Phong and
Toan

# ERP System Java Desktop Application: Project Summary


## Presentation:Hồ Lâm Hải Phong
## Nguyễn Đức Toàn

## Instructor: Dr. Le Ngoc Hieu

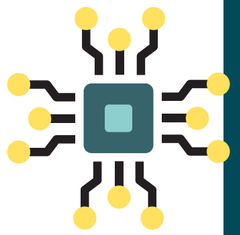


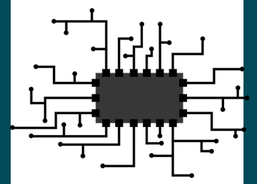
Visit Our Website
**TPGreat.com**

# STACK OVERVIEW

## OOP with Java

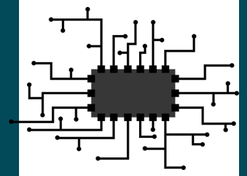
- Encapsulation: Look for classes with private attributes and public getter/setter methods.
- Inheritance: Identify classes using the extends keyword and interfaces using the implements keyword.
- Polymorphism: Find instances of interfaces with multiple implementing classes, method overloading (same name, different parameters), or method overriding (same name and parameters in a subclass).
- Abstraction: Locate abstract classes and interfaces focusing on essential features.

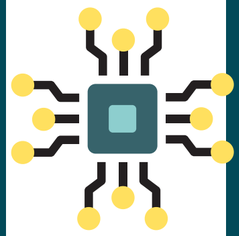




## JavaFX

- FXML: Look for files with the .fxml extension defining UI structure.
- MVC Pattern: Identify potential Model (data classes), View (FXML or UI code), and Controller (Java classes handling logic) components.
- Rich Controls: Find usage of JavaFX UI components like Button, Label, TextField, TableView, etc.
- CSS Styling: Look for .css files or the style attribute used for UI customization.
- Event Handling: Find code using methods like setOnAction, setOnMouseClicked, etc., to handle user interactions.





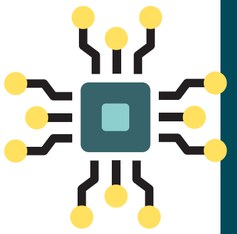
## MySQL Database

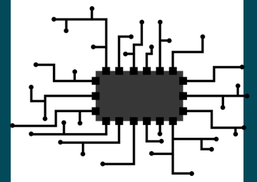
- FXML: Look for files with the .fxml extension defining UI structure.
- MVC Pattern: Identify potential Model (data classes), View (FXML or UI code), and Controller (Java classes handling logic) components.
- Rich Controls: Find usage of JavaFX UI components like Button, Label, TextField, TableView, etc.
- CSS Styling: Look for .css files or the style attribute used for UI customization.
- Event Handling: Find code using methods like setOnAction, setOnMouseClicked, etc., to handle user interactions.

# STACK OVERVIEW

## JavaFX SDK

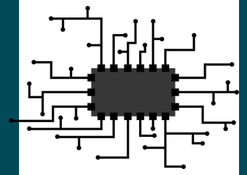
- Core UI Controls: I will look for the usage of fundamental UI elements like Button, Label, TextField, TableView, ScrollPane.
- javafx.fxml: I will check if you are using the javafx.fxml library to load FXML layout files.
- Layout Panes: I will identify if you are employing layout panes such as BorderPane, VBox, HBox, StackPane to organize UI components.
- javafx.scene.media: I will see if the javafx.scene.media library is used for video playback functionalities.

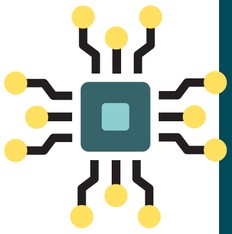




## MySQL Connect

- SQL Query Execution: I will search for code that executes SQL queries and retrieves results.
- Data Updates: I will look for operations like INSERT, UPDATE, and DELETE for managing data.
- Used within: I will examine if these operations are contained within a utility class potentially named similarly to DBConnection.java.





