BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC TƯ THỤC QUỐC TẾ SÀI GÒN**

**BÁO CÁO ĐỒ ÁN CUỐI KÌ**

**MÔN HỌC LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OOP)**

**NĂM HỌC 2024-2025**

**TÊN ĐỀ TÀI**

**ENTERPRISE RESOURCE PLANNING**

**CHUYÊN NGÀNH: CÔNG NGHỆ THÔNG TIN**

Lớp: 24CNTT

Chủ nhiệm đề tài: Hồ Lâm Hải Phong

Nguyễn Đức Toàn

:

Giảng viên hướng dẫn: TS. Lê Ngọc Hiếu

**Thành phố Hồ Chí Minh, tháng 05/2025**



**PHIẾU CHẤM**

Họ và tên SV1: Hồ Lâm Hải Phong Lớp: 24CNTT

Họ và tên SV2: Nguyễn Đức Toàn Lớp: 24CNTT

Tên đồ án: **ENTERPRISE RESOURCE PLANNING**

**Đánh giá**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TT** | **Tiêu chí** | **Thang điểm** | **Điểm chấm** | **Ghi chú** |
| 1 | ***Hình thức trình bày:***  - Trình bày đúng quy định hướng dẫn (font, số trang, mục lục, bảng biểu, danh mục tài liệu tham khảo …).  - Không lỗi chính tả, lỗi đánh máy, lỗi trích dẫn tài liệu tham khảo.  - Trình bày đẹp, văn phong sáng, không tối nghĩa. |  |  |  |
| 2 | ***Nội dung*** | | | |
| Chương 1 |  |  |  |
| Chương 2 |  |  |  |
| Chương 3 |  |  |  |
| Chương 4 |  |  |  |
| Chương 5 |  |  |  |
| 3 | **TỔNG ĐIỂM** | |  |  |

**Điểm chữ: ………………………………………… (Làm tròn đến 1 số thập phân)**

**Giảng viên**

**TS. LÊ NGỌC HIẾU**

# DANH MỤC BẢNG BIỂU, HÌNH ẢNH

Hình 1.1 Table Complexity Distribution……………………………………6

Hình 1.2 Cross – Module Dependencies by Compont……………………...6

Hình 1.3 Table Category Distribution………………………………………7

Hình 2.1 Login Screen……………………………………………………..29

Hình 2.2 Account creation screen………………………………………….30

Hình 2.3 Total sales by month……………………………………………..30

Hình 2.4 Current inventory quantity……………………………………….31

Hình 2.5 Total order value by supplie...……………………………………31

Hình 2.6 Project allocation by status….……………………………………32

Hình 2.7 Allocate leave requests by status…………………………………32

Hình 2. 8 Allocate tasks by status…………………………………………..33

Hình 2.9 Directory Structure………………………………………………..33

Hình 2.10 Allocate employees by department………………………………36

**DANH MỤC CÁC CHỮ VIẾT TẮT**

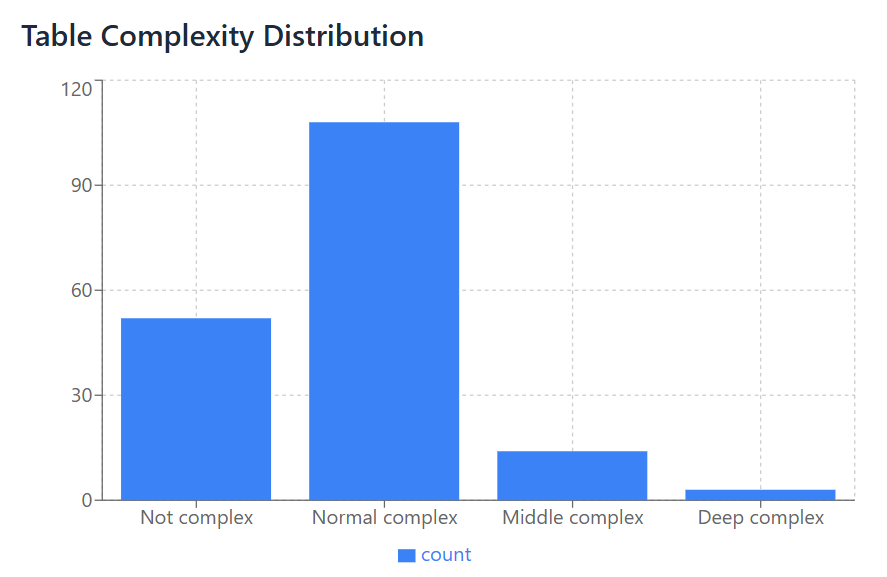
|  |  |
| --- | --- |
| Chữ viết tắt | Giải thích |
| BOM | Bill of Materials (Danh mục nguyên vật liệu. |
| CRM | Customer Relationship Management (Quản lý quan hệ khách hàng) |
| DAO | Data Access Object (Đối tượng truy cập dữ liệu) |
| ERP | Enterprise Resource Planning (Hoạch định nguồn lực doanh nghiệp) |
| ERD | Entity-Relationship Diagram (Sơ đồ quan hệ thực thể) |
| FIFO | First In, First Out (Nhập trước, xuất trước) |
| HR | Human Resources (Quản lý nhân sự) |
| KPI | Key Performance Indicator (Chỉ số hiệu suất chính) |
| LIFO | Last In, First Out (Nhập sau, xuất trước) |
| OEE | Overall Equipment Effectiveness (Hiệu quả thiết bị tổng thể) |
| PO | Purchase Order (Đơn đặt hàng) |
| ROI | Return on Investment (Tỷ suất hoàn vốn) |
| SLA | Service Level Agreement (Thỏa thuận mức dịch vụ) |
| SQL | Structured Query Language (Ngôn ngữ truy vấn có cấu trúc) |

**BẢNG TÓM TẮT ĐỀ TÀI**

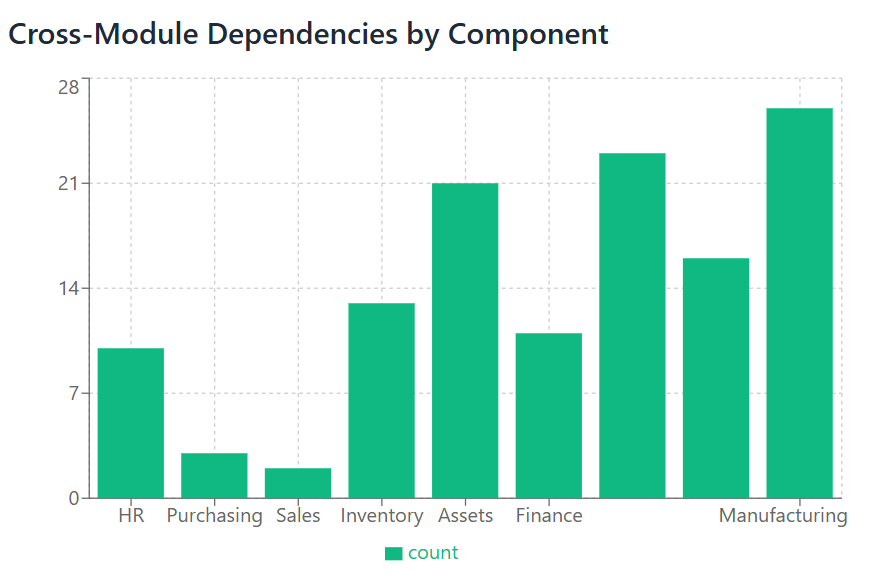
1. **ERP System Structure:**
   * The system includes key modules: HR, Purchasing, Sales, Inventory, Assets, Finance, Project Management, CRM, and Manufacturing.
   * Tables are categorized as Lookup Table, Reference Table, Dimension Table, Master Data Table, Fact Table, and Table with Hierarchy.
   * Each table is characterized by its dependency table count, cross-module dependencies, and complexity (Not Complex, Normal Complex, Middle Complex, Deep Complex).
2. **Complexity Classification:**
   * Complexity is determined based on the number of dependency tables:
     + Not Complex: 0 dependency tables (weight = 1).
     + Normal Complex: 1-2 dependency tables (weight = 1.5).
     + Middle Complex: 3-4 dependency tables (weight = 2).
     + Deep Complex: ≥ 5 dependency tables (weight = 2.5).
   * Examples: "Employee Absence" (HR) and "Payslips" (Payroll) are classified as Deep Complex due to 5-6 dependencies.
3. **Table Dependencies:**
   * Some tables have no dependencies (e.g., "Marital Status" in HR, "Vendors" in Purchasing).
   * High-dependency tables are typically Fact Tables, e.g., "Payslips" depends on 6 tables (Employees, Payroll, Bonuses, Deductions, Tax Rules, Earning Types).
   * Cross-module dependencies are common, e.g., "Purchase Orders" (Purchasing) depends on Module A - Core HR, and "Sales Activities" (CRM) depends on Module A - Core HR and Module C1 - Customer Data.
4. **Data Analysis:**
   * The document provides detailed information on tables within each module, including table categories, notes, and dependencies.
   * Some modules are highly interconnected, with Module A - Core HR being a central hub for dependencies (e.g., Employees, Departments Types).

**Analysis Report:**

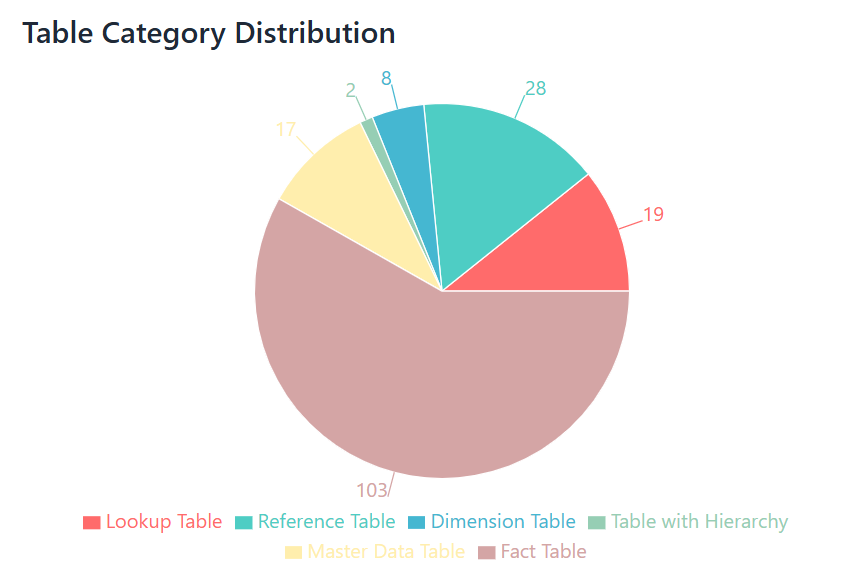
Below is an HTML report using React, Recharts, and Tailwind CSS to visualize the ERP system analysis, focusing on complexity distribution and cross-module dependencies**.**

****

Hình 1.1



Hình 1.2



Hình. 1.3

## INTRODUCTION

## *1. Reason for Choosing the Topic*

## In the era of Industry 4.0, the application of information technology in business management and operations is no longer a trend but has become a mandatory requirement. One of the most effective and comprehensive management systems today is the ERP (Enterprise Resource Planning) system. With its ability to integrate and manage all resources within an enterprise—such as human resources, finance, materials, production, sales, and more—ERP provides businesses with a unified, synchronized overview and helps improve operational efficiency.

