

**Faculty of Computer Science and Engineering**

****PROJECT REPORT**  
**Object-Oriented Programming (OOP) Course****

**Academic Year 2024–2025**

**PROJECT TITLE:**

**SPY AGENCY 2024**

**Student:**

Đường Tri Nhân - 97482403193

**Class:** 24CNTT

**Lecturer:**

Dr.Lê Ngọc Hiếu

May, Academic Year 2024–2025

Table of Contents

[Task Assignment Table 4](#_Toc30521)

[List of Abbreviations 5](#_Toc5338)

[Introduction 6](#_Toc30236)

[1. Reason for choosing the topic. 6](#_Toc14676)

[2. Research Objectives. 7](#_Toc14951)

[3. Research Goals 7](#_Toc6123)

[4. General Description of the Developed System 7](#_Toc23126)

[CONTENT 9](#_Toc19104)

[Chapter 1: Theoretical Basis and Current Research Status 9](#_Toc14804)

[1.1. Scientific Basis 9](#_Toc14321)

[1.1.1. Overview of Object-Oriented Programming 9](#_Toc31481)

[1.1.2. Java Swing Technology in Desktop Application Development 10](#_Toc10861)

[1.1.3. MySQL Database Management System and SQL Language 10](#_Toc11635)

[1.1.4. Data Structures and Java Collections Framework 11](#_Toc23408)

[1.2. Current Research Status 12](#_Toc10946)

[1.2.1. Modern Technology Status 12](#_Toc30111)

[1.2.2. Analysis of Similar Management and Data Analysis Applications and Solutions 13](#_Toc23677)

[Chapter 2: Proposed Solution and Application Design Analysis 15](#_Toc15284)

[2.1. Proposed Solution 15](#_Toc25610)

[2.1.1. General Architecture 15](#_Toc21702)

[2.1.2. Functional Requirements 16](#_Toc21062)

[1. User Management and Authentication 16](#_Toc986)

[2. Agent and Mission Data Management 17](#_Toc25942)

[3. Dashboard and Data Visualization 17](#_Toc16617)

[4. Additional Features 18](#_Toc17936)

[2.1.3. Non-Functional Requirements 18](#_Toc26409)

[1. Security: 18](#_Toc18848)

[2. Usability: 18](#_Toc2724)

[3. Reliability & Robustness: 18](#_Toc8528)

[4. Maintainability & Extensibility: 19](#_Toc22356)

[5. Performance: 19](#_Toc29087)

[2.2. System and User Interface Design Analysis 19](#_Toc11877)

[2.2.1.System Architecture 19](#_Toc21263)

[2.2.2 User Interface Design Analysis 21](#_Toc6735)

[2.3 User Interface Information. 25](#_Toc4681)

[2.4 MySQL Database Structure 28](#_Toc14556)

[2.4.1 Table Description 28](#_Toc29328)

[2.4.2. Constraints and Relationships 31](#_Toc29280)

[CHƯƠNG 3: SYSTEM OPERATION MECHANISM 33](#_Toc3223)

[3.1 Overall Structure 33](#_Toc20396)

[3.2 Model Layer 34](#_Toc30003)

[3.3 DAO Layer (Data Access Object) 34](#_Toc3193)

[3.4 Controller Layer 35](#_Toc29320)

[3.5 View Layer (Java Swing) 35](#_Toc12903)

[3.6 Chart and ChartDAO Layers 36](#_Toc31641)

[3.7 Connection Configuration (Config) 36](#_Toc10437)

[3.8 User Authentication Handling 37](#_Toc23912)

[CHAPTER 4: EXPERIMENTS AND EVALUATION 37](#_Toc18361)

[4.1 Testing Environment and Procedure 37](#_Toc31241)

[4.2 Testing Results 38](#_Toc12673)

[4.3 Product Evaluation 39](#_Toc31058)

[Conclusion 41](#_Toc2787)

[1. Successfully implemented User Management and Authentication 41](#_Toc16402)

[2. Developed a comprehensive module for managing agents and related data entities 42](#_Toc13325)

[3. Implemented data visualization features using charts 42](#_Toc27124)

[4. Effectively applied Object-Oriented Programming (OOP) techniques and Java technologies 42](#_Toc28011)

[5. Completed a well-structured project and documentation system 42](#_Toc9074)

[References 44](#_Toc28739)

**Task Assignment Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| | **No.** | | --- | | **Full Name** | **Task Description** | **Notes** |
| **1** | **Đường Tri Nhân 97482403193** | In charge of the entire project |  |

**List of Abbreviations**

|  |  |
| --- | --- |
| **Abbreviation** | **Explanation** |
| **DAO** | **Data Access Object** |
| **CRUD** | **Create, Read, Update, Delete** |
| **DB** | **Date Base** |
| **OOP** | **Object-Oriented Programming** |
| **UI** | **User Interface** |
| **MVC** | **Model - View - Controller** |
| **PK** | **Primary Key** |
| **JDBC** | **Java DataBase Connectivity** |

**Introduction**

1. **Reason for choosing the topic.**

In the context of the rapid development of information technology, the application of software for managing, storing, and analyzing data has become an inevitable trend across many fields, especially in professions that require high security, accuracy, and fast processing speeds. The intelligence sector is a typical example, where managing a large volume of data about agents, missions, skills, organizations, and complex relationships has always been a challenging and specialized problem.

However, in reality, there are currently few simple and user-friendly spy data management applications that still fulfill all the necessary functions for managing, reporting, and visualizing data directly on a desktop interface. Most existing systems tend to be complex, difficult to access, or lack visualization tools that help users quickly and effectively monitor and evaluate information.

Therefore, our team chose to undertake the project **“SpyAgency2024”** with the aim of developing a desktop software for managing intelligence data featuring a user-friendly and intuitive interface. The application integrates full functionalities to manage information about agents, missions, skills, teams, and the relationships among components within the intelligence organization. The software not only stores and retrieves information but also supports **statistical analysis and data visualization** through charts using the **JFreeChart library**, enabling users to easily monitor figures, trends, and the operational status of agents as well as the entire intelligence system.

In particular, the project utilizes **Java Swing** — a powerful and popular Java GUI library — to build the software interface, combined with **MySQL** for data storage and **Object-Oriented Programming (OOP)** principles to design the program structure into separate layers such as **Model, DAO, Controller, Chart, ChartDAO, View, Config, and Icon**. Through this, the project helps the student team practice designing and building software following a multi-layer architecture, ensuring scalability, maintainability, and ease of future upgrades.

