizes `React.js` with `Vite` for development, creating a robust client-side application that manages user interface rendering and state. This modern framework enables dynamic, responsive interfaces and high performance through virtual DOM implementation. `Vite` enhances the development experience with features like hot module replacement, reducing iteration cycles.

The backend is built with the `Spring Framework` in Java, structured in a layered architecture that promotes separation of concerns. This architecture comprises:

- Presentation Layer (REST API controllers)
- Business Logic Layer (services)
- Data Access Layer (repositories)
- Domain Layer (entities and value objects)

The database employs a multi-tenant design using PostgreSQL's schema-based separation, which efficiently handles multiple freight forwarding companies (tenants) within a single database instance while maintaining strict data isolation. Key advantages include:

- **Data Isolation:**
  - Each tenant's data resides in its own schema.
  - Cross-tenant data access is prevented through schema-level security.
- **Resource Optimization:**
  - Shared database instance reduces infrastructure costs.
  - Efficient resource utilization via connection pooling.
- **Tenant Management:**
  - Dynamic tenant provisioning through automated schema creation.
  - Flexible tenant-specific customizations.
- **Performance Considerations:**
  - Optimized query performance through schema-level indexing.

  - Reduced database connection overhead.

A centralized tenant management system supports this multi-tenant implementation, handling tenant registration, schema management, and data isolation enforcement.

This architectural approach maintains simplicity in client-server communication and provides several advantages for the freight forwarding system:

- Clear separation of concerns for easier maintenance and enhancements.
- Scalable design for potential growth in functionality and user base.
- Secure multi-tenant implementation protecting sensitive data.
- Efficient resource utilization aligned with team capacity.

4.3 Tools and Technologies Used

4.3.1 Frontend Technologies

The frontend of the application is developed using the following technologies:

- **React.js:** A popular JavaScript library for building user interfaces, enabling the creation of dynamic and responsive web applications.
- **Vite:** A modern build tool that enhances the development experience with features such as fast hot module replacement and optimized production builds.
- **Ant Design:** A comprehensive design system and UI component library that provides pre-built components for building aesthetically pleasing and user-friendly interfaces.

4.3.2 Backend Technologies

The backend is constructed using the following technologies:

- **Java:** A robust, object-oriented programming language that serves as the foundation for backend development.
- **Spring Framework:** A powerful framework that simplifies Java application development, promoting good design practices through its modular architecture and dependency injection.
- **QueryDSL:** A framework that provides a type-safe way to construct SQL queries, enhancing the development process and maintaining code readability.

- **PostgreSQL:** An advanced, open-source relational database management system known for its reliability and performance, utilized for data storage and management.

4.3.3 Development Tools

The following tools are utilized to streamline the development process:

- **Postman:** A collaboration platform for API development that provides tools for designing, testing, and documenting APIs.
- **Docker:** A containerization platform that enables the creation, deployment, and management of applications in isolated environments, ensuring consistency across different environments.
- **GitHub:** A web-based version control platform that facilitates collaboration among developers, providing tools for code management and version control.
- **Jira:** A project management tool that helps in tracking issues, managing tasks, and facilitating agile project management processes.

Chapter 5

System Implementation

5.1 Technologies and Libraries Used

5.1.1 Frontend Technologies

Our frontend implementation utilizes several modern technologies and libraries to create a responsive, maintainable, and user-friendly interface:

- **Ant Design**: We implemented this comprehensive design system to provide high-quality React components for building elegant, consistent, and accessible user interfaces. Ant Design offered a robust set of UI elements that adhere to design principles and best practices, significantly reducing development time while maintaining visual consistency throughout the application.
- **Redux Toolkit**: This official, opinionated toolkit for efficient Redux development simplified our store setup, reduced boilerplate code, and provided utilities for common Redux patterns. Its standardized approach to state management helped us maintain a predictable state container across the entire application.
- **Redux Toolkit Query**: We leveraged this powerful data fetching and caching tool built into Redux Toolkit for automated data refetching, cache invalidation, and optimistic updates with minimal configuration. This significantly improved our API integration by providing a streamlined approach to data management.
- **React Router**: This standard routing library for React applications enabled navigation between different components, allowing for dynamic routing and declarative route definitions. It provided a consistent user experience while navigating through different sections of the application.

- **CKEditor**: For content creation functionality, we integrated this modern WYSIWYG rich text editor with features like image handling, formatting tools, and source code editing capabilities. This enhanced the user experience for creating and editing textual content within our system.
- **Next.js Components**: We incorporated select Next.js elements for improved rendering performance, enhanced SEO capabilities, and streamlined application development. These components helped optimize our application's performance and search engine visibility.