## Java Collections

- List: I will check for the use of List implementations such as ArrayList<Product> or ObservableList<CartItem>.
- Map: I will look for the use of Map implementations like HashMap for caching or lookups.
- Data Flow: I will analyze how collections are used to store and manipulate data within your code.

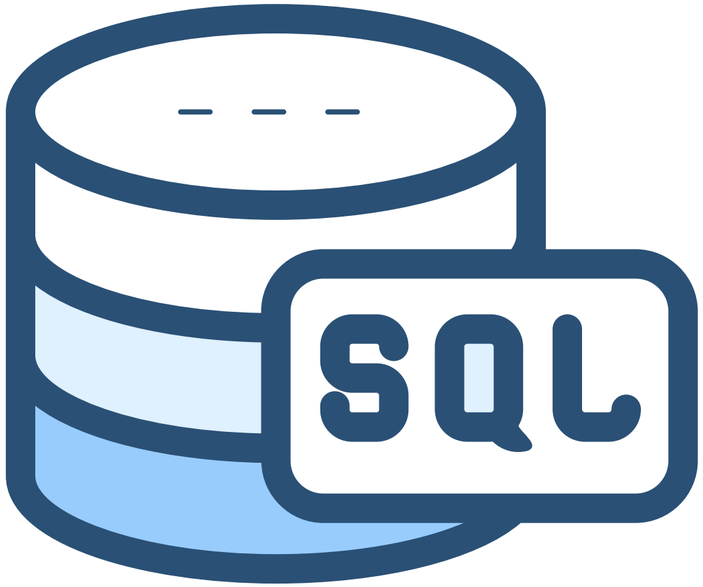
# Problem and Research

## Current Application Landscape

The current landscape involves managing various aspects of a business through separate interfaces or potentially manual processes. There's a need for a centralized view to understand key metrics and data points across different departments and functionalities.

## Existing Solutions (The Gap to Address)

While there might be existing ways to access the underlying data (e.g., individual database queries, separate reports), a unified, interactive dashboard application can offer a more efficient and user-friendly experience. It allows for quick visualization and analysis of critical information without requiring deep technical knowledge or switching between multiple systems.





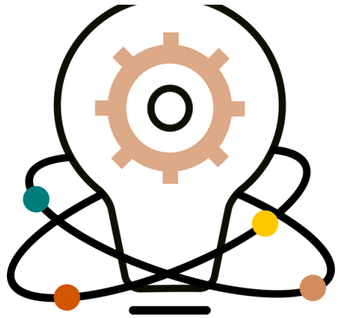

## The Desktop Dashboard Solution:

This Java app provides a business data overview via charts, leveraging:

- OOP Java: Clear code structure (model, DAO) for maintainability.
- GUI (Swing, JFreeChart): Intuitive interface with tabs and charts.
- Data Integration: Connects various sources (employees, orders, inventory, etc.) via DAO.
- Customization & UX: Allows tab selection, chart details, refresh, and export.


## Learning Opportunity:

Developing this app offers practical experience in OOP, Swing/JFreeChart GUI, and data handling via DAO for business data visualization.



controller: Handles logic between UI (View) and data (Model).


model: Represents application data objects.

DAO (Data Access Object): Manages data interaction (e.g., database).

# System architecture

view: Contains user interface components.

utils (Utilities): Contains common helper functions.

Phong and Toan

# SUBSTANCE




Visit Our Website
TPGreat.com

# Introduction


# Reason for Choosing the Topic

- In the era of Industry 4.0, the application of IT in business management is a mandatory requirement.
- The ERP system is one of the most effective and comprehensive management systems.
- However, current ERP systems (e.g., SAP, Oracle) come with high costs and complex processes.

- This creates a barrier for small and medium-sized enterprises (SMEs).
- The project aims to develop a Java Desktop ERP application to provide a feasible solution for SMEs.

# Research Objectives



- Study the theoretical foundation of ERP systems.
- Survey the actual needs of small enterprises.
- Design the software architecture.
- Select appropriate technologies and develop the application.
- Conduct system testing and complete documentation.