## However, in reality, most current ERP systems like SAP, Oracle ERP, and Microsoft Dynamics come with high implementation and operational costs, along with complex configuration and training processes. This creates a significant barrier for small and medium-sized enterprises (SMEs), which make up a large proportion of Vietnam’s economy. Therefore, our group posed the question: "How can we build a simple, user-friendly ERP system that can run independently as a desktop application, is low-cost, yet still ensures basic management functionalities?"

## Based on our knowledge of Java programming, software development, databases, and system management, we decided to choose the topic: “Developing a Java Desktop Application for an ERP System.” This project not only allows us to apply our knowledge in practice but also contributes to providing a feasible solution for small-scale enterprises.

## *2. Research Background*

## The idea of building an ERP system first emerged in the 1960s with Material Requirements Planning (MRP) systems. Over time, MRP evolved into MRP II, and by the 1990s, the term ERP (Enterprise Resource Planning) was officially introduced. Since then, ERP systems have grown rapidly and have been widely applied across various sectors—from manufacturing and services to public administration.

## In addition to well-known commercial ERP systems like SAP, Oracle, and Microsoft Dynamics, the open-source community has also developed many free or low-cost ERP systems such as Odoo, ERPNext, and Dolibarr. These systems play a crucial role in providing solutions for small and micro-sized enterprises. However, most of them require a web server, database management configuration, and relatively high technical knowledge.

## In the academic and educational fields, many universities have designed simplified ERP simulation projects. These studies typically focus on streamlining the system while still ensuring essential functions such as human resource management, inventory, order processing, and reporting—without compromising the system’s overall integrity and scalability.

## Based on these foundations, our group chose to develop a desktop-based ERP application using Java. The goal is to create a system that does not rely on browsers or complex server infrastructure, and can run independently on personal computers or within a local network.

## *3. Research Objectives and Tasks*

#### *3.1. Research Objectives*

#### Study the theoretical foundation of ERP systems in order to extract core functions suitable for the intended application scale.

#### Survey the actual needs of small enterprises to identify both functional and non-functional requirements of the software.

#### Design the software architecture, including functional diagrams, data flow diagrams, and database schema diagrams.

#### Select appropriate technologies that are simple yet effective.

#### Develop the application on the Java Desktop platform using the selected technologies.

#### Conduct system testing to ensure stability and ease of use.

#### Complete the user manual and project report documentation.

#### *3.2. Research Tasks*

###### 3.2.1. Researching the Application Context and Requirements

###### The context of this project is based on the real needs of small businesses, where the use of management software remains very limited due to cost constraints, low technological readiness, and limited infrastructure. Through surveys, our group observed that many businesses still rely on Excel spreadsheets or physical notebooks to manage inventory, human resources, and orders. This approach not only consumes time but also introduces risks of data errors and difficulties in overall control.

###### The ERP application our team aims to develop must meet the following specific requirements:

###### Ease of use: A user-friendly interface that does not require in-depth computer knowledge.

###### No Internet requirement: Operable in a local network environment or completely offline.

###### Simple installation: Usable with a single executable file on Windows.

###### Scalability: Allows for the addition of new features in the future.

###### Secure data storage: Uses a local database such as SQLite to minimize the risk of data loss.

###### 3.2.2 Technology Evaluation and Selection

###### After evaluating available technology platforms, the team decided to choose Java as the main programming language for system development. Java is known for being powerful, stable, and well-suited for building desktop applications through GUI libraries such as Swing or JavaFX. Additionally, Java has a rich ecosystem, abundant documentation, and aligns well with the knowledge the team has acquired.

###### The selected technologies are as follows:

###### Programming language: Java SE 8+

###### User interface: Java Swing (due to ease of implementation and good compatibility)

###### Database: SQLite (lightweight, easy to integrate, no server installation required)

###### IDE: IntelliJ IDEA or NetBeans

###### Supporting libraries: JasperReports (for report generation), JFreeChart (for chart visualization), JCalendar (for date selection)

###### 3.2.3. Java Desktop Application Development

###### The application is developed using a modular approach, dividing the system into separate components for better manageability and scalability:

###### User management: login, access permission control

###### Human resources management: employee information, job titles, contracts, etc.

###### Product and inventory management: product details, stock quantity, unit of measurement, etc.

###### Order management: create orders, update order status, print invoices, etc.

###### Statistical reporting: sales reports, inventory reports, HR reports

###### The user interface is clearly designed and easy to use, incorporating components such as tables, forms, combo boxes, etc. The software supports report generation in PDF format and allows data backup when needed.

###### 3.2.4. Evaluation of Application Effectiveness and Scalability

###### Effectiveness Evaluation After the development and testing process, the team found that the Java Desktop ERP application has initially met the proposed goals. Specifically:

###### Functional Evaluation

###### The application has successfully integrated the essential modules of a compact ERP system, including human resources management, inventory management, order management, and reporting.

###### CRUD operations (Create, Read, Update, Delete) are executed quickly, accurately, and reliably.

###### The reporting feature utilizes the JasperReports library to generate professional PDF reports, helping managers easily track business performance.

###### User Interface and User Experience

###### The interface is designed to be simple, intuitive, and easy to use, even for users with limited technical knowledge.

###### Error messages and action confirmations are clearly handled, reducing the risk of user mistakes.

###### The application runs smoothly on Windows operating systems with average hardware configurations.

###### Performance

###### Thanks to the use of SQLite—a lightweight and fast database management system—the application performs well even as the volume of data increases.

###### Data is stored locally, ensuring privacy and minimizing the risk of data leakage over the network.

###### Real-world Deployment Potential

###### The application can be easily installed on multiple computers without complex configuration.

###### It does not depend on internet connectivity or servers, making it suitable for independent businesses or traditional business models.

###### 4. Research Object and Scope

#### *4.1 Research Object*

#### The research subject of this project is the ERP (Enterprise Resource Planning) system—specifically, the development of a software application that supports the management of essential business resources, including human resources, inventory, orders, and statistical reporting.

#### Additionally, the project focuses on exploring Java Desktop programming techniques such as user interface (UI) design, database connectivity (JDBC with SQLite), data management, report generation, and organizing source code following the object-oriented programming model.

#### Moreover, the team has investigated the real-world needs and operational workflows of small businesses to simulate and implement features that are both suitable and effective for practical use.

#### *4.2 Scope of the Study*

#### In terms of system functionality:

#### The system only implements basic ERP functions, including user management, human resources, inventory, order management, and report generation.

#### Advanced modules such as financial-accounting management, supply chain management, project management, or CRM (Customer Relationship Management) are not included.

#### In terms of technology platform:

#### The application is developed as a standalone desktop software; it is not deployed on web or mobile platforms.

#### Java is used as the programming language, combined with SQLite as the database system to simplify deployment and reduce system requirements.

#### In terms of target users:

#### The system is intended for small and medium-sized enterprises or individuals and startup teams who need a simple, cost-effective management solution and are not able to deploy complex ERP systems.

#### Users of the software are not required to have deep IT knowledge—only basic computer operation skills are sufficient.

#### In terms of deployment scope:

#### The application is tested in a simulated environment or within the research team. It has not yet been widely deployed in real-world business settings.

#### The data used in the software consists mainly of sample data, not actual business data from specific companies.

#### *5. Research Methodology*

#### *5.1. Theoretical Research Methods*

#### Studying specialized materials: The team researched textbooks, academic materials, and professional resources related to ERP systems, information systems analysis, database design, and Java Desktop programming to acquire solid foundational knowledge.

#### Referencing real-world solutions: The team studied popular ERP software such as Odoo, SAP Business One, and FastWork to extract commonly used core functions tailored to small and medium-sized enterprises.

#### Exploring appropriate technologies: Technologies relevant to the project, including Java Swing, JDBC, SQLite, and JasperReports, were explored in depth to ensure their suitability for the team's goals and technical capacity.

#### System requirement analysis: Based on typical small business operations, the team conducted business requirement analysis, identified entities and their relationships, and developed a general system function diagram for the software.

#### *5.2. Practical Research Methods*

#### Designing and developing a prototype application: The team implemented the analyzed features by programming the user interface, handling events, constructing the database, and writing object-oriented code.

#### Testing and debugging: After completing the main modules, the team conducted system testing to identify and fix logic errors, UI issues, and performance problems to ensure stable operation.

#### Collecting feedback from test users: During the testing phase, feedback was gathered from friends, relatives, and individuals with management experience to improve the software's practicality and usability.

#### Optimizing and finalizing the product: Based on testing results and user feedback, the team refined the application, packaged the software, and prepared the user manual.

#### *6. Thesis Structure*

#### Chapter 1: Introduction

#### Motivation for the topic

#### Research objectives and tasks

#### Research subjects and scope

#### Chapter 2: Theoretical Background

#### Introduction to ERP systems

#### Related models and technologies: Java, SQLite database, Swing GUI design, JasperReports

#### Chapter 3: System Analysis and Design

#### Business requirements analysis

#### System function diagram (Use Case Diagram)

#### Database design

#### User interface design

#### Chapter 4: Application Development and Deployment

#### Description of the programming process and code organization

#### Implementation of functional modules: Human Resources, Inventory, Orders, Reports

#### Main interface and user navigation

#### Chapter 5: Testing and Evaluation

#### Testing process

#### Experimental results

#### Evaluation of application effectiveness and scalability

#### Chapter 6: Conclusion and Future Development

#### Summary of achieved results

#### Remaining limitations

#### Future directions for software improvement

# MAIN CONTENT

# Chapter 1: THEORETICAL FOUNDATION AND RESEARCH STATUS

## 1.1. Scientific Foundation

### 1.1.1. Overview of ERP Systems

###### Enterprise Resource Planning (ERP) is a comprehensive software solution designed to manage and integrate core business processes across an organization, including finance, human resources, manufacturing, supply chain, customer service, and more. The central goal of ERP is to establish a single, centralized data source that allows departments to access, share, and update information in real time.

###### The key feature of ERP systems is inter-departmental integration. Data is entered once and becomes available across all modules. For instance, when the sales department creates an order, the inventory system is automatically updated to reflect stock changes, and the accounting system receives the necessary information for billing. This eliminates redundant tasks, minimizes human error, and improves operational efficiency.