Carrying out this project not only helps our team consolidate and enhance our knowledge of Java programming, skills in building desktop interfaces using Java Swing, and working with MySQL databases, but also enables us to master techniques for data processing and visualization through charts, serving practical data analysis and management tasks. At the same time, the project’s product can be applied in real-world scenarios or serve as a reference foundation for developing data management software for organizations, businesses, or other fields with similar needs.

For these reasons, we believe that the project **“SpyAgency2024”** is a practical choice, suitable for the course requirements and has high practical application value.

1. **Research Objectives.**

The project **“SpyAgency2024”** is carried out with the purpose of developing a desktop application using Java Swing to **scientifically, systematically, and efficiently manage,** store, and analyze intelligence data. The software simplifies the process of managing information about agents, missions, skills, and intelligence organizations, while also supporting users in performing statistical operations, searching, and visualizing data quickly and intuitively through vivid charts.

Through this project, the student team aims to **apply the theoretical knowledge they have learned about Object-Oriented Programming (OOP), Java Swing, and MySQL database management to a practical problem**, contributing to improving their software design and development skills, as well as honing teamwork, organization, and the implementation of a complete software project.

1. **Research Goals**

**- Design and build a database system** to store information about agents, missions, skills, organizations, and their relationships in a logical, tight, and scalable manner.

- **Develop a user-friendly desktop graphical interface** using Java Swing, ensuring ease of use, intuitiveness, and full support for essential data management functions.

- **Construct a user access control system** with secure login functionality using SHA-256 hashed passwords.

- **Implement complete data management functions** (CRUD: Create – Read – Update – Delete – Search) for tables containing information on agents, missions, skills, organizations, teams, and related linkage tables.

- **Integrate data statistics and visualization features using the JFreeChart library.**

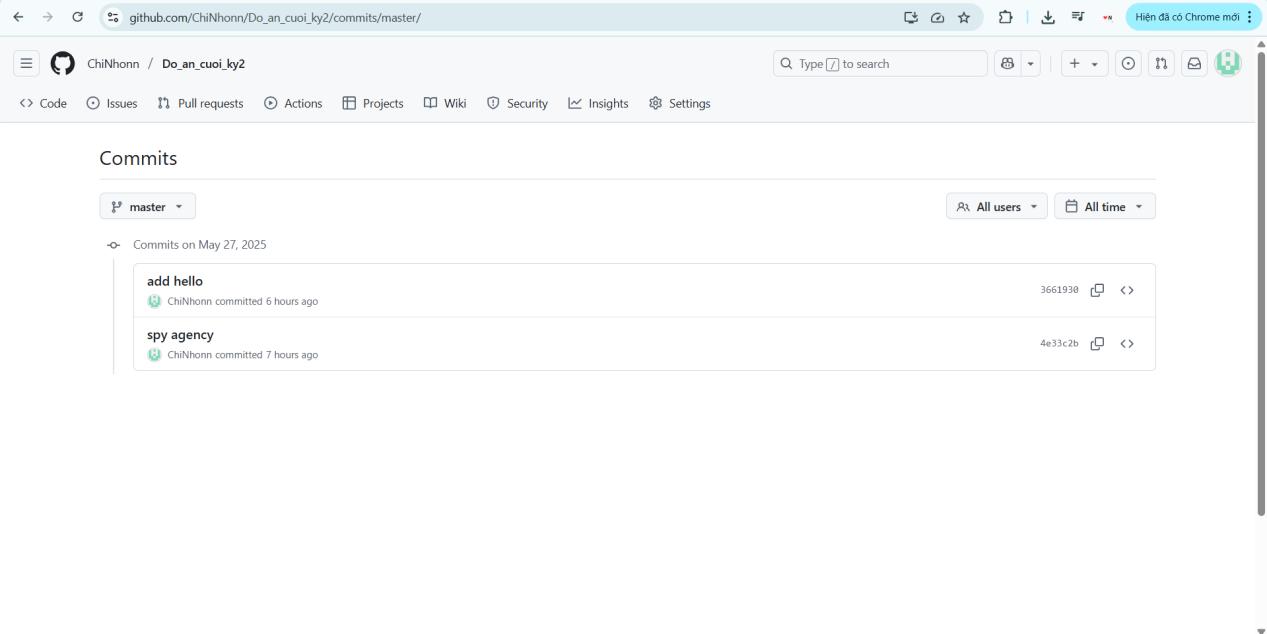
**- Adhere to Object-Oriented Programming (OOP)** principles and apply a proper multi-layered software architecture model: **Model – DAO – Controller – Chart – ChartDAO – View – Config – Icon**, to ensure maintainability, scalability, and reusability of the system.

1. **General Description of the Developed System**

The **Spy Agency Management** system is developed as a desktop application using the Java programming language, with a graphical user interface **(GUI)** designed in **Java Swing** and data stored in a **MySQL** database.  
This system enables effective and intuitive management of data related to agents, missions, teams, skills, languages, affiliated units, and security levels.

Data visualization functions through **statistical charts using the JFreeChart library** help the leadership quickly grasp the status of task assignments, the number of agents, security levels, and personnel distribution in intelligence operations, thereby minimizing errors, preventing fraud, and improving management efficiency.

**My Github:**



**CONTENT**

**Chapter 1: Theoretical Basis and Current Research Status**

### ****1.1.**** Scientific Basis

#### ****1.1.1.**** Overview of Object-Oriented Programming

**Object-Oriented Programming (OOP)** is a programming paradigm based on the concept of "**objects**," where these objects contain data (called **attributes**) and actions that manipulate the data (called **methods**). OOP offers a visual, structured, and manageable approach, especially suitable for systems with complex data and multiple interacting objects.

During the development of the **SpyAgency2024** project, the team fully applied the fundamental principles of object-oriented programming to design and organize the source code logically, ensuring scalability and maintainability. Specifically:

****+** Encapsulation:**  
Ensures that important object data is protected from unwanted external influences. **Model classes** such as Agent, Mission, Skill, Team, etc., declare their attributes as **private** and provide public **getter/setter** methods to safely access the data.