# Research Scope

- The system only implements basic ERP functions.
- The application is developed as a standalone desktop software.
- Target users: Small and medium-sized enterprises (SMEs).


# II.Theoretical Background and Research Status

01 Theoretical Background

02 Research Status

```
erp_system/
├── src/main/java/
│   ├── DAO/
│   ├── model/
│   ├── view/
│   ├── controller/
│   └── utils/
├── erp_system.sql
├── README.md
└── ...
```

# Theoretical Background

ERP is a comprehensive software solution that integrates core business processes.

The goal is to establish a single centralized data source.

OOP is an efficient way to model business processes in ERP development

JavaFX is a modern platform for developing rich GUIs for Java desktop applications.

MySQL and SQL are used to define, query, and manage data in relational databases.

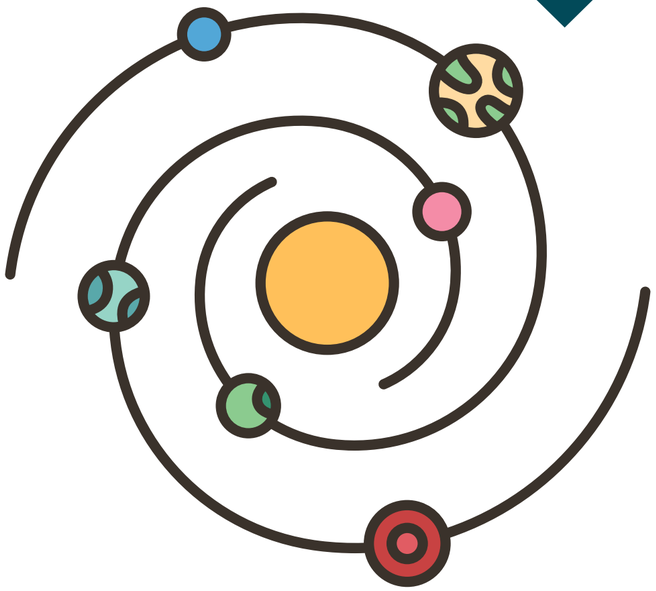
The Java Collections Framework provides powerful data structures and algorithms

JFreeChart is a Java library for creating various types of charts for data visualization.

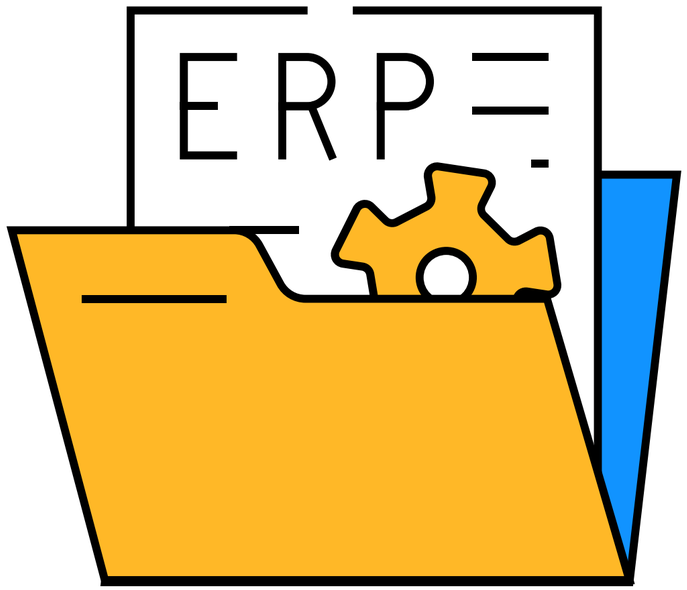
# Research Status


Modern ERP technology trends: Cloud ERP, Modular ERP, Mobile ERP, AI in ERP, Open-source ERP.

Analysis of similar ERP applications: ERPNext, Odoo, Dolibarr, SAP Business One, Oracle NetSuite.