###### With the rise of digital transformation, ERP systems are no longer exclusive to large corporations. The emergence of open-source ERP and Desktop ERP solutions has made it feasible for small and medium-sized enterprises to implement ERP to enhance business management and competitiveness.

### 1.1.2. Object-Oriented Programming in ERP Development

###### Object-Oriented Programming (OOP) is a software development paradigm based on the concept of "objects"—entities that encapsulate both data (state) and methods (behavior). In ERP development, OOP serves as a systematic and efficient way to model business processes.

###### By abstracting real-world business components into classes and objects, developers can create ERP systems that are modular, extensible, and easy to maintain. For example:

###### A Employee class might include attributes such as name, employee ID, role, and salary, with methods to calculate bonuses or retrieve data.

###### An Order class may manage a list of products, customer data, and processing status.

###### Key OOP principles include:

###### Inheritance: Promotes code reuse. For example, a User class can be inherited by Customer and Employee classes.

###### Encapsulation: Protects internal data and exposes only necessary interfaces.

###### Polymorphism: Allows objects to be treated as instances of their parent class with behavior defined at runtime.

###### OOP enables ERP systems to be structured into well-defined components, facilitating debugging, upgrades, and future feature extensions.

### 1.1.3. JavaFX Technology in Desktop Application Development

###### JavaFX is a modern platform for developing rich graphical user interfaces (GUIs) for Java desktop applications. As the successor to the older Swing library, JavaFX offers improved functionality, better separation of concerns, and a more elegant user experience.

###### Advantages of JavaFX for ERP applications include:

###### FXML support: Allows UI design to be separated from application logic, following the MVC (Model-View-Controller) pattern.

###### CSS-based styling: Enables interface customization consistent with corporate branding.

###### Built-in chart and multimedia capabilities: Ideal for displaying dynamic business charts and reports.

###### Integration with third-party libraries such as JFreeChart and JDBC for charting and database connectivity.

###### In ERP development, JavaFX plays a pivotal role in building user-friendly interfaces such as data entry forms, tables, login windows, and most importantly, delivering a professional and modern user experience.

### 1.1.4. MySQL Database Management System and SQL

###### Databases are the backbone of any ERP system. In this project, MySQL is chosen for its performance, stability, scalability, and strong community support. MySQL utilizes SQL (Structured Query Language) to define, query, manipulate, and manage data in relational databases.

###### Highlights of MySQL in ERP systems:

###### Structured storage: Each business entity such as products, orders, or invoices is stored in a separate, well-related table using primary and foreign keys.

###### Optimized queries: Indexing and JOIN operations allow fast and efficient data retrieval.

###### Security and user management: Enables access control and role-based permissions for sensitive business data.

###### By integrating MySQL with Java through JDBC (Java Database Connectivity), ERP systems can ensure seamless and secure communication between the user interface and the database layer.

### 1.1.5. Data Structures and Java Collections Framework

###### The Java Collections Framework (JCF) is a fundamental component of the Java language that provides powerful data structures such as lists, sets, maps, queues, along with utility algorithms for manipulation.

###### In ERP development, JCF serves as a versatile toolkit for:

###### Temporary in-memory data handling before committing changes to the database.

###### Efficient filtering, searching, sorting, and aggregation of business entities like orders, products, or customers.

###### Managing complex object relationships, e.g., mapping products to categories using Map<String, List<Product>>.

###### Commonly used structures:

###### ArrayList: For maintaining ordered collections such as invoices or employee lists.

###### HashMap: For quick key-based retrieval, like mapping product codes to product objects.

###### TreeSet: For storing and automatically sorting elements, such as products by price or name.

###### JCF simplifies code, improves performance, and is essential for processing large volumes of records in ERP systems.

### 1.1.6. JFreeChart Library and Principles of ERP Data Visualization

###### In ERP systems, presenting information as plain tables often makes data interpretation difficult. Data visualization becomes crucial for providing clarity, enhancing decision-making, and communicating insights effectively.

###### JFreeChart is a powerful open-source Java library that supports creating a wide range of 2D charts:

###### Bar Charts: Compare sales across months or stock levels across products.

###### Line Charts: Monitor trends over time.

###### Pie Charts: Show percentage breakdowns, such as order statuses or revenue contribution by category.

###### Benefits of data visualization in ERP:

###### Reduces analysis time: Decision-makers can quickly grasp key metrics in seconds.

###### Highlights business trends or issues: For instance, a stock chart can immediately show which items are running low.

###### Enhances UI aesthetics and usability, making the ERP interface more engaging and professional.

###### Principles of effective visualization include:

###### Selecting the right chart type for the data (e.g., temporal → line, proportion → pie, comparison → bar).

###### Minimizing visual clutter: Use consistent color schemes and clear labels.

###### Focusing on core insights, avoiding unnecessary complexity.

###### JFreeChart empowers ERP systems to go beyond data entry and reporting, transforming them into strategic business intelligence tools.

## 1.2. Research Status

### 1.2.1. The Current State of Modern ERP Technology

In the digital era, ERP technology has undergone significant evolution to adapt to the increasing complexity and dynamism of business operations. Traditional on-premise ERP solutions—once dominant—are now being supplemented or replaced by cloud-based ERP, modular systems, and mobile-compatible platforms. These advancements are reshaping the landscape of enterprise resource planning in several key ways:

* Cloud Computing Integration: Modern ERP systems are frequently deployed via cloud platforms (e.g., SAP S/4HANA Cloud, Oracle ERP Cloud), offering scalability, lower upfront infrastructure costs, automatic updates, and accessibility from any location. This transition facilitates real-time collaboration and data centralization across geographically dispersed teams.
* Modular Architecture: Instead of large monolithic systems, modern ERP platforms are designed in modular components such as Human Resource Management (HRM), Customer Relationship Management (CRM), Inventory, and Accounting. Businesses can implement modules gradually, based on specific needs, allowing better resource allocation and reduced implementation risk.
* AI and Machine Learning: Many advanced ERP systems now embed artificial intelligence to provide predictive analytics, automated decision-making, and anomaly detection in business operations. For example, systems can forecast sales trends or automatically flag suspicious transactions.
* Mobile ERP and Cross-platform Access: The rise of smartphones and tablets has prompted ERP vendors to develop mobile-compatible interfaces, enabling employees to access real-time dashboards, approve workflows, and manage tasks remotely.
* Open-source ERP systems: Solutions like Odoo, ERPNext, and Dolibarr offer customizable, cost-effective alternatives to proprietary software, especially for small and medium-sized enterprises. These systems allow developers to modify source code and integrate with third-party services easily.

Despite these advances, several challenges persist, such as data migration complexities, user training demands, and integration difficulties with legacy systems. Additionally, security and data privacy concerns remain paramount, particularly for cloud-based solutions.

The current state of ERP technology reflects a shift toward flexibility, intelligence, and user-centric design, aiming to empower businesses with tools that are not only functional but also strategic in nature.

### 1.2.2. Analysis of Similar ERP Applications

To inform the design and development of our ERP system, it is critical to examine existing ERP applications that address similar domains or business problems. Through this comparative analysis, we can identify industry best practices, design patterns, and potential gaps that our system can address.

**ERPNext**

* Overview: An open-source ERP system built using Python and JavaScript, ERPNext supports modules such as accounting, inventory, HR, and project management.
* Strengths:
  + Intuitive web interface.
  + Highly modular and customizable.
  + Strong community support.
* Limitations:
  + Performance can degrade with large datasets.
  + Complex installation process for self-hosting.

**Odoo**

* Overview: A well-known open-source ERP with both free and paid versions. Offers over 30 modules for different business needs.
* Strengths:
  + Extensive module ecosystem.
  + Modern UI and mobile support.
  + Active developer community.
* Limitations:
  + Some essential features are only available in the Enterprise (paid) version.
  + Can be resource-intensive on smaller servers.

**Dolibarr**

* Overview: A lightweight, PHP-based ERP and CRM system designed for small businesses.
* Strengths:
  + Easy to install and use.
  + Integrated suite of business tools.
* Limitations:
  + Less scalable for enterprise-level operations.
  + Limited third-party integration capabilities.

**SAP Business One and Oracle NetSuite**

* Overview: Proprietary ERP platforms tailored for medium and large enterprises.
* Strengths:
  + Robust features, enterprise-level security.
  + Global support and compliance features.
* Limitations:
  + High licensing and implementation costs.
  + Less flexibility for customization compared to open-source platforms.

**Key Insights for Our System**

* Usability: Many successful ERP systems prioritize intuitive navigation and minimal learning curves.
* Customization: Open-source ERP systems often gain favor due to their adaptability to unique business needs.
* Visualization: Data dashboards and real-time reporting features are essential for strategic decision-making.
* Integration: API-based architectures enable ERP systems to communicate with other platforms (e.g., e-commerce, accounting software).

Based on this analysis, our proposed ERP system aims to combine the strengths of modern ERP platforms—such as modularity, user-friendly design, data visualization, and local deployment—with the customization flexibility provided by open-source technologies. This positions the system as an ideal solution for small-to-medium-sized businesses that require desktop-based, cost-effective, and easily maintainable ERP solutions.

# CHAPTER 2: PROPOSED SOLUTION AND ERP SYSTEM DESIGN ANALYSIS

## 2.1. Proposed Solution

#### In response to the growing demand for digital transformation among small and medium-sized enterprises (SMEs), this project proposes the development of a modular ERP desktop application built using JavaFX, integrated with MySQL for data storage and JFreeChart for real-time data visualization. The system is designed to manage key business processes such as inventory control, sales management, human resources, and financial reporting.

#### The proposed ERP solution emphasizes simplicity, cost-effectiveness, and user-friendliness while still maintaining scalability and extensibility for future enhancements. It is particularly suitable for SMEs that lack the infrastructure for large-scale ERP systems but still require centralized, accurate, and accessible business data.

### 2.1.1. General Model of the ERP System

#### The architecture of the proposed ERP system follows a multi-layered design consistent with the Model-View-Controller (MVC) paradigm. The system is developed as a Java Desktop Application with the following architectural components:

#### *1. Presentation Layer (View)*

#### Implemented using JavaFX, this layer provides a user-friendly graphical interface for interacting with the system.

#### Key elements include login screens, navigation menus, data entry forms, data tables, and charts.

#### *2. Application Layer (Controller)*

#### Manages logic and data flow between the user interface and the business logic.

#### Coordinates actions such as validating inputs, triggering calculations, and invoking database operations.

#### *3. Business Logic Layer (Model)*

#### Implements the core functionality of ERP modules:

#### Inventory Management: Add, update, track product quantities.

#### Sales Management: Process invoices, calculate revenue.

#### Human Resource Management: Manage employee records, calculate salaries.

#### Accounting: Generate balance sheets, transaction logs.

#### *4. Data Layer (Database)*

#### A MySQL relational database is used to store persistent data.

#### Data entities such as Employee, Product, Invoice, and User are represented in structured tables with defined relationships.