****+** Inheritance**:****  
Allows child classes to inherit common attributes and methods from parent classes, reducing code duplication and establishing a clear hierarchical relationship between objects. For example, the project may build a BaseDAO class containing basic database operation methods, which AgentDAO, MissionDAO, and other DAO classes inherit for reuse and extension.

****+** Polymorphism**:****  
Enables different objects to use a common interface but implement it in their own way. This feature is demonstrated through **method overriding and method overloading mechanisms**

****+** Abstraction**:****  
Allows developers to focus on core components while hiding complex implementation details. Abstract classes or interfaces define required behaviors, with subclasses providing the implementation. This simplifies the system and increases flexibility and scalability in software development.

Applying these fundamental OOP principles in the **SpyAgency2024** project not only helps organize the source code clearly, making it easy to read and maintain, but also creates a solid foundation for the team to develop new features in the future without affecting the existing system structure.

**1.1.2. Java Swing Technology in Desktop Application Development**

**Java Swing** is a powerful user interface library within the Java Standard Edition, widely used to develop desktop applications. Swing provides a diverse set of GUI components such as buttons, labels, text boxes, tables, panels, etc., along with customizable interfaces and flexible event handling capabilities, enabling developers to build software with user-friendly and easy-to-use interfaces.

In the **SpyAgency2024** project, Java Swing plays a key role as the main technology for designing and developing the user interface of the agent management software. The prominent features applied during development include:

**+ Rich set of UI components:** Swing offers comprehensive components like JButton, JLabel, JTextField, JComboBox, JTable, JScrollPane, etc., which help the team design data input forms, display data tables, and control panels flexibly and efficiently.

**+ Layout management:** Swing supports arranging UI components using various layout managers such as BorderLayout, FlowLayout, and GridLayout, helping create clear and neat working windows.

**+ Event handling:** The application leverages Swing’s event handling mechanism to respond to user actions such as button clicks, table selections, data searches, and chart creation...

**+ UI customization:** Java Swing allows changing colors, fonts, borders, icons, etc., to create a consistent, user-friendly, and visually intuitive interface.

**+ Integration with data visualization libraries:** Although Swing lacks a visual builder like JavaFX’s SceneBuilder, it is fully compatible with libraries such as **JFreeChart,** facilitating easy embedding of charts into the application

#### ****1.1.3.**** MySQL Database Management System and SQL Language

In the **SpyAgency2024** project, the system uses **MySQL** as the primary database management system, deployed via **XAMPP** — a popular server emulator integrating Apache, MySQL, PHP, and phpMyAdmin to simplify database installation and management on personal computers

**+ Table-based data management:** The system organizes and stores data in tables, each consisting of rows (records) and columns (fields). Tables are linked via **primary keys and foreign keys** to ensure data integrity and logical relationships between entities such as agents, missions, skills, and organizations.

**+ Data querying with SQL:** The system uses SQL to perform database operations like inserting (INSERT), updating (UPDATE), deleting (DELETE), searching (SELECT), and aggregating data from tables.

**+ Database management via phpMyAdmin (XAMPP):** Alongside MySQL Workbench, the team uses **phpMyAdmin** integrated in XAMPP to create tables, edit data, and quickly test SQL queries during development.

**+ Java connectivity through JDBC:** To enable Java Swing software to communicate with the database, the team uses **JDBC (Java Database Connectivity)** — a standard Java API for connecting and executing SQL commands. The connection class is organized separately for better management and reuse across the project.

Using **XAMPP** offers many benefits such as easy installation, convenient local database management, and suitability for internal desktop projects like **SpyAgency2024**.

#### ****1.1.4.**** Data Structures and Java Collections Framework

During the development of the **SpyAgency2024** application, choosing appropriate data structures played an important role in optimizing performance and simplifying data handling. The project utilized **JFreeChart** — an open-source Java library supporting the creation of professional 2D charts. In the **SpyAgency2024** project, the team used JFreeChart to visualize intelligence data and statistical results.

**Types of charts used**:****

**+ Bar Chart:** Displays the top agents with the highest number of missions.

**+ Pie Chart:** Shows the distribution of agents by nationality.

**+ Line Chart:** Tracks the number of missions over the years.

**Dataset:** The data for charts is organized in the form of DefaultCategoryDataset for bar and line charts, and DefaultPieDataset for pie charts, helping to visualize information effectively

**Chart customization:** JFreeChart supports customizing titles, axis labels, colors, fonts, and sizes, enabling the creation of visually appealing and clear charts

**Integration with Java Swing:** Charts are embedded directly into Swing via ChartPanel, allowing charts to be displayed alongside data tables and management forms within the same working window.

**Principles of effective visualization:** The team focuses on selecting chart types appropriate for each data set, using distinguishable colors and logical layouts to help users easily grasp information without confusion.

### ****1.2.**** Current Research Status

#### ****1.2.1.**** Modern Technology Status

In today’s era of rapid digital transformation, information technology has been continuously advancing, providing numerous powerful tools and platforms to support data collection, management, analysis, and visualization across many fields, especially in security and intelligence. Notable technologies and platforms include:

**Desktop application development platforms:**

**Java,** especially with the **Java Swing library**, remains a popular and stable tool for building desktop applications due to its strong data processing capabilities, user-friendly interfaces, and ease of integration with external libraries. Other platforms like .**NET (C#) and Python (Tkinter, PyQt)** are also widely used in developing data management software.

**Database management systems:**

**MySQL, PostgreSQL, and SQL Server** continue to be popular choices for data storage. Additionally, NoSQL databases such as **MongoDB** are gradually being adopted for systems requiring flexible and unstructured data processing.

**Data visualization libraries**:****  
Besides **JFreeChart** for Java, libraries like **Matplotlib, Seaborn (Python), D3.js (JavaScript)**, and advanced data analytics/BI platforms like **Power BI** and **Tableau** are widely used for data visualization and analysis. However, most of these tools serve general analysis purposes and are less often implemented in specialized intelligence applications.

**Development and project management tools:**  
Modern IDEs such as **NetBeans, IntelliJ IDEA, and Eclipse** provide excellent support for Java Swing development. Database management tools like **phpMyAdmin** (within **XAMPP**) and **MySQL Workbench** help design and test databases visually. Source code management with **Git** and project building with **Maven** have become popular standards.