# III.Proposed Solution and System Design Analysis


01 Development of a modular ERP desktop application using JavaFX, MySQL, and JFreeChart

02 Management of key business processes: Inventory control, sales management, HR, financial reporting

03 Emphasis on simplicity, cost-effectiveness, and user-friendliness.

04 General Model: Multi-layered architecture, MVC paradigm.

05 Functional Requirements: User Management, Inventory Management, Sales and Invoice Processing, Human Resources Management, Financial and Statistical Reporting, Data Backup and Recovery.

06 Non-functional Requirements: Usability, Performance, Scalability, Security, Reliability, Maintainability, Portability

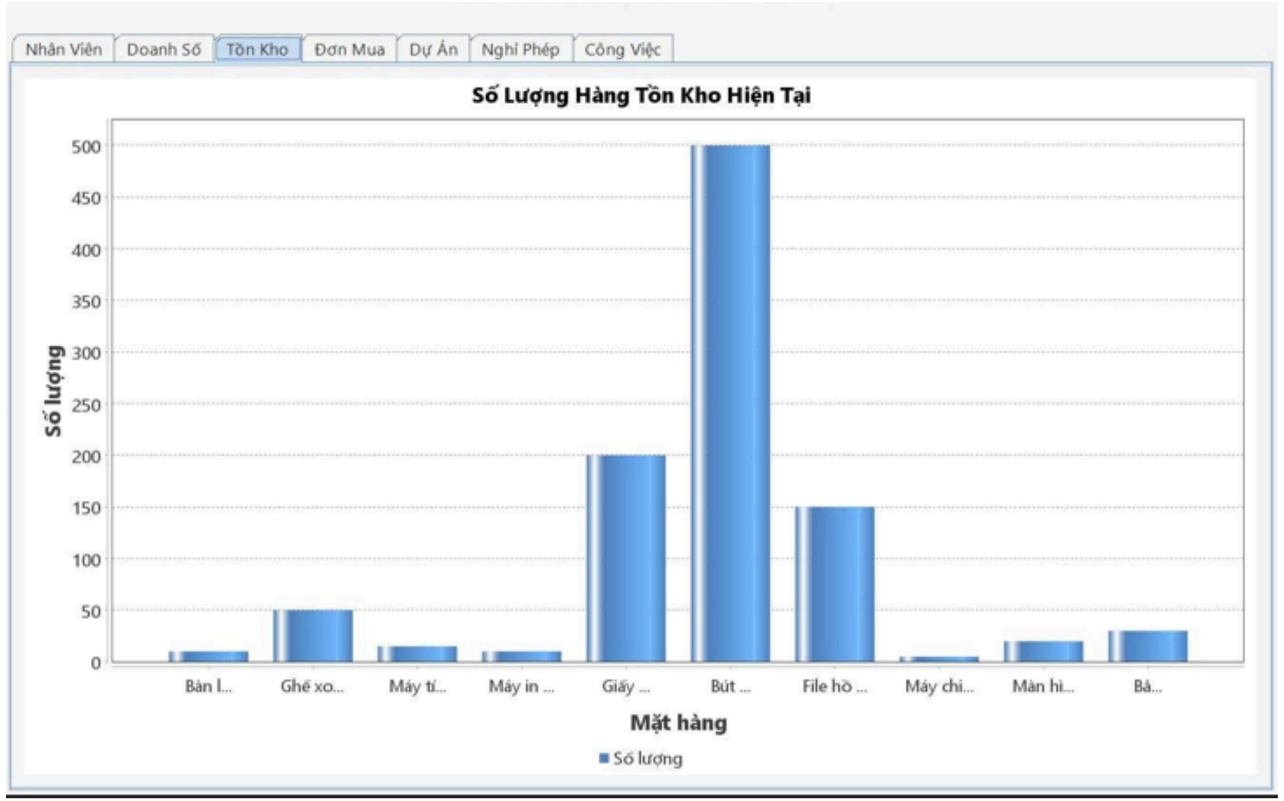