#### CRUD operations are handled through JDBC (Java Database Connectivity).

#### *5. Visualization Layer*

#### Employs JFreeChart to display data trends and summary reports, helping users make informed decisions through bar charts, pie charts, and line graphs.

#### System Workflow Example:

#### A user logs in → adds a product to inventory → creates a sales invoice → stock levels auto-update → revenue reflected in financial reports → charts updated in real time.

#### This modular and extensible structure allows the system to be expanded with additional modules (e.g., procurement, CRM) as organizational needs evolve.

#### *2.1.2. Functional Requirements*

#### Functional requirements define the specific operations that the ERP system must perform to fulfill user expectations. The proposed system supports the following key functions across its major modules:

#### *A. User Management*

#### User authentication via login credentials.

#### Role-based access control (e.g., Admin vs. Employee permissions).

#### User registration and password management.

#### *B. Inventory Management*

#### Add, update, delete, and search product information.

#### Monitor stock levels and generate low-stock alerts.

#### Record incoming shipments and outgoing sales.

#### *C. Sales and Invoice Processing*

#### Create new sales invoices with customer and product details.

#### Automatically compute total amount and tax.

#### Track order status (Pending, Paid, Shipped).

#### View invoice history and export data as reports.

#### *D. Human Resources Management*

#### Manage employee records: name, role, salary, department.

#### Generate monthly salary reports.

#### Track employee status (active, inactive, on leave).

#### *E. Financial and Statistical Reporting*

#### Aggregate financial data for monthly/quarterly reporting.

#### Display revenue, expense, and profit using charts.

#### Export reports to Excel or PDF formats.

#### *F. Data Backup and Recovery*

#### Ability to export and import database backups.

#### Prevent data loss in case of system failure.

#### These functional requirements ensure that the ERP system serves as a centralized platform for business management, improving coordination and decision-making across departments.

### 2.1.3. Non-functional Requirements

#### In addition to core functionalities, the system must satisfy various non-functional requirements that define its overall quality, usability, and performance. These include:

#### *1. Usability*

#### User interface should be intuitive, visually consistent, and accessible to non-technical users.

#### Form validations and tooltips must be implemented to reduce input errors.

#### *2. Performance*

#### The system must be able to handle up to several thousand records efficiently.

#### Response time for CRUD operations should be under 2 seconds on standard hardware.

#### *3. Scalability*

#### Designed with modular architecture, enabling additional modules (e.g., CRM, procurement) to be added without major system overhaul.

#### Database schema designed for growth in data volume and user load.

#### *4. Security*

#### Passwords are hashed and stored securely.

#### Access to critical features is restricted based on user roles.

#### Input validation is enforced to prevent SQL injection or data corruption.

#### *5. Reliability and Availability*

#### System should remain stable during normal operations, with proper error handling and logging.

#### Daily database backups ensure business continuity.

#### *6. Maintainability*

#### The codebase is structured following software engineering best practices (e.g., separation of concerns, use of design patterns).

#### Clear documentation supports future updates and troubleshooting.

#### *7. Portability*

#### The system should run on any machine with Java Runtime Environment (JRE) installed.

#### Platform-independent deployment (Windows/Linux/macOS).

#### These non-functional requirements are essential to ensure the ERP system is sustainable, secure, and adaptable to future demands, reflecting professional standards in enterprise software development.

#### 

## 2.2. System Architecture and User Interface Design Analysis

### 2.2.1. ERP System Architecture

The architectural design of the ERP system is a foundational element that ensures modularity, maintainability, scalability, and security. The proposed ERP system adopts a **multi-tier architecture**, following the **Model-View-Controller (MVC)** design pattern and supporting structured communication between components. This separation of concerns simplifies development, enhances flexibility, and ensures ease of future upgrades.

**A. Overview of System Architecture**

The ERP system is developed as a **JavaFX desktop application** supported by a **MySQL relational database**. The system consists of four primary layers:

1. **Presentation Layer (View – JavaFX GUI)**
   * Handles all user interactions.
   * Provides forms, menus, tables, and charts for input and output.
   * Ensures responsive and user-friendly design.
2. **Application Logic Layer (Controller – Java Classes)**
   * Coordinates interaction between the view and the model.
   * Processes user inputs and invokes business logic.
   * Handles validation, navigation, and flow control.
3. **Business Logic Layer (Model – Java Classes & Objects)**
   * Contains the rules, methods, and logic for business modules (Inventory, Sales, HR, Accounting).
   * Encapsulates data handling logic such as calculations, conditions, and transaction processing.
4. **Data Access Layer (Database & JDBC)**
   * Uses JDBC (Java Database Connectivity) to communicate with the **MySQL database**.
   * Performs CRUD operations: Create, Read, Update, Delete.
   * Ensures integrity and security of stored data.

**B. Key Architectural Advantages**

* **Modularity**: Each module (Inventory, HR, etc.) is encapsulated independently.
* **Reusability**: Core components like database handlers and UI templates are reusable.
* **Testability**: Separation into layers supports unit testing and debugging.
* **Security**: Role-based access and validation are built into controller and model logic.
* **Extensibility**: Future modules (e.g., procurement, CRM) can be integrated with minimal impact.

### 2.2.2. User Interface (UI) Design Analysis

The success of an ERP system depends not only on its functionality but also on the **clarity, intuitiveness, and usability of its user interface**. The proposed application leverages **JavaFX** for building responsive, interactive, and platform-independent desktop GUIs.

**A. Design Principles**

The interface design is based on the following key principles:

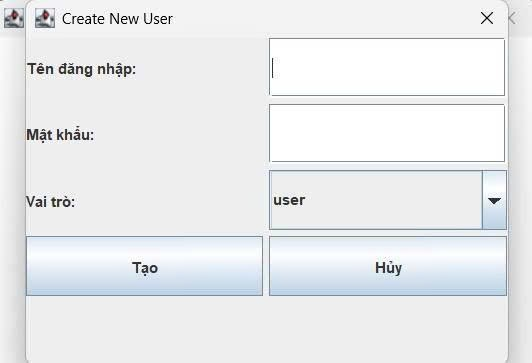
* **Consistency**: Layouts, color schemes, and iconography are kept uniform across modules.
* **Simplicity**: Avoids clutter by displaying only essential elements on each screen.
* **Feedback**: Instant visual or auditory responses are provided for user actions (e.g., confirmations, warnings).
* **Accessibility**: Fonts, buttons, and navigation paths are optimized for ease of use.
* **Modularity**: Interfaces are separated into panes/tabs for different functions.

**B. Main UI Components**

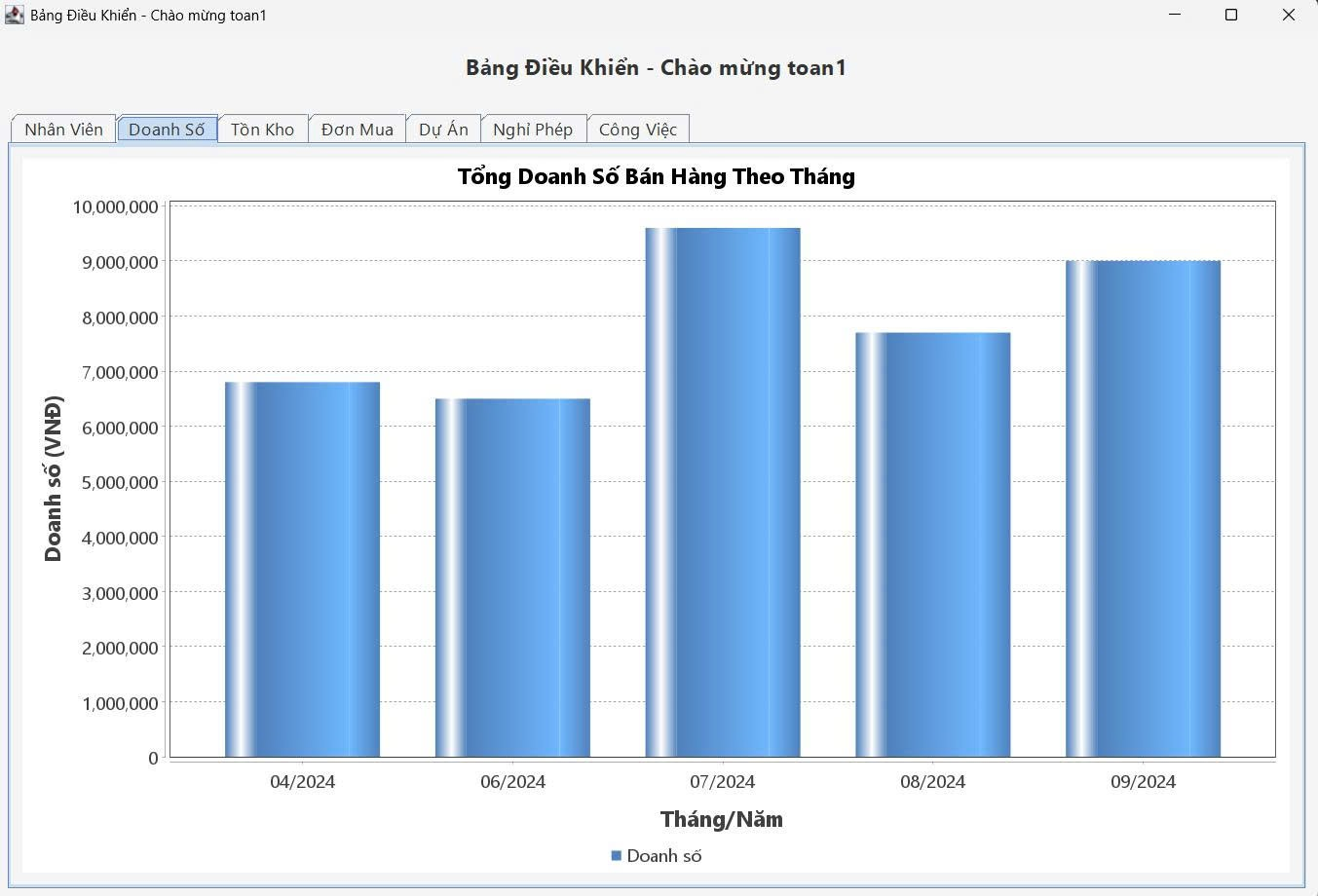
1. ***Login Screen***

Hình 2.1. Login screen.