**Assessment:**  
With the rapid advancement of modern technologies, developing specialized intelligence data management and analysis software like **SpyAgency2024** is entirely feasible. The challenge lies in selecting the right technologies and integrating components optimally to fully meet business requirements and ensure easy future expansion.

#### ****1.2.2.**** Analysis of Similar Management and Data Analysis Applications and Solutions

Currently, several systems and applications have been developed to support the management and analysis of intelligence data or similar organizational personnel data. However, certain limitations still exist:

**General personnel or organizational management software:**  
There are many software solutions that support personnel and organizational management. However, these applications usually only serve the needs of managing personal information and tasks without the capability for in-depth statistical analysis or data visualization specifically for the intelligence field.

**Mobile or web-based applications:**Some mobile or web applications assist in storing and updating personnel information, but few focus on mission data statistics or the relationships between individuals in intelligence organizations. Moreover, they often lack data visualization features to support analytical processes.

**Business Intelligence (BI) tools:**Software such as **Power BI, Tableau, or Google Data Studio** can connect to and analyze data from multiple sources. However, they require users to have BI knowledge and technical skills to configure connections and design dashboards. Additionally, these are general-purpose analytical tools, not specifically packaged or tailored for intelligence agencies or internal security departments.

**Open-source projects:**Some programming communities or research groups have developed personnel management or intelligence data projects on a small scale. However, these projects are often experimental or academic in nature, lacking optimized user interfaces and missing direct data visualization features within desktop software.

**Summary:** From the above status, it is clear that there are few complete and user-friendly solutions specifically for intelligence data management and analysis. Therefore, the **SpyAgency2024** project aims to build a simple, easy-to-use desktop application that integrates comprehensive features for managing agents and missions, with built-in data visualization through charts within the software itself, while ensuring extensibility and internal data security.

## Chapter 2: Proposed Solution and Application Design Analysis

### ****2.1.**** Proposed Solution

The application **"SpyAgency2024"** is proposed as a desktop **software** built **using the Java programming language** with the **Java Swing library** for designing the graphical user interface (GUI). Data related to agents, missions, skills, teams, and their relationships will be stored and managed in a MySQL database. The database is deployed via **XAMPP** and managed using the **phpMyAdmin tool**.

Data statistics and intelligence visualization functions will be implemented using the **JFreeChart library**, integrated directly into the Swing application through **ChartPanel** to optimize the build process.

#### ****2.1.1.**** General Architecture

The application is designed with a **multi-layered** architecture following the **Model–View–Controller (MVC)** pattern to ensure modularity, maintainability, and extensibility:

**View Layer (Java Swing):**  
 Responsible for displaying information and receiving user interactions via graphical interfaces designed with Java Swing components such as JButton, JTextField, JTable, JComboBox, JLabel, etc. Charts are embedded directly into the interface windows using **ChartPanel** from the **JFreeChart library**.

**Controller Layer:**  
 Acts as the bridge between the view and data processing layers. Controllers handle user-generated events, invoke methods from DAO or Chart classes to retrieve data, process results, and update the corresponding views.

**Model Layer:**  
 Represents the data entities in the application such as Agent, Mission, Skill, Team, etc. These classes use private attributes and provide public getter/setter methods to ensure encapsulation and data protection.

**Data Access Object Layer (DAO):**  
 Handles direct database operations through **JDBC**. DAO classes provide CRUD methods (Create, Read, Update, Delete), as well as basic search and statistical queries. Separating DAO reduces dependencies between business logic and data access code.

**Chart and ChartDAO Layer:**

**+ ChartDAO:** Responsible for querying specific statistical data from the database, supporting charts such as mission count by year or agent distribution by nationality.

**+ Chart:** Receives data from ChartDAO, processes it, and displays charts using **JFreeChart** in forms like bar charts, pie charts, and line charts, embedded directly in the Swing interface.

**Config Layer:**  
 Contains database connection parameters such as URL, username, and password, and provides methods to establish connections via **JDBC**. Having a separate Config class facilitates easy management and modification of connection settings when needed.

**Icon Layer**:****  
 Stores and manages image icon files used in the user interface to ensure a visually appealing and consistent graphical appearance

Summary, The general architecture of the application is designed with **modularity** in mind, ensuring each layer fulfills its responsibilities properly. This approach minimizes dependencies among components, increases reusability, and facilitates maintenance and future extensibility.

#### ****2.1.2.**** Functional Requirements

##### ****1.**** User Management and Authentication

**Login/Logout**:****

+ The system provides a login interface where users can enter their username and password.

+ User credentials are authenticated against data stored in the **Account** table of the MySQL database.

+ Passwords are encrypted using the **SHA-256** algorithm before being stored.

+ Displays an error message upon failed login attempts and supports account locking after multiple consecutive failures.

+ Stores the current user session information upon successful login and redirects to the **Dashboard** interface.

+ Supports logout functionality

**User Profile Management:**

+ Displays basic information (full name, email, role) of the currently logged-in user.

+ Allows users to change their own password.

**User Account Management:**

+ Admin users are allowed to view the list of accounts, add new users, edit user information (email, role), or delete users from the system.

##### ****2.**** Agent and Mission Data Management

**Display of Agents, Missions, Skills, and Organizations:**

Data is presented in JTable format with appropriate columns for each table.

Allows sorting and filtering based on specific fields such as agent name, nationality, or birth year.

**Add New Records:**

Provides input forms to add new records to tables like Agent, Mission, Skill, and Organization.

Validates input data before saving to the database.

**Update Records**:****

Allows users to select a record for editing and save changes to the database.

**Delete Records:**

Allows users to delete a record with a confirmation prompt before performing the deletion.

##### ****3.**** Dashboard and Data Visualization

**Statistical Charts using JFreeChart**:****

**Bar Chart:**

Displays the top N agents with the highest number of missions.

Shows the number of missions carried out each year.

**Pie Chart:**

Displays the distribution of agent nationalities or gender ratios.

**Line Chart:**

Visualizes the trend of missions over the years.

##### ****4.**** Additional Features

**Relationship Management between Tables (Agent – Mission – Team – Skill)**:****

Manages linkage tables such as **AgentMission**, **AgentSkill**, **TeamRel**, allowing users to add, edit, and delete relationships between entities.

#### ****2.1.3.**** Non-Functional Requirements

##### ****1. Security:****

User passwords are hashed using the **SHA-256** algorithm and stored securely.