# IV.System Operation Mechanism

**Software Package and Class Layer Implementation:**

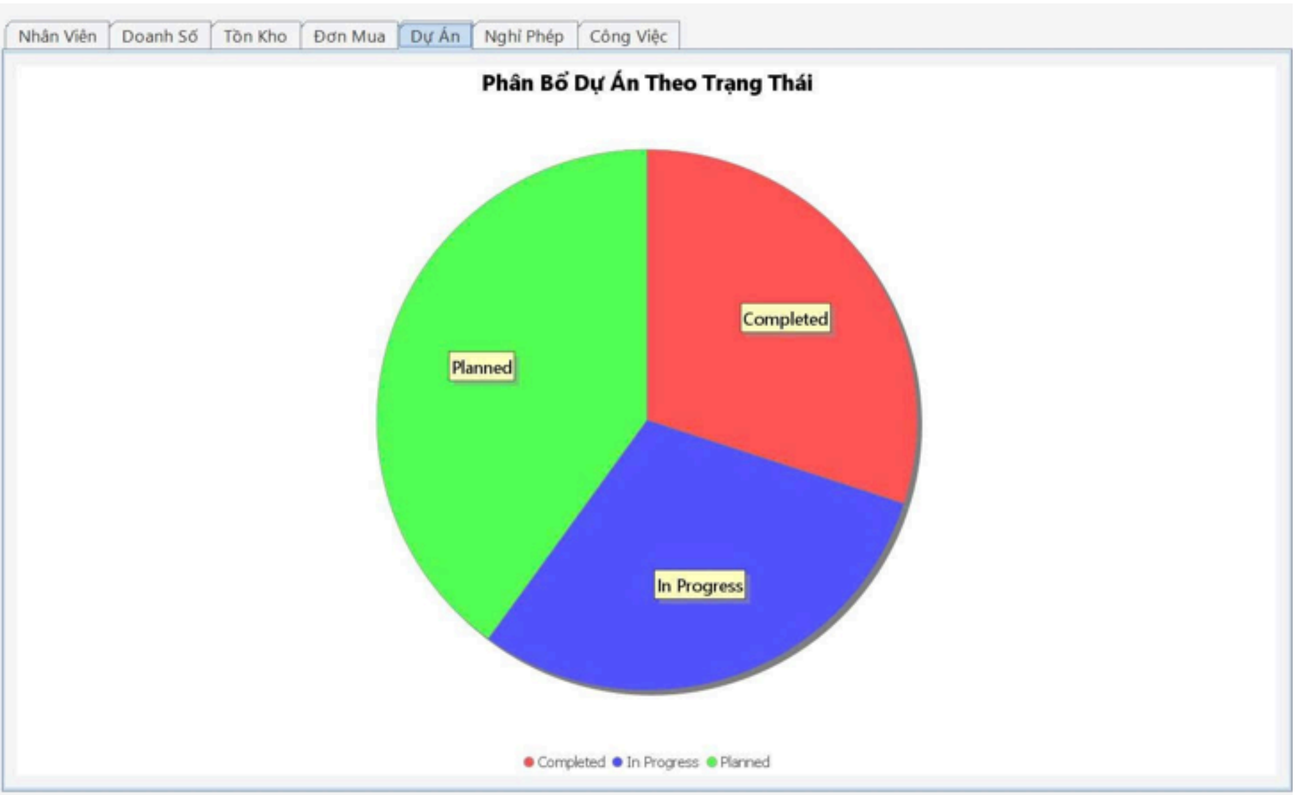
Class Layers:
Controller
Model (entity classes),
DAO (data access objects),
Service (business logic),
Utilities (helper classes).

**User Authentication Mechanism**

Session Management.
Password Encryption (SHA-256, BCrypt).
Role-Based User Authorization (RBAC).



# IV.System Operation Mechanism




## Data Management Mechanism

Database Connection (MySQL, JDBC).
CRUD Operations (Create, Read, Update, Delete).
Java Collections in Data Management (ArrayList, HashMap, etc.).

## Chart Display Mechanism

JFreeChart Integration (SwingNode).
Bar Chart.
Pie Chart.
Line Chart.