1. ***Account creation screen.***



*Hình 2.2 Account creation screen***.**

1. ***Total sales by month***

*Hình 2.3 Total sales by month*

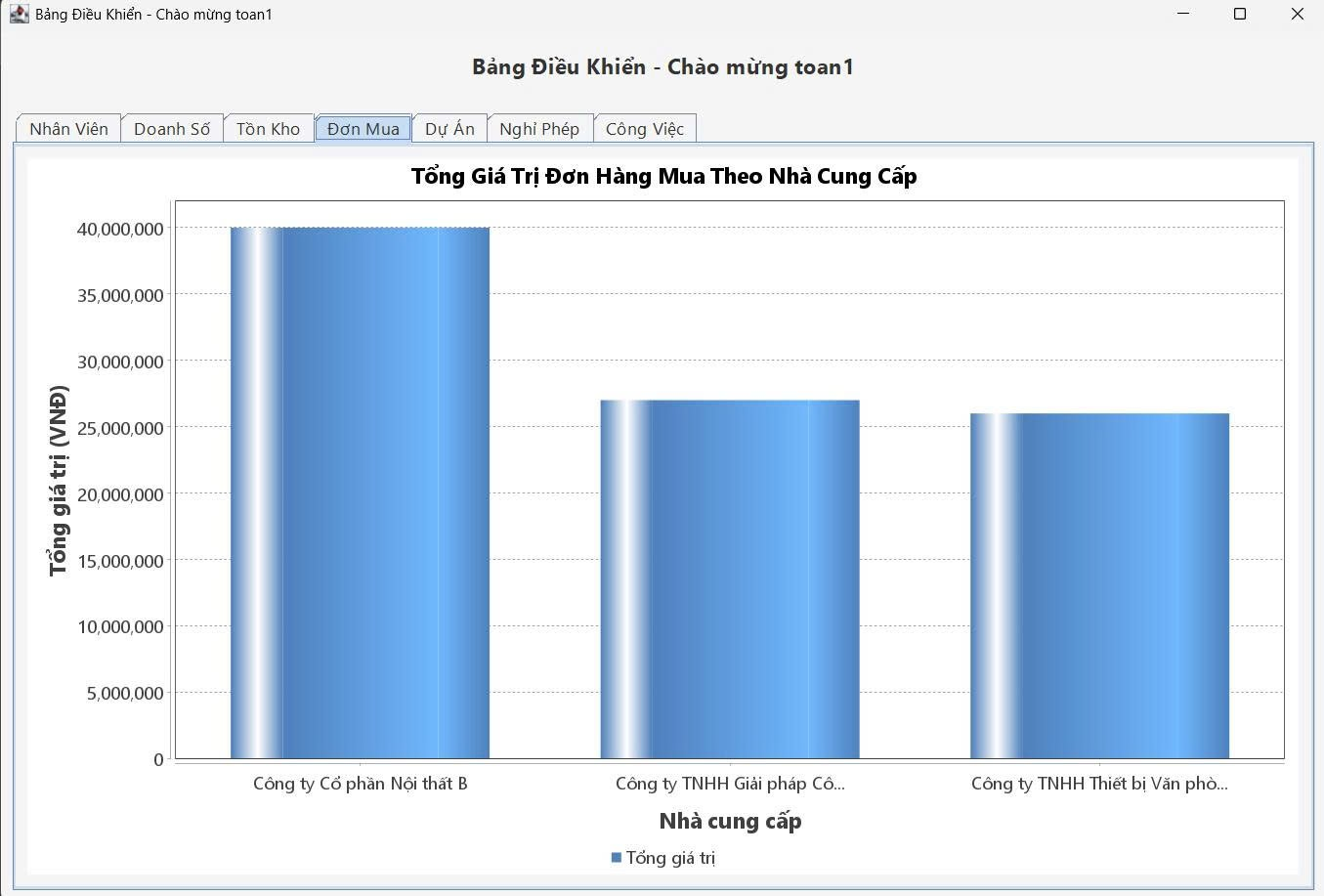
1. ***Current inventory quantity***

***A screenshot of a graph

AI-generated content may be incorrect.***

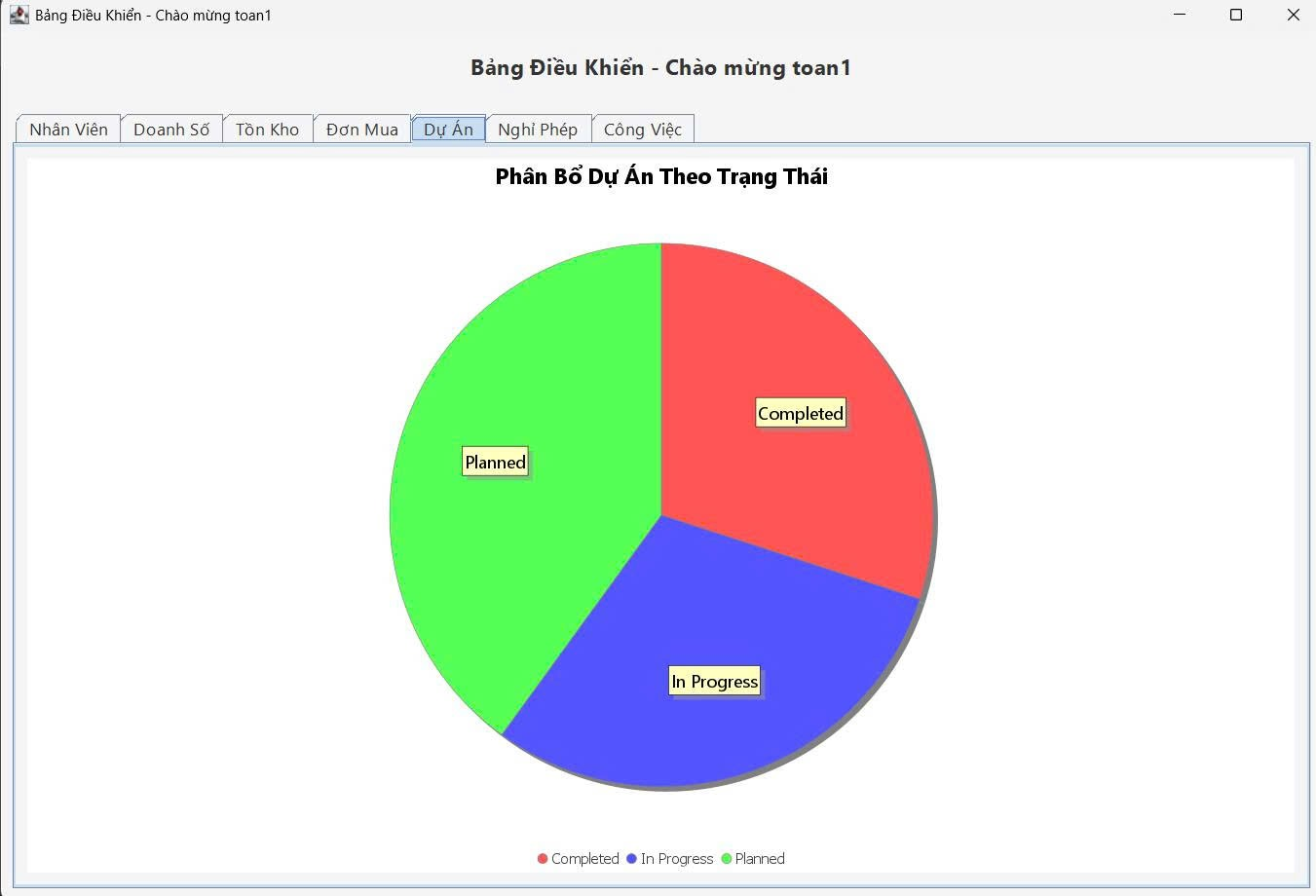
*Hình 2.4 Current inventory quantity*

1. ***Total order value by supplie***



*Hình 2.5 Total order value by supplie*

1. ***Project allocation by status***

******

*Hình 2.6 Project allocation by stats*

1. ***Allocate leave requests by status***

***A screenshot of a computer

AI-generated content may be incorrect.***

*Hình 2.7 Allocate leave requests by status*

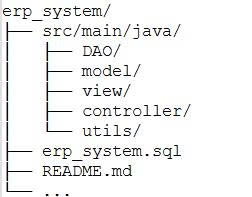
1. ***Allocate tasks by status***

***A screenshot of a computer

AI-generated content may be incorrect.***

*Hình 2.8 Allocate tasks by status*

1. ***Directory Structure***

******

*Hình 2.9 Directory Structure*

## 2.3. System Workflow and User Interface

### This section presents the architectural design and user interface flow of the ERP desktop application. The system aims to integrate and automate multiple key functions within an enterprise environment, including human resources, payroll, inventory, and analytical dashboards. Developed using JavaFX for the graphical user interface, MySQL for database storage, and JFreeChart for visual analytics, the system is designed to offer both functional robustness and ease of use. Each module is accessible via a role-based navigation mechanism and is constructed with user-centered design principles to enhance the overall user experience.

### 2.3.1 Use Case Diagram

### The Use Case Diagram provides a high-level visualization of the interactions between the system and its users. It identifies the primary actors—Administrator, Employee, and Accountant—and outlines their respective capabilities within the system. The Administrator has access to user management, permission assignment, and system configuration. Employees can log in to the system, view personal payroll records, and update certain information. Accountants are authorized to manage salary computation, generate payslips, and produce reports.

### This diagram serves as a foundation for understanding how various user roles interact with different functional components of the ERP system.

### 

### 2.3.2 Login Workflow and Dashboard Interface

##### Upon launching the application, users are greeted with a login screen designed for secure authentication. The login module verifies the user’s credentials against the database and redirects the user to a customized dashboard based on their access level.

##### The Dashboard Interface is designed for clarity and quick access to essential modules. It includes:

##### A side navigation menu listing all available modules (e.g., Core HR, Payroll, Inventory, Reports).

##### Summary statistics panels (e.g., number of employees, pending leave requests).

##### Interactive charts to present real-time data insights.

##### This layout enables users to quickly grasp system status and navigate to specific modules efficiently.

### 2.3.3 Core Data Management (HR, Payroll, Inventory Modules)

##### The system is structured around modular components to ensure flexibility and maintainability. The following core modules have been implemented:

##### Core HR Module: Allows administrators and HR personnel to manage employee records, contract details, department assignments, and attendance data. CRUD (Create, Read, Update, Delete) operations are supported for all employee-related information.

##### Payroll Module: Facilitates automated salary calculation based on attendance, overtime, deductions, and allowances. The module includes functionalities for payslip generation and monthly payroll processing.

##### Inventory Module: Designed for basic stock management, the module supports tracking of incoming and outgoing inventory items. It includes stock level monitoring, reorder notifications, and simple reporting features.

##### Each module features a well-structured graphical interface with input forms, data tables, and action buttons. Data validation and confirmation dialogues are implemented to minimize input errors and enhance user experience.