Unauthorized access to sensitive data and administrative functions is strictly prevented

##### ****2.**** Usability****:****

The user interface is designed to be friendly and intuitive

Buttons and notifications are clearly labeled and visually accessible..

The system responds quickly to user actions, ensuring smooth operation.

##### ****3.**** Reliability & Robustness****:****

The application operates stably, even with large data volumes.

Exception handling is implemented effectively, with clear and user-friendly error messages.

Data integrity is ensured during add, update, and delete operations.

##### ****4.**** Maintainability & Extensibility****:****

The source code is well-organized in a modular structure: Model – DAO – Controller – Chart – View.

Object-Oriented Programming (OOP) principles and consistent naming conventions are followed.

It is easy to enhance or extend modules and add new features without affecting the existing system.

##### ****5.**** Performance****:****

Database queries are optimized for fast execution.

Displaying large tables or charts does not negatively impact the overall performance of the application.

**2.2. System and User Interface Design Analysis**

#### ****2.2.1.****System Architecture

The **SpyAgency2024** application is designed and built using a **layered architecture**, following the **Model – View – Controller (MVC)** design pattern with additional specialized support layers. This architecture ensures a clear separation of the user interface, business logic, and data access, promoting modularity, ease of maintenance, and extensibility during development.

Main Components of the System Architecture:

**Presentation Layer (View)**

Built using **Java Swing**, with components such as JButton, JTextField, JLabel, JTable, JComboBox, and ChartPanel (from JFreeChart) to display information and handle user interactions.

View classes are responsible for presenting data and forwarding user actions to the Controller for processing.

The **Util layer** contains common utility functions shared across the entire application, such as password hashing, data formatting, and other helper methods. A typical example is the password hashing function that uses the **SHA-256 algorithm** to protect user login information before storing or comparing it with data in the database.  
By implementing password encryption in the Util layer, security logic is decoupled from other layers, thereby improving code **reusability** and **maintainability**.

Examples: AgentManagementPanel.java, MissionDashboardPanel.java, AffiliationManagementPanel.java.

**Controller Layer**

Responsible for handling events triggered from the user interface, receiving data from the View, sending requests to the DAO or ChartDAO, processing the data, and updating the View accordingly..

Controllers are organized by functional modules, such as AgentController.java, MissionController.java, DashboardController.java, etc...

**Data Access Layer (DAO)**

Responsible for directly querying the MySQL database via JDBC.

Performs CRUD (Create, Read, Update, Delete) operations and search functions for each table such as AgentDAO.java, MissionDAO.java, AffiliationDAO.java, etc.

The DBConnection class in the Config package manages the creation and closing of database connections.

**Data Model Layer (Model)**

Contains POJO classes representing data entities such as Agent.java, Mission.java, Skill.java, Team.java, etc.

Each class includes private attributes and public getter/setter methods to facilitate data exchange between the different layers of the application.

**Chart & Statistics Layer (Chart & ChartDAO)**

**ChartDAO:** Executes specific queries to retrieve statistical data for charts (e.g., top agents by number of missions, nationality distribution, etc.).

**Chart:** Receives data from **ChartDAO**, processes it, and generates visual charts using the **JFreeChart library**. These charts are embedded into the Swing interface using ChartPanel.

Supported chart types include bar charts, pie charts, and line charts.

**Configuration Layer (Config)**

Contains system configuration classes and files such as DBConnection.java, which handles database connection parameters and initialization.

Implements the Singleton design pattern to ensure only one database connection instance exists at any time.

**Icon Layer**

Stores and manages icon image files used in the user interface, ensuring a consistent and intuitive visual experience throughout the application..

**Summary**:****

The architecture of the **SpyAgency2024** application is clearly divided according to an extended **MVC model**, with additional specialized layers including **Chart, ChartDAO, Config**, and Icon to meet the business requirements and data visualization features of an intelligence management system. This structure ensures::

+ Ease of maintenance, debugging, and upgrades

+ Scalability for new feature additions

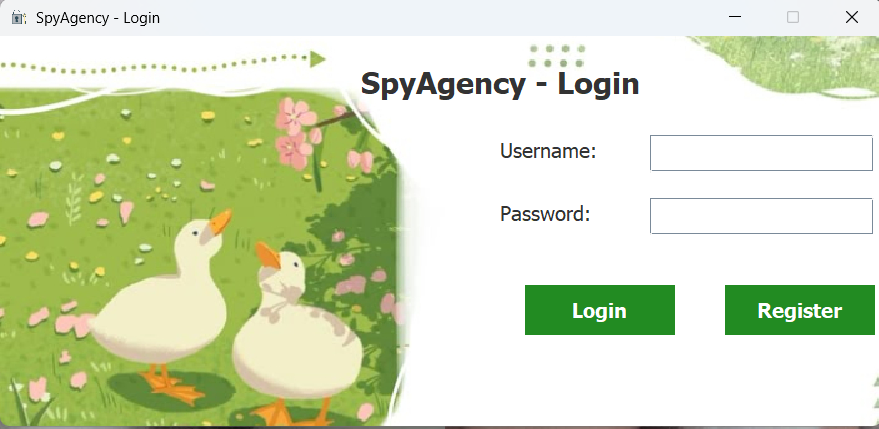
+ Enhanced stability, security, and code reusability

****2.2.2** User Interface Design Analysis**

The user interface of the **SpyAgency2024** application is developed using **Java Swing**, integrating components such as JButton, JLabel, JTextField, JTable, JComboBox, and ChartPanel (**from the JFreeChart library**) to display data and allow direct interaction with users.

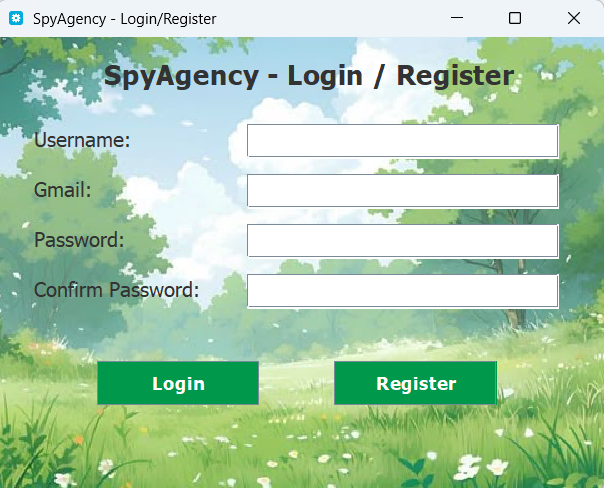
The design objective is to deliver **a clean, intuitive, and user-friendly interface** that effectively supports intelligence data management and analytical operations..