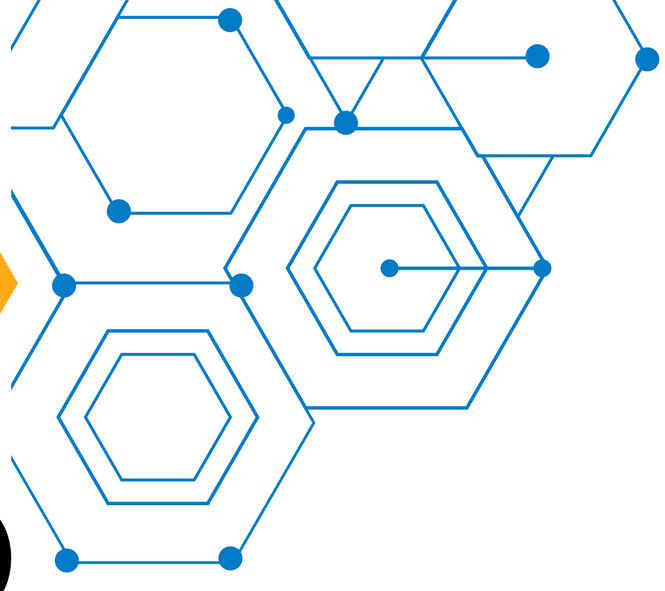
# V.Experimentation and Product Evaluation

- Experimentation Process
- Test Scenarios (for each module)
- Evaluation Methods
- Experiment Results
- Product Evaluation

# Experimentation Process

**Hardware: Personal computer (CPU, RAM, Storage depending on data size).**

Software:Operating System: Windows, macOS, Linux.
Java Development Kit (JDK) appropriate version.
NetBeans/IntelliJ IDEA/Eclipse (or other IDE) to run the application.
MySQL (or other database management system if used) for data storage and retrieval.
JFreeChart library (integrated into the project).




# Test Scenarios (for each module)

- Employee" Module:Verify the display of the employee distribution pie chart by department with different data (with employees, without employees, multiple departments).
- Check the detailed information upon interaction (click) with chart sections.
- Test the data refresh functionality.

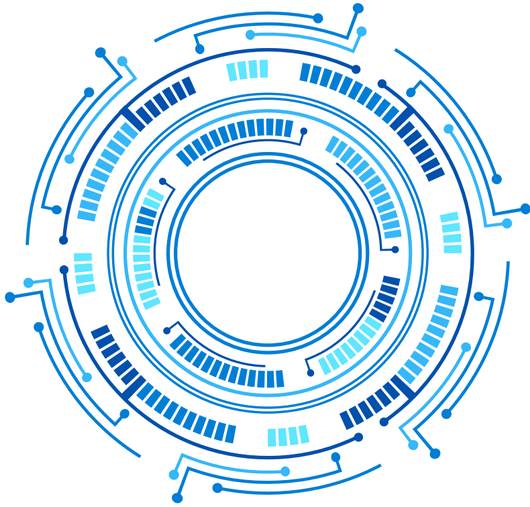
- Sales" Module:Verify the display of the total sales bar chart based on time filters ("This Month", "This Quarter", "This Year", "All") with varying sales data.
- Check the detailed information upon interaction (click) with chart bars.
- Test the data refresh functionality and filter changes.

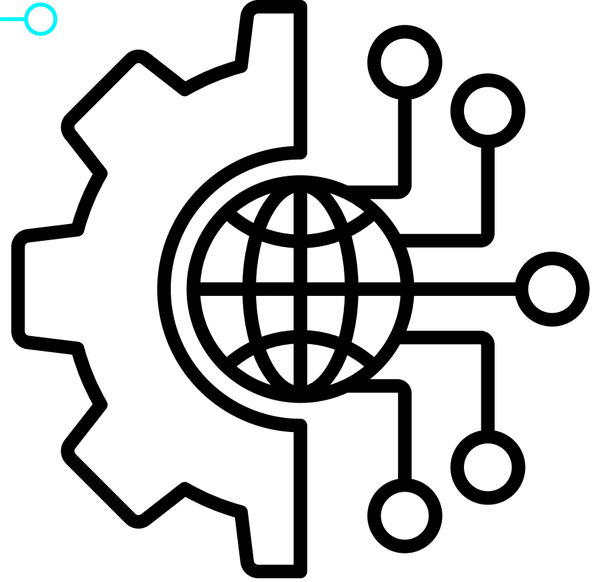
- Inventory" Module:Verify the display of the inventory quantity bar chart with different data (with products, without products, varying quantities).
- Check the detailed information upon interaction (click) with chart bars.
- Test the data refresh functionality.