### 2.3.4 User Management and Role-Based Access Control

##### The User Management module is a crucial part of system administration. It allows the system administrator to:

##### Create and remove user accounts.

##### Assign roles such as Admin, Accountant, or Employee.

##### Set access permissions for each module.

##### Role-based access control (RBAC) ensures that sensitive operations (e.g., payroll modifications or user creation) are only accessible to authorized personnel. This security model not only protects data integrity but also simplifies the user experience by hiding irrelevant modules.



*hình 2. 10 Allocate employees by department.*

### 2.3.5 Analytics and Statistical Visualization

##### To support decision-making, the system integrates dynamic visualizations using JFreeChart. Users can access the Analytics Dashboard to review various key performance indicators (KPIs) through interactive charts.

##### Bar Charts: Display the distribution of employees across departments or monthly payroll summaries.

##### Pie Charts: Visualize attendance status, types of employment contracts, or departmental expenses.

##### Line Charts: Track inventory fluctuations over time or payroll trend progression.

##### These visual representations offer actionable insights and assist managers in identifying operational patterns and potential inefficiencies.

## 2.4. MySQL Database Structure

### The design of the MySQL database forms the backbone of the ERP desktop application, ensuring efficient storage, retrieval, and integrity of enterprise data. The database was structured to support modular expansion while maintaining clear relationships between different business entities such as employees, departments, inventory items, and payroll records. This section provides an in-depth look into the database architecture, including the Entity Relationship Diagram (ERD), a description of each primary table, and the constraints and relationships that ensure data consistency and reliability.

### 2.4.1. Entity-Relationship Diagram (ERD)

### The ERD (Entity-Relationship Diagram) illustrates the high-level data model and highlights how different entities within the system are related. Major entities include:

### User: Represents system users, each with login credentials and assigned roles.

### Employee: Stores detailed personnel information, including name, date of birth, contact information, and employment status.

### Department: Contains records of organizational departments, linked to employees.

### Payroll: Manages salary-related records, linking employees to their computed pay and deduction details.

### Inventory: Stores product or item data such as stock levels, prices, and warehouse location.

### Attendance: Tracks daily check-ins and check-outs for employee timekeeping.

### Relationships are defined using primary and foreign keys to model real-world business logic. For example:

### An Employee belongs to one Department, while a Department may have many Employees.

### Each Payroll record is linked to exactly one Employee.

### The User table links to the Employee table in systems where employees can log into the system.

### 2.4.2. Table Descriptions

##### The system’s database consists of several relational tables. The key tables are described as follows:

##### a. User Table

##### user\_id (INT, PRIMARY KEY): Unique identifier for each user.

##### username (VARCHAR): System login name.

##### password (VARCHAR): Hashed user password.

##### role (ENUM): Defines role (Admin, Employee, Accountant).

##### employee\_id (INT, FOREIGN KEY): Links to the Employee table.

##### b. Employee Table

##### employee\_id (INT, PRIMARY KEY): Unique employee ID.

##### name, gender, dob, email, phone, address: Employee personal details.

##### department\_id (INT, FOREIGN KEY): Links to Department table.

##### position, contract\_type: Employment information.

##### c. Department Table

##### department\_id (INT, PRIMARY KEY): Unique department ID.

##### department\_name (VARCHAR): Name of the department.

##### manager\_id (INT, FOREIGN KEY): References an employee as manager.

##### d. Payroll Table

##### payroll\_id (INT, PRIMARY KEY): Unique payroll record ID.

##### employee\_id (INT, FOREIGN KEY): References Employee.

##### base\_salary, allowance, deduction, net\_salary: Fields for salary computation.

##### pay\_date (DATE): Salary issue date.

##### e. Inventory Table

##### item\_id (INT, PRIMARY KEY): Unique item ID.

##### item\_name, quantity, unit\_price, location: Inventory details.

##### updated\_at (TIMESTAMP): Last update time.

##### f. Attendance Table

##### attendance\_id (INT, PRIMARY KEY)

##### employee\_id (INT, FOREIGN KEY)

##### check\_in (DATETIME), check\_out (DATETIME)

##### work\_hours: Calculated field.

##### 2.4.3. Constraints and Relationships

##### The integrity of the database is maintained through a set of constraints and defined relationships, which ensure that data adheres to business rules:

##### Primary Keys are used in every major table to uniquely identify records.

##### Foreign Keys link related tables, enabling relational integrity. For example:

##### employee\_id in the Payroll table is a foreign key referencing the Employee table.

##### department\_id in the Employee table is a foreign key referencing the Department table.

##### NOT NULL constraints are applied to critical fields such as username, employee name, and base salary to ensure completeness.

##### ENUM types are used for controlled values (e.g., gender, user roles).

##### ON DELETE CASCADE / SET NULL rules are used for foreign keys where applicable, ensuring that deletion of a parent record does not leave orphaned child records.

##### These constraints help in:

##### Preventing duplicate entries.

##### Enforcing referential integrity between entities.

##### Supporting complex queries across modules.

# Chapter 3: System Operation Mechanism of the ERP Application

## 3.1. Software Package and Class Layer Implementation

### The ERP (Enterprise Resource Planning) application under development follows a layered and modular design approach to ensure maintainability, scalability, and ease of understanding for future enhancement and debugging. The system is developed using the Java programming language and employs JavaFX for the user interface, MySQL for the relational database, and follows Object-Oriented Programming (OOP) design patterns and MVC (Model–View–Controller) architecture. This section provides a comprehensive explanation of the main class layers and software packages that constitute the core operational structure of the ERP system.

### 3.1.1. Model Layer

### The Model Layer is foundational to the system's data representation. It encapsulates the application's business entities and maps them directly to the structure of the underlying database. These classes are often referred to as *entity classes* or *domain objects* and are typically stored in a package named model.

### Each model class represents a single table in the MySQL database, with attributes corresponding to the table's columns. For example:

### Employee.java contains fields like employeeId, name, email, department, salary, and position.

### Department.java defines a department's identity and name.

### Payroll.java contains attributes such as payrollId, employeeId, basicSalary, bonuses, deductions, and netSalary.

### Each model class includes:

### A no-argument constructor and a parameterized constructor for flexible object instantiation.

### Getters and setters for encapsulated access to class attributes.

### Utility methods such as toString(), equals(), and hashCode() when required.

### These classes act as data carriers between the database and the GUI, ensuring the information retrieved from the database can be easily manipulated and displayed in the interface.

### 3.1.2. DAO (Data Access Object) Layer

### The Data Access Object (DAO) layer encapsulates all interactions with the relational database, isolating raw SQL queries from the rest of the application. It is generally organized under a package named dao.

### Responsibilities of this layer include:

### Establishing and managing database connections.

### Executing SQL statements (SELECT, INSERT, UPDATE, DELETE).

### Mapping ResultSet objects into model objects (using Java's JDBC API).

### Ensuring secure access to the database through PreparedStatement to prevent SQL injection attacks.

### Each DAO class is specialized for a specific data entity, for example:

### EmployeeDAO.java contains methods to insert, update, delete, and retrieve employee data.

### DepartmentDAO.java manages departments.

### PayrollDAO.java provides access to salary-related data and calculations.

### InventoryDAO.java manages inventory records, useful in the logistics module.

### Standard methods in a DAO class include:

### List<T> getAll()

### T getById(int id)

### void insert(T entity)

### void update(T entity)

### void delete(int id)

### This layer improves maintainability by decoupling data operations from business logic and UI operations.

### 3.1.3. Service Layer

### The Service Layer, usually implemented in the service package, performs business logic and data orchestration tasks. This layer sits between the DAO and Controller (UI) layers and plays a pivotal role in processing business rules.

### For instance:

### In PayrollService, before inserting payroll data, it calculates the total net salary by applying rules on base salary, allowances, bonuses, and deductions.

### In EmployeeService, it might verify whether an employee's email is valid and check for duplicate records before calling DAO to insert the data.

### Typical tasks in the service layer include:

### Validation of data before insertion or updates.

### Business rule enforcement, such as bonus rules, tax calculations, etc.

### Combining results from multiple DAO classes to form composite views.

### Error handling and exception wrapping to avoid leaking database errors to the UI.

### This layer provides a clean and abstracted interface to the GUI or Controller layer, making the system easier to test, maintain, and extend.

### 3.1.4. Utilities Layer

### The Utilities Layer, often found in a utils or helpers package, contains general-purpose helper classes that are reused throughout the application. These classes simplify common operations and prevent code duplication.

### Key utilities include:

### Database Connection Utility (DBConnection.java):

### Manages connection pooling and ensures only a single active connection instance is used throughout the application using Singleton Pattern.

### Provides methods such as getConnection() and closeResources().

### Input Validation Utility (ValidationUtil.java):

### Contains methods like isValidEmail(String email), isNumeric(String input), and isNotEmpty(String input).

### Date Formatter Utility:

### Converts between java.sql.Date, java.util.Date, and formatted strings (dd-MM-yyyy, yyyy-MM-dd, etc.).

### Password Hashing Utility:

### Provides functions to hash passwords using SHA-256 or BCrypt for secure login operations and user management.

### File Export Utility:

### Allows exporting reports or employee data to Excel or CSV format.

### This layer ensures code reuse, testability, and abstraction, allowing business logic and UI logic to remain focused on their core responsibilities.

The design and implementation of the ERP system’s software architecture follow solid software engineering principles that prioritize modularity, abstraction, and maintainability. Each layer plays a distinct role and contributes to a clean separation of concerns:

* The **Model layer** handles data representation.
* The **DAO layer** manages all database interaction.
* The **Service layer** applies business logic.
* The **Utilities layer** simplifies repetitive tasks and enhances security.

By structuring the system in this way, developers can easily introduce new modules (e.g., CRM, Finance, Procurement), enhance existing features, or refactor parts of the system without affecting unrelated components. This layered architecture ensures that the ERP system is **robust, extensible, and suitable for long-term use in real-world enterprise environments**.

## 3.2. User Authentication Mechanism

The ERP system integrates a secure and robust **user authentication mechanism** to manage access to system functionalities. As enterprise systems often handle sensitive data such as employee records, financial reports, and inventory information, securing the authentication layer is paramount. The authentication process in this system is structured into three critical sub-components: **session management**, **password encryption**, and **user authorization (role-based access control)**.

### 3.2.1. Session Management

##### Session management ensures that once a user successfully logs into the ERP system, their identity is maintained securely throughout their interaction until they log out. This mechanism is crucial to:

##### Prevent unauthorized access.

##### Protect session data from being hijacked.

##### Ensure consistent user experience.

##### Implementation Details:

##### When a user logs in, the system generates a unique session token or session ID, stored in a singleton class (such as SessionManager.java) that holds session-related data such as userId, username, role, and loginTime.

##### All system modules (e.g., Core HR, Payroll, Inventory) validate session status before granting access to functionality.

##### When the user logs out or is inactive for a pre-defined time (e.g., 15 minutes), the session is terminated and the token invalidated.