**Login Screen:**



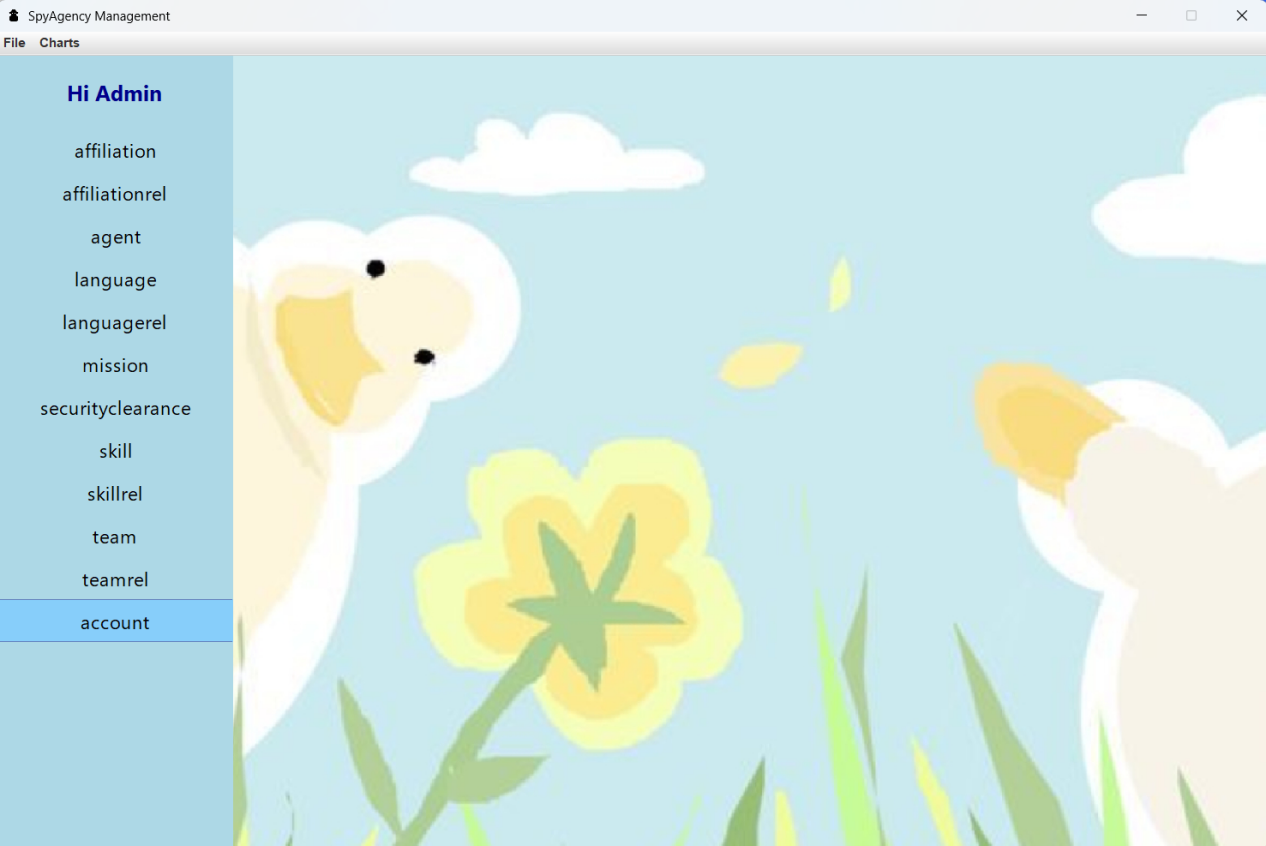
*Hình 1*

**Account Registration Screen :**



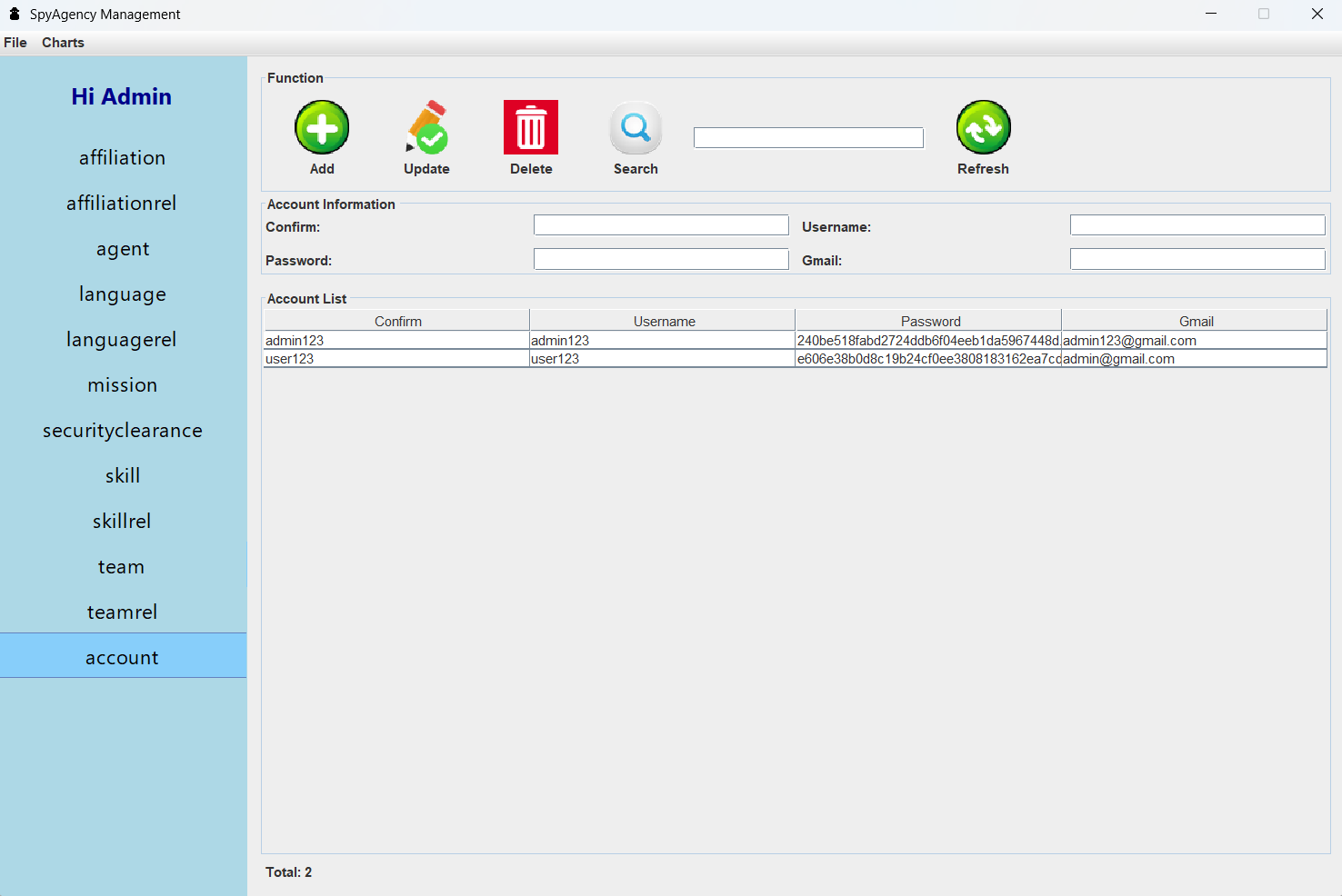
*Hình 2*

**Dashboard:**

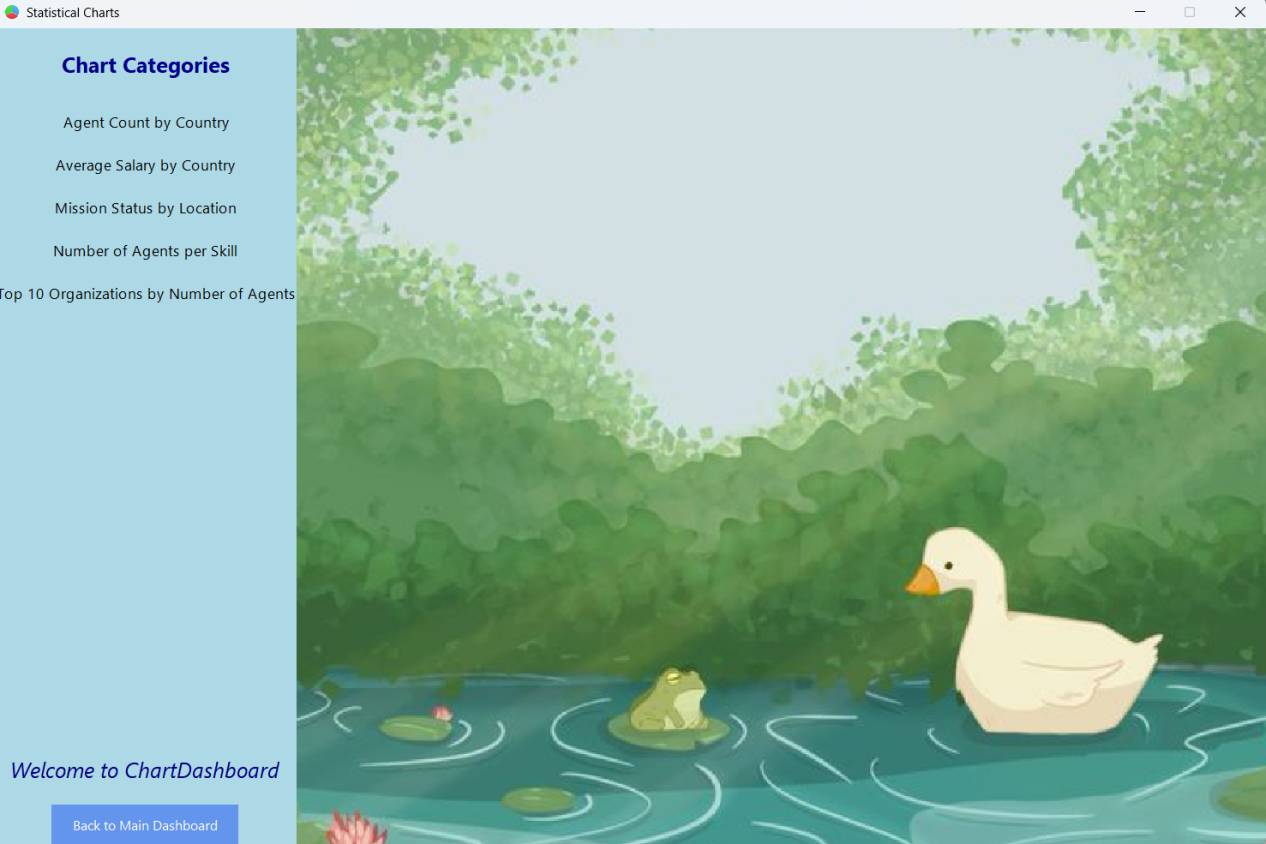


*Hình 3*

**User Management Interface Screen:**

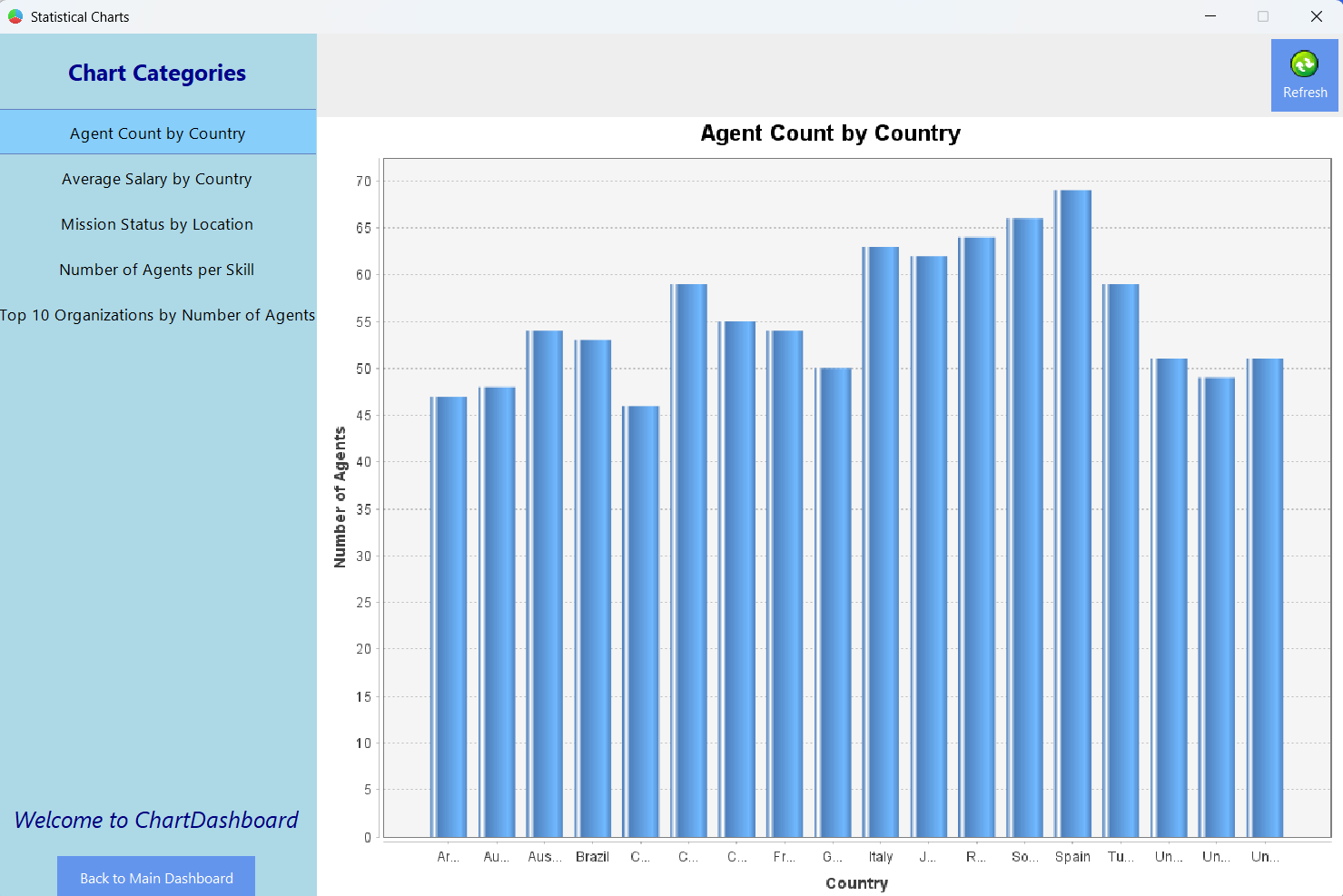


*Hình 4*

**Main Chart Interface Screen:** 

*Hình 5*

**Chart Viewing Screen :**



*Hình 6*

**2.3 User Interface Information.**

This section describes the process by which a user logs into the Spy Agency Management System and how the system displays the main interface after a successful login

**-** Launching the application**:**

+ The login interface is the first screen displayed, controlled by the LoginInterface2.java class.

**-** User login process**:**

+ The intelligence officer enters their **username and password** into the input fields (JTextField and JPasswordField)..

+ They click the "**Login**" button, which is represented by an icon styled in a modern management software design.

**-** Login processing**:**

+ The button click event is handled via an **ActionListener**

+ The system calls the checkLogin(username, password) method to verify credentials.

The checkLogin method interacts with the AccountDAO class to check the user’s account information stored in the MySQL database.

Passwords are validated after being hashed using the **SHA-256** algorithm to ensure security

**Login scenarios**:****

Successful login:

The login window is closed.

The application transitions to display the DashboardPanel, which directly controls the main home interface .

A welcome message appears in the dashboard: **"Hi Admin".**