5.1.2 Backend Technologies

The backend of our system was built using robust Java-based technologies:

- **Spring Boot**: This Java-based framework simplified the development of our standalone, production-grade application by providing auto-configuration, embedded servers, and application metrics. Spring Boot's convention-over-configuration approach accelerated development while maintaining high standards for security and performance.
- **Java Persistence API (JPA)**: We implemented this Java specification for managing relational data as it provides a standard approach to Object-Relational Mapping (ORM), significantly reducing boilerplate database access code. JPA allowed us to work with database entities using object-oriented principles rather than SQL statements.
- **Java JWT**: This library for JSON Web Token implementation in Java provided secure mechanisms for data transmission and authentication between parties. We utilized it to implement stateless authentication, enhancing security while maintaining scalability.

These technologies were selected based on their robustness, community support, documentation quality, and alignment with our project requirements. The combination provided a solid foundation for developing a scalable, maintainable, and performant freight forwarding management system.

5.2 Source Code

5.2.1 Source Code Management

To manage our source code effectively, our project is organized using a modular approach, separating frontend and backend into distinct components. We use Git for version control, allowing the team to work concurrently on different features without conflicts.

5.2.2 Frontend Source Code Structure

The frontend source code is organized using a modular approach, where each feature or module can be developed independently. This not only improves maintainability but also simplifies future extensions.

The frontend project has the following main directory structure:

- **/apps** - Contains the main application
  - **/lcp-apps/src** - Main source code of the application
    - **/app** - Contains application initialization files
      - `app.tsx` - Main file defining the application structure
      - `styles.scss` - Default style definitions
      - `main.tsx` - Entry point of the application
- **/libs** - Shared code libraries
  - **/common** - Common code used throughout the application
  - **/components** - Reusable UI components
  - **/modules** - Functional modules of the application
    - **/auth** - Authentication management
    - **/clients** - Client management
    - **...**

Each functional module is organized with a consistent structure:

- **/assets** - Static assets like images and fonts
- **/hooks** - Custom React hooks for reusable logic
- **/components** - Module-specific UI components
- **/containers** - Higher-level components containing business logic
- **/i18n** - Internationalization files for the module
- **/pages** - Main pages within the module
- **/redux** - State management with Redux
- **/services** - API services and data handling

- **/types** - TypeScript type definitions

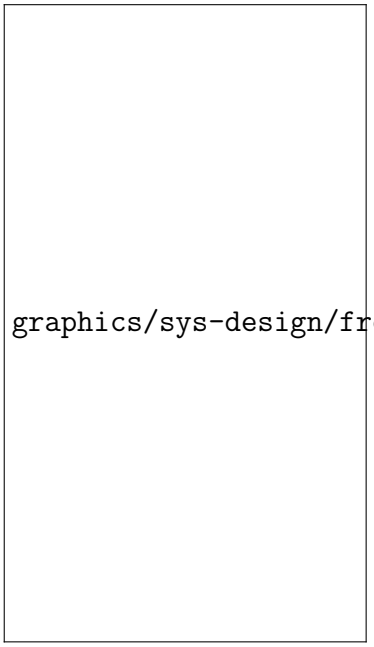


Figure 5.2.1: Frontend Source Code Structure

This structure allows us to develop each feature independently while maintaining code reusability and consistency throughout the project.

5.2.3 Backend Source Code Structure

The backend is developed using Java with Spring Boot, following a layered architecture pattern that promotes separation of concerns and maintainability. Our backend follows standard Java package naming conventions with a comprehensive structure organized by technical responsibility.

The main package structure reflects both cross-cutting concerns and feature modules:

- **com.lcp** - Root package that contains all application code
  - **base.dto** - Contains base data transfer object classes that establish inheritance hierarchies and common patterns for all DTOs in the system
  - **common** - Houses utilities, constants, and common components shared across the entire application
  - **configuration** - Contains Spring configuration classes, including bean definitions, property sources, and application profiles

  - **exception** - Centralizes exception handling with custom exception classes and global exception handlers
  - **security** - Implements authentication, authorization, and other security-related functionality
  - **util** - Provides general-purpose utility classes that support various operations throughout the application

Each business feature module follows a consistent layered architecture:

- **/controller** - Contains REST API endpoints that handle HTTP requests and responses. Controllers are responsible for request validation, authentication verification, and routing to appropriate service methods.
- **/dto** - Data Transfer Objects that define the structure of data exchanged between the client and server. These objects isolate the internal entity model from external API contracts.
- **/entity** - JPA entity classes that map directly to database tables. These classes include field definitions, relationships, constraints, and JPA annotations.
- **/mapper** - Contains object mapper classes that transform data between entities and DTOs, ensuring clean separation between persistence and API layers.
- **/repository** - Implements the data access layer using Spring Data JPA repositories. These interfaces provide methods for CRUD operations and custom queries against the database.
- **/service** - Houses the business logic implementation. Service classes orchestrate operations across multiple repositories, implement transaction boundaries, and enforce business rules.
- **/common** - Module-specific utilities, constants, and shared components that are relevant only within the scope of the particular module.