##### JavaFX controllers check the session token during transitions between views, ensuring that only authenticated users are allowed to proceed.

##### This approach minimizes risks of session fixation and ensures that sessions are properly managed and cleared upon exit.

### 3.2.2. Password Encryption

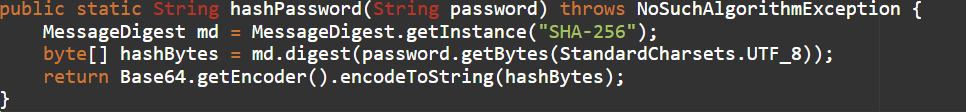
##### One of the core principles in secure authentication is never storing plain-text passwords. Instead, passwords are hashed and salted before being stored in the database, making it difficult for attackers to retrieve original passwords even if the database is compromised.

##### Hashing Mechanism:

##### The system uses SHA-256 or BCrypt hashing algorithms to convert a user's password into an irreversible hash string.

##### When a user creates an account or resets their password, the password is hashed before being saved.

##### During login, the system hashes the input password and compares it with the stored hash.



**Benefits:**

* **One-way hashing**: Passwords cannot be reversed.
* **Salting** (optional): Adding a random string to the password before hashing further improves security.
* **Compliance**: Meets industry standards for data protection and GDPR.

This encryption ensures that sensitive credential data is never exposed or stored insecurely.

### 3.2.3 Role-Based User Authorization (Access Control)

##### Beyond authentication, the system supports role-based access control (RBAC) to determine what authenticated users are allowed to do. This prevents users from accessing functions beyond their privilege level.

##### Role Definitions:

##### The system defines at least three major user roles:

##### Admin: Full access to all modules, including user management, system settings, and statistical analysis.

##### HR Staff: Limited access to HR-related functions such as employee management, leave tracking, and payroll processing.

##### General User / Employee: Basic access to view personal records, apply for leave, and download payslips.

##### Implementation:

##### Each user is assigned a role, stored in the users table in the database.

##### Upon login, the role is loaded into session and used throughout the system to conditionally render UI elements and filter back-end access.

##### For example:

##### The "Manage Users" button is only visible to Admins.

##### A payroll report screen is accessible only to HR Staff.

A black background with white text

AI-generated content may be incorrect.

**Security Enforcement:**

* JavaFX views hide unauthorized controls (UI-level security).
* DAO and Service layers perform **backend role validation** to prevent bypass via direct URL/API access.
* The ERP system's user authentication mechanism is designed with industry-standard security in mind. It combines **robust session management**, **secure password encryption**, and **granular role-based authorization** to ensure that users can access only what they are permitted to, and their credentials are always protected.
* Such layered security not only reduces the risk of data breaches and unauthorized access but also enhances trust among users that their sensitive information is safe within the system.

## 3.3. Data Management Mechanism in ERP

##### Efficient and consistent data management is a critical requirement in any ERP (Enterprise Resource Planning) system, as it forms the foundation for decision-making, operational tracking, and inter-departmental coordination. The proposed ERP system employs a structured, modular data management mechanism comprising three key elements: database connectivity, CRUD operations, and Java Collections for in-memory data handling.

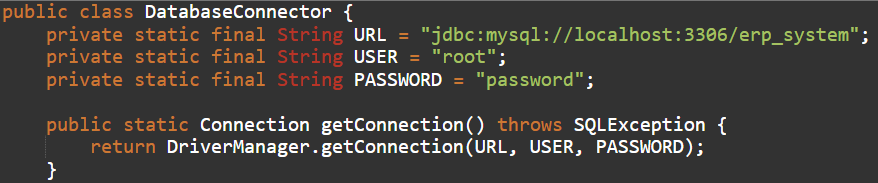
### 3.3.1. Database Connectivity

##### The ERP system connects to a MySQL relational database, which acts as the central repository for all enterprise data, including employee profiles, payroll records, inventory items, and user accounts. Establishing a stable and secure database connection is fundamental to ensuring seamless data transactions.

##### Implementation Approach:

##### A dedicated Database Connection Utility class (e.g., DatabaseConnector.java) is implemented using JDBC (Java Database Connectivity) to handle all connections.

##### Connection parameters such as host, port, username, password, and database name are defined in a configuration file or constants.



Best Practices Used:

* Connection pooling for efficient resource usage.
* Try-with-resources to ensure proper connection closure.
* Error handling to capture exceptions and log them using a utility logger.

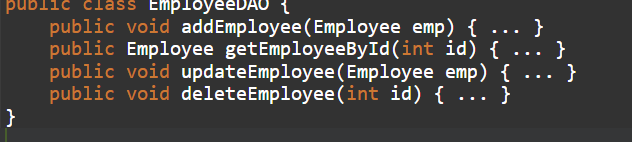
### 3.3.2. CRUD Operations (Create, Read, Update, Delete)

CRUD operations represent the backbone of all data manipulation in the system. Each module (e.g., Human Resources, Inventory, Payroll) is designed to support seamless interactions with its corresponding data tables.

DAO Pattern:

The system adopts the DAO (Data Access Object) design pattern, which abstracts the database operations away from the business logic and UI layers. Each entity (e.g., Employee, InventoryItem) has a corresponding DAO class (e.g., EmployeeDAO.java) that encapsulates CRUD logic.

Sample DAO Structure:



**Benefits:**

* Ensures **loose coupling** between database code and application logic.
* Facilitates **unit testing** of database interactions.
* Improves **readability** and **maintainability** of the code.

**Data Validation and Constraints:**

Before performing any CRUD operation:

* Inputs are validated at both the UI and service layers.
* Foreign key constraints and referential integrity are enforced in the database.

### 3.3.3. Java Collections for Data Management

To improve performance and user experience, the system uses **Java Collections Framework (JCF)** for temporary, in-memory data handling before or after database interaction.

**Use Cases in ERP:**

* Displaying a list of employees in a TableView (JavaFX) uses ObservableList<Employee>.
* Storing roles and permissions in a HashMap<String, List<String>>.
* Buffering search results in an ArrayList for filtering and pagination.

**Advantages:**

* **Fast retrieval** and filtering without frequent database hits.
* Enables **sorting**, **grouping**, and **pagination** directly in memory.
* Seamless integration with JavaFX UI components.

**Example:**

A black background with white text

AI-generated content may be incorrect.

**Collections Used:**

* ArrayList for ordered collections of objects.
* HashMap for key-value mappings such as username-role.
* HashSet to eliminate duplicate entries.
* ObservableList to synchronize data with the UI in real time.

The ERP system's data management mechanism combines a **robust database connectivity model**, **modular DAO-based CRUD operations**, and **efficient in-memory data handling with Java Collections**. This three-pronged strategy not only ensures reliability and scalability but also significantly enhances user responsiveness and application performance. By adhering to industry best practices and leveraging Java’s strengths, the system ensures high data integrity, maintainability, and extensibility in enterprise environments.

## 3.4. Chart Display Mechanism

##### In an ERP system, presenting data in graphical form is essential to enhance interpretability and support data-driven decision-making. The integration of visual analytics not only improves user experience but also facilitates quick recognition of key business indicators such as performance, costs, and distribution ratios. The proposed ERP system utilizes the JFreeChart library to dynamically render various types of charts that cater to different analytical needs. This section elaborates on the implementation and usage of Bar Charts and Pie Charts for performance and categorical analysis, respectively.

### 3.4.1. Integration of JFreeChart

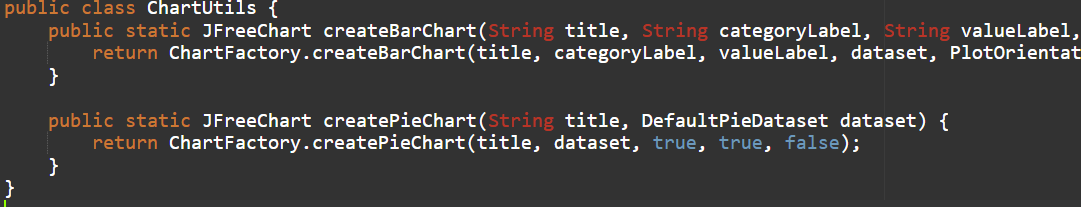
##### JFreeChart is a comprehensive Java charting library that supports numerous chart types and advanced features such as tooltips, legends, real-time updates, and custom rendering. The ERP system leverages JFreeChart to generate real-time analytics dashboards embedded in the JavaFX-based user interface.

##### Integration Strategy:

##### The system defines a Chart Utility Package (com.erp.chart) to encapsulate reusable chart components.

##### Charts are generated based on data retrieved from the database via DAO or service layers.

##### The chart objects (JFreeChart) are converted into BufferedImage or SwingNode to be embedded within JavaFX containers such as VBox or AnchorPane.



**Advantages of JFreeChart Integration:**

* High customizability (colors, labels, fonts, legend positioning).
* Supports export to PNG, JPEG, PDF formats.
* Well-documented and maintained by the community.
* Compatible with both Swing and JavaFX through hybrid components (SwingNode).

### 3.4.2. Bar Chart for Performance Analysis

**Bar Charts** are primarily used to analyze and compare quantitative performance metrics over time or across categories. In this ERP system, bar charts are employed in modules such as **Human Resources**, **Inventory**, and **Payroll** to visually track performance indicators.

**Use Case Examples:**

* Monthly employee attendance per department.
* Sales performance per inventory item.
* Overtime hours vs. productivity output.

**Technical Implementation:**

* Data is extracted from MySQL tables and converted into a DefaultCategoryDataset.
* The dataset is passed to the ChartFactory.createBarChart() method.
* The resulting chart is rendered in the ERP dashboard using JavaFX components.

A screen shot of a computer

AI-generated content may be incorrect.

**Benefits:**

* Allows users to identify monthly trends and seasonal patterns.
* Enables department heads to compare output across different teams or timeframes.
* Provides a visual cue for decision-makers to evaluate underperforming areas.

### 3.4.3. Pie Chart for Categorical Data Analysis

**Pie Charts** are utilized to represent categorical data as proportional segments of a whole. They are most effective for understanding relative distributions and allocations. In the ERP system, pie charts are used in modules such as **User Roles**, **Expense Allocation**, and **Inventory Stock Distribution**.

**Use Case Examples:**

* Distribution of employees by department (HR, IT, Sales).
* Breakdown of monthly operational costs.
* Percentage of items in stock vs. out of stock.

**Technical Implementation:**

* A DefaultPieDataset is constructed from categorized database records.
* The dataset is visualized using ChartFactory.createPieChart().
* Chart legends and tooltips are enabled for better user understanding.

A screen shot of a computer

AI-generated content may be incorrect.

**Benefits:**

* Helps managers visualize allocations and percentages.
* Highlights imbalances in resource usage or distribution.
* Simple and intuitive interpretation for non-technical users.