- Purchase Order", "Project", "Leave Request", "Task" Modules: (Similar to the "Employee" module - verify pie/bar chart display with different data, detailed information on interaction, and refres

- Dashboard Customization" Module:Verify the ability to select/deselect tabs for display.
- Test the saving and loading of customization settings.
- Verify the dashboard redisplay according to saved settings.

- Chart Export" Module:Verify the ability to export the current tab's chart to a PNG file.
- Check the file name and format after exporting.





## Product Evaluation

- Correctness: Compare the data displayed on charts and detailed information with the actual data from the database (if applicable).
- Completeness: Ensure all tabs and functionalities (filters, refresh, customization, export) operate as expected.
- Stability: Test that the application does not crash or encounter unexpected errors during various operations.
- UI/UX: Evaluate the intuitiveness and ease of use of the interface, the clarity of the charts and information.
- Code Quality: Review the code structure (package division, variable/method naming), readability, and maintainability (OOP principles, DAO usage).
- Documentation: Assess the clarity and completeness of comments within the code (if any).

## Experiment Results

- Login Testing: (If login functionality exists) Verify successful/failed login attempts with different accounts.
- Data Management Testing: Verify the accurate display of data from DAOs onto the charts.
- Statistics and Chart Testing: Evaluate the accuracy of values on the charts and detailed information upon interaction.
- Performance Testing: (Basic level) Observe the data loading and chart rendering times, especially with larger datasets.
- Customization Testing: Verify the outcome of customizing tab display.
- Export Testing: Check the generated PNG image file after chart export.

# Product Evaluation

## Advantages

Core functionalities (displaying charts for employees, sales, inventory, purchase orders, projects, leave requests, tasks) are complete.
Relatively clear architecture with separation between View (DashboardFrame), Model (model), and Data Access (DAO).
Application of OOP principles (encapsulation in model, abstraction in DAO).
Intuitive and user-friendly interface with tabs and charts.
Integration of the powerful JFreeChart library for data visualization.
Basic tab customization and chart export functionalities are implemented.
Basic error handling (e.g., displaying error messages when data loading fails).

## Limitations

Performance may degrade when handling large datasets (further testing required).
The aesthetics of the Swing interface might not be modern (can be improved using different look and feels or more advanced UI libraries).
Lack of automated testing (JUnit) to ensure stability and ease of maintenance.
Feature extensibility might need consideration for easier addition of new chart types or functionalities.
User session management (if applicable) might need more explicit implementation.

# VI.Conclusion

Based on testing and evaluation, this Swing application dashboard has achieved its goal bases in providing a visual view of business information data through charts. Clean application architecture, OOP principles contribution and good library integration JFreeChart. The ability to customize the display and appearance of the base chart is also developed, providing a better user experience.

However, there are still some limitations that need to be considered in future development versions. Performance when handling large amounts of data needs to be optimized to ensure a smooth experience. The user interface, while intuitive, could be improved aesthetically to give it a more modern feel. Adding automated testing will be important to ensure stability and ease of maintenance of the application as changes or feature extensions occur. Finally, user session management needs to be implemented explicitly and securely if the application is intended for a multi-user environment.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam diam lorem, tempor vitae commodo a, auctor a tortor. Vivamus suscipit sit amet nibh id venenatis. Curabitur sollicitudin, nisl at commodo auctor

# Contact Us

0393-975-xxx

www.TPGreat..com

TPGreatÂgmail.com

ThaoDien,tp THU DUC

Phong and
Toan

# Thank You

## For Your Attention

Visit Our Website
**TPGreat.com**