This layered architecture provides several benefits:

- Clear separation of concerns, making the codebase easier to understand and maintain
- Improved testability with well-defined boundaries between layers
- Consistent patterns across different modules that reduce cognitive load for developers
- Flexibility to modify implementation details in one layer without affecting others

- Enhanced security through proper encapsulation of business logic and data access

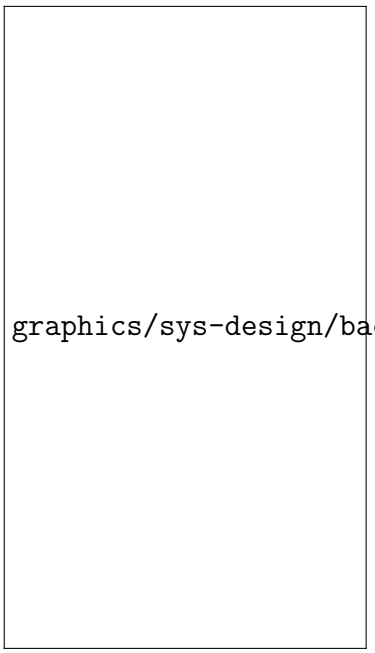


Figure 5.2.2: Typical Module Internal Structure

The architecture follows the dependency inversion principle, with controllers depending on services, which depend on repositories, creating a clean flow of control and data throughout the application.

5.3 Interfaces implementation

Chapter 6

Deployment, Testing and System Evaluation

6.1 System Testing

6.1.1 API Testing

6.1.2 Web Interface Testing

6.2 Deployment

6.2.1 Frontend Deployment

6.2.2 Backend Deployment

6.3 System Evaluation

Chapter 7

Summary, Evaluation and Development Direction

7.1 Summary

7.2 System Evaluation

7.3 Future Development Direction

References

[1] Cor-Paul Bezemer and Andy Zaidman. Multi-tenant saas applications: maintenance dream or nightmare? In *Proceedings of the Joint ERCIM Workshop on Software Evolution and International Workshop on Principles of Software Evolution*, pages 88–92, 2010. URL: https://dl.acm.org/doi/10.1145/1862372.1862393.

[2] Stephen J. Bigelow and Alexander S. Gillis. What is multi-tenancy (multi-tenant architecture)?, 2024. Accessed: 2024-12-13. URL: https://www.techtarget.com/whatis/definition/multi-tenancy.

[3] Frederick Chong, Gianpaolo Carraro, and Roger Wolter. Multi-tenant data architecture, 2006. URL: https://docs.microsoft.com/en-us/previous-versions/aa479086(v=msdn.10).

[4] European Commission. Digital transport and logistics forum, 2023. URL: https://transport.ec.europa.eu/transport-themes/digital-transport-and-logistics-forum-dtlf_en.

[5] World Wide Web Consortium. Web services architecture, 2022. URL: https://www.w3.org/standards/webdesign/.

[6] Docker. What is Docker?, n.d. Accessed: 2024-12-13. URL: https://docs.docker.com/get-started/docker-overview/.

[7] United Nations Economic, Social Commission for Asia, and the Pacific. Digital and sustainable trade facilitation report 2022, 2022. URL: https://www.unescap.org/kp/2022/untf-survey-2023.

[8] Roy Thomas Fielding. Architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California*, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[9] Organisation for Economic Co-operation and Development. The digital transformation of logistics, 2021. URL: https://www.oecd.org/trade/topics/trade-facilitation/.

[10] United Nations Economic Commission for Europe. Digital transformation for sustainable trade facilitation, 2021. URL: https://unece.org/trade/uncefact/mainstandards.

[11] United Nations Economic Commission for Europe. Trade facilitation and electronic business, 2023. URL: https://unece.org/trade/uncefact.

[12] Node.js Foundation. Introduction to node.js, n.d. Accessed: 2024-12-13. URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs.

[13] freightify. Powering Digital Transformation for Freight Forwarders, n.d. Accessed: 2024-12-13. URL: https://www.freightify.com/about-us.

[14] GeeksforGeeks. Html5 | Introduction, 2023. Accessed: 2024-12-13. URL: https://www.geeksforgeeks.org/html5-introduction/.

[15] GeeksforGeeks. Css Introduction, 2024. Accessed: 2024-12-13. URL: https://www.geeksforgeeks.org/css-introduction/.

[16] GeeksforGeeks. Introduction to Java, 2024. Accessed: 2024-12-13. URL: https://www.geeksforgeeks.org/introduction-to-java/.

[17] GeeksforGeeks. Introduction to Javascript, 2024. Accessed: 2024-12-13. URL: https://www.geeksforgeeks.org/introduction-to-javascript/.