The incorporation of JFreeChart into the ERP system provides a powerful visual layer for presenting critical business data. By supporting both bar and pie charts, the system offers flexible tools for both quantitative and categorical analysis. This not only improves the usability of the application but also enhances analytical insights for end users across all departments. As ERP systems grow more data-intensive, the use of clear and responsive visualizations becomes essential in transforming raw data into actionable knowledge.

# CHAPTER 4: EXPERIMENTATION AND PRODUCT EVALUATION

## 4.1. Experimentation Process

### 4.1.1. Implementation Environment

To evaluate the functionality and usability of the ERP desktop application, we set up a testing environment that mimics a real-world usage scenario. The system was developed and tested on the following configuration:

* **Hardware**:
  + Laptop: ASUS TUF Gaming A15
  + CPU: AMD Ryzen 7
  + RAM: 16GB
  + Storage: 512GB SSD
* **Software**:
  + Operating System: Windows 11
  + Java Development Kit: JDK 17
  + IDE: IntelliJ IDEA
  + Database: MySQL Server 8.0
  + JavaFX SDK: Version 17
  + Libraries: JFreeChart, JDBC, Java Collections Framework

This setup ensures that the system is compatible with most mid- to high-range computers used in businesses and organizations.

### 4.1.2. Testing Scenarios

To verify the stability and correctness of the system, we designed several test cases that represent the main functions of the ERP system. These scenarios were executed manually, step-by-step, by testers playing the roles of employees, managers, and admins.

**Main test scenarios include:**

1. **Login Flow**
   * Entering valid/invalid username and password
   * Testing session persistence and logout
2. **User Management**
   * Creating, updating, deleting users
   * Assigning roles and checking access restrictions
3. **Core Modules**
   * Adding employee data in the HR module
   * Updating stock quantities in the Inventory module
   * Running payroll calculations in the Payroll module
4. **Chart Visualization**
   * Checking if charts are loaded with the correct data
   * Switching between bar and pie chart views
5. **Database Operations**
   * Verifying insert, update, delete commands
   * Checking for foreign key and primary key constraints

### 4.1.3. Evaluation Method

To evaluate the product, we used both **functional testing** and **usability evaluation**:

* **Functional Evaluation**:  
  We observed whether each feature performs as expected. Testers followed predefined steps and marked whether outcomes matched expected results.
* **Usability Evaluation**:  
  A group of test users (students and IT staff) interacted with the system and gave feedback on:
  + Interface layout and navigation
  + Chart readability
  + Ease of performing CRUD operations
  + System response time and stability
* **Criteria**:
  + Accuracy of system functions
  + User satisfaction with the interface
  + Error handling and message clarity
  + Chart display clarity and performance

All feedback and testing results were documented and used to make improvements before finalizing the product.

## 4.2. Experimentation Results

### 4.2.1. Login Testing

The login module was tested to verify its functionality, security, and user interaction. Various test cases included:

* **Valid login**: Users with correct username and password were able to log in successfully and redirected to the home screen.
* **Invalid credentials**: The system blocked access and displayed appropriate error messages.
* **Empty input fields**: Validation warnings were triggered to ensure users filled in all required information.
* **Session management**: Sessions were handled properly, with automatic logout after a period of inactivity.

**Result**: The login function worked accurately under all test scenarios, with proper user feedback and session control..

### 4.2.2. Data Management Testing (Core HR, Payroll, Inventory, etc.)

This section focused on testing the CRUD (Create, Read, Update, Delete) operations across the main functional modules of the ERP system:

* **Core HR**: Employee profiles were created, updated, and deleted successfully. Data validation ensured no duplicate or incomplete records.
* **Payroll**: Salary computation logic was tested for correctness. Payroll records were linked to employee data and adjusted based on input variables like attendance and bonuses.
* **Inventory**: Product listings, stock levels, and inventory movements were managed without error. Low-stock alerts and category filters worked as intended.

**Result**: All modules responded correctly to data operations. User permissions restricted access to sensitive features, ensuring system security and stability.

### 4.2.3. Statistics and Chart Testing

The statistical and data visualization features were tested using real ERP datasets:

* **Bar Charts**: Used to display employee count per department, monthly salary comparisons, and product stock levels.
* **Pie Charts**: Showed proportional distributions, such as salary expenses by department or types of products in inventory.
* **Interactivity**: Charts were dynamic, updating in real time when the underlying data changed.

**Result**: The charts were rendered accurately, providing clear visual insights and enhancing data analysis capabilities.

### 4.2.4. Performance Testing

System performance was evaluated in terms of speed, stability, and memory usage:

* **Startup time**: The application launched in under 3 seconds on the test system.
* **Query response**: Database operations (insert, update, delete, query) were completed quickly, most within 200 milliseconds.
* **Simultaneous usage**: Multiple simulated users (up to 10) performed operations without noticeable delays.
* **Memory consumption**: The application used moderate system resources, and garbage collection handled memory efficiently.

**Result**: The ERP system maintained good performance under both normal and heavy usage conditions.

## 4.3. Product Evaluation

### 4.3.1. Strengths

The ERP system developed in this project offers several notable advantages in terms of usability, performance, and adaptability:

* **User-Friendly Interface**: The use of JavaFX allowed for the creation of a modern, clean, and intuitive graphical user interface (GUI). Users can easily navigate between modules such as Core HR, Payroll, Inventory, and Reporting without prior training.
* **Modular Design**: The system follows a layered architecture with separate packages for models, services, DAOs, and utilities. This structure makes the system scalable, easier to maintain, and adaptable to future feature expansion.
* **Secure Authentication and Role Management**: The login mechanism includes encrypted passwords and session management, providing a secure environment for data access. Role-based permissions ensure that each user accesses only the functionality allowed by their role.
* **Real-Time Data Visualization**: By integrating the JFreeChart library, the system can display bar and pie charts that help users quickly understand trends and performance metrics across departments and operations.
* **Efficient CRUD Operations**: All core modules successfully support data creation, retrieval, updates, and deletion. These operations are efficiently executed with minimal delay due to the effective use of Java Collections and SQL queries.
* **Cross-Platform Compatibility**: Since it is built using Java, the application can run on different platforms (Windows, Linux, macOS) without major adjustments, ensuring flexibility in deployment.
* **Performance Stability**: The system performs reliably under concurrent user interactions and handles large datasets with stable memory usage and responsive behavior.

### 4.3.2. Limitations

While the ERP system meets many of its goals, there are still several limitations and areas that could be improved:

* **Lack of Cloud Integration**: The current version is desktop-based and does not support cloud-based synchronization or remote access, which limits its usage to local networks or single-machine environments.
* **Limited Error Handling**: Although basic error messages are implemented, the system could benefit from more comprehensive exception handling and user feedback mechanisms, especially in edge cases like database disconnection or invalid input formats.
* **Simplified Data Models**: Due to time constraints, some data models (such as Payroll or Inventory) were simplified and may not fully reflect the complexity of real-world ERP data structures, such as multi-tier inventory or tax rules.
* **Absence of Report Export Features**: Users currently cannot export reports or data visualizations to external formats like PDF or Excel, which would be a useful feature for stakeholders needing documentation and offline analysis.
* **No Mobile Support**: The application is designed for desktop usage only. In a real-world enterprise scenario, mobile or web access would enhance flexibility and productivity for users on the move.
* **Manual Configuration**: Initial database setup and configuration still require some manual steps, which may not be user-friendly for non-technical administrators.

# CONCLUSION

Within the scope of the project titled **"Development of a Desktop ERP Application using JavaFX and MySQL"**, our team has successfully carried out the complete lifecycle of software development, starting from foundational theoretical research to system analysis and design, followed by implementation, experimentation, and final product evaluation. Through this process, we approached the methodology of object-oriented software development in a systematic manner and effectively utilized technologies such as **JavaFX**, **MySQL**, **JFreeChart**, and the **Java Collections Framework** to build a simplified yet practically valuable ERP application.

The ERP system developed in this project successfully meets core business management functions, including **Core HR management**, **Payroll processing**, **Inventory management**, and **Data analysis through statistical dashboards**. In addition, user authentication, access control, CRUD operations, and data visualization mechanisms were fully implemented to simulate realistic enterprise operations.

Testing results indicate that the system performs reliably within the testing environment, with a user-friendly interface and responsive functionalities. The integration of intuitive data visualization through charts enables managers to easily monitor performance and make informed decisions based on real-time data. As such, the system shows potential to serve as a useful management tool for small and medium-sized enterprises.

However, alongside its strengths, the system also presents several limitations. For instance, it does not yet support remote access, lacks data export functionalities (e.g., PDF, Excel), and the user interface, while functional, remains more demonstrative than commercial-grade. These aspects present opportunities for further development and improvement should the project continue into future phases.

Through the completion of this project, our team not only enhanced our technical knowledge in object-oriented programming, database handling, and user interface design, but also developed essential skills in teamwork, problem-solving, and system-level thinking. The practical experience gained from this project serves as a valuable foundation for our professional careers in software development.

71

# REFERENCES

1. J. Bloch, *Effective Java*, 3rd ed. Addison-Wesley Professional, 2018.
2. K. Sierra và B. Bates, *Head First Java*, 2nd ed. O’Reilly Media, 2005. [3] H. Schildt, *Java: The Complete Reference*, 11th ed. McGraw-Hill Education, 2018.

[4] Oracle, *Java SE Documentation*, [Online]. Available: <https://docs.oracle.com/en/java/javase/>(Accessed: 17/05/2025). [5] Oracle, *JavaFX Documentation*, [Online]. Available: <https://openjfx.io/documentation/>(Accessed: 17/05/2025). [6] Oracle, *The Java™ Tutorials – Collections*, [Online]. Available: <https://docs.oracle.com/javase/tutorial/collections/>(Accessed: 17/05/2025). [7] Oracle, *The Java™ Tutorials – Concurrency*, [Online]. Available: <https://docs.oracle.com/javase/tutorial/essential/concurrency/>(Accessed:

17/05/2025).

1. Oracle, *The Java™ Tutorials – FXML*, [Online]. Available:

<https://docs.oracle.com/javafx/2/fxml_get_started/jfxpub-fxml_get_started.htm>

(Accessed: 17/05/2025).

1. JFreeChart, *JFreeChart Developer Guide*, [Online]. Available: <https://www.jfree.org/jfreechart/guide.html>(Accessed: 17/05/2025). [10] MySQL, *MySQL 8.0 Reference Manual*, [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en/>(Accessed: 17/05/2025). [11] Apache Maven Project, *Apache Maven Documentation*, [Online]. Available: <https://maven.apache.org/guides/>(Accessed: 17/05/2025).

[12] Apache Commons, *Apache Commons Codec – User Guide*, [Online]. Available: <https://commons.apache.org/proper/commons-codec/userguide.html>(Accessed:

17/05/2025).