[18] GeeksforGeeks. Ant Design, n.d. Accessed: 2024-12-13. URL: https://www.geeksforgeeks.org/ant-design/.

[19] Nielsen Norman Group. Ux guidelines for enterprise applications, 2023. URL: https://www.nngroup.com/articles/.

[20] World Bank Group. Logistics performance index: Connecting to compete 2022, 2022. URL: https://lpi.worldbank.org/.

[21] C. Harris. Microservices vs. monolithic architecture, n.d. Accessed: 2024-12-13. URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith.

[22] Christopher Ireland, David Bowers, Michael Newton, and Kevin Waugh. A classification of object-relational impedance mismatch. In *First International Conference on Advances in Databases, Knowledge, and Data Applications*, pages 36–43. IEEE, 2009. URL: https://ieeexplore.ieee.org/document/4782514.

[23] Jamie Juviler. What Is Github? (And What Is It Used For?), 2021. Accessed: 2024-12-13. URL: https://blog.hubspot.com/website/what-is-github-used-for.

[24] JWT. Introduction to json web tokens, n.a. Accessed: 2024-12-13. URL: https://jwt.io/introduction.

[25] Ethan Marcotte. *Responsive Web Design*. A Book Apart, 2011. URL: https://abookapart.com/products/responsive-web-design.

[26] Mark Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, 2011. URL: https://www.oreilly.com/library/view/rest-api-design/9781449317904/.

[27] Michael S. Mikowski and Josh C. Powell. *Single Page Web Applications: JavaScript End-to-end*. Manning Publications, 2013. URL: https://www.manning.com/books/single-page-web-applications.

[28] MuleSoft. What is an Api (Application Programming Interface)?, n.d. Accessed: 2024-12-13. URL: https://www.mulesoft.com/api/what-is-an-api.

[29] Mozilla Developer Network. Progressive enhancement, n.a. URL: https://developer.mozilla.org/en-US/docs/Glossary/Progressive_Enhancement.

[30] Next.js. Next.js, n.a. Accessed: 2024-12-13. URL: https://nextjs.org/.

[31] National Institute of Standards and Technology. Cloud computing security reference architecture, 2022. URL: https://csrc.nist.gov/publications/sp.

[32] International Labour Organization. The future of work in the transport sector, 2022. URL: https://www.ilo.org/global/topics/future-of-work/.

[33] International Maritime Organization. Facilitation of international maritime traffic, 2023. URL: https://www.imo.org/en/OurWork/Facilitation/.

[34] United Nations Industrial Development Organization. Supply chain management: Concepts and practice, 2022. URL: https://www.unido.org/resources.

[35] United Nations Industrial Development Organization. Inclusive and sustainable industrial development in the digital era, 2023. URL: https://www.unido.org/resources-publications-flagship-publications.

[36] World Customs Organization. Global trade facilitation, 2023. URL: http://www.wcoomd.org/.

[37] World Trade Organization. E-commerce and digital trade, 2021. URL: https://www.wto.org/english/tratop_e/ecom_e/ecom_e.htm.

[38] World Trade Organization. World trade report 2022: Climate change and international trade, 2022. URL: https://www.wto.org/english/res_e/publications_e/wtr22_e.htm.

[39] PostgreSQL. About, n.d. Accessed: 2024-12-13. URL: https://www.postgresql.org/about/.

[40] Redis. Redis, n.a. Accessed: 2024-12-13. URL: https://redis.io/docs/latest/.

[41] Chris Richardson. *Microservices Patterns: With Examples in Java*. Manning Publications, 2018. URL: https://www.manning.com/books/microservices-patterns.

[42] Sanity. React.js overview, 2024. Accessed: 2024-12-13. URL: https://www.sanity.io/glossary/react-js.

[43] Indeed Editorial Team. What is Freight Forwarding? Definition, Benefits and Key Stages, 2024. Accessed: 2024-12-13. URL: https://www.indeed.com/career-advice/career-development/freight-forwarder-meaning.

[44] Jira Team. Welcome to Jira, n.d. Accessed: 2024-12-13. URL: https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software.

[45] Spring Team. Spring boot, n.a. URL: https://spring.io/projects/spring-boot.

[46] Typescript. Typescript, n.a. Accessed: 2024-12-13. URL: https://www.typescriptlang.org/.

[47] International Telecommunication Union. Digital trends in transport and logistics, 2022. URL: https://www.itu.int/en/publications/.

Appendix A

Task Assignment

Vu Minh Quan

- Research on freight forwarding business and operations
- Conduct system analysis and use case design
- Design and develop database architecture

Hoang Dai Nam

- Research on freight forwarding business requirements
- Evaluate and select appropriate technologies
- Design system architecture
- Develop user interface and experience design