Failed login:

An error message is displayed directly on the interface: **"Incorrect username or password.”**.

**-** Dashboard Interface**:**

The main interface is divided into two areas:

The **left sidebar (Sidebar)** contains functional menu items that are **customized based on the user's role** in the system.

On the **right side**, a **background image** is displayed to add color and improve the visual appearance of the interface, creating a more **comfortable and user-friendly experience.**

**Other buttons include**:

**Logout:** Returns the user to the login screen when they want to switch to a different account **.**

**Exit:** Closes the application quickly, helping to save time.

Other special functions are located in the **left sidebar** of the main screen. Each button opens a different page, and each page contains full features appropriate for a management application. The functions include:

- Add or Create:

+ The user fills in all required information in the JTextField input fields.

+ Click the **ADD** button

+ The data is automatically saved to the list and also inserted into the database table through the established connection.

- Update or Edit:

+ After selecting a row from the information table below, the user can edit the data as desired.

+ Click the **Update** button.

+ The new information will be updated automatically, and the database will also be updated accordingly.

- Delete:

+ After selecting a row from the information table, the user can choose to delete it

+ Click the **Delete** button

+ A confirmation dialog will appear asking whether the user wants to delete the record.

+ If confirmed, the selected information will be deleted, and the database will be updated to reflect the removal.

- Refresh:

+ When clicked, all displayed data will be refreshed

+ Any previously filled input fields will be cleared and reset to blank

- Search:

+ There is a separate input field where users can enter the information they want to search for.

+ When the **Search** button is clicked

+ All relevant information matching the keyword will be displayed.

Besides the management pages, there is also a similar management system but with a special feature: the hash information is displayed below for users to view and to modify the data through CRUD functions.

**Statistics and Charts:**

Regarding the interface, the ChartDashboard features a visually appealing and user-friendly background, creating a comfortable experience for users. Additionally, the left sidebar contains buttons which, when clicked, display different charts. Although users cannot create their own custom charts, the generated charts are designed to be visually attractive, allowing users to gain a comprehensive overview. All data is connected to MySQL, so any additions, deletions, or updates automatically affect the charts. Next to the charts, there is a refresh button that allows users to update the charts to reflect the most current information. This enables users to quickly grasp the latest data, supporting development and promoting progress in various fields.

**2.4 MySQL Database Structure**

#### ****2.4.1**** Table Description

**- Table account**

**Objective:** User Account Management

**Columns**:****

id (INT, PK, AI): Account ID

username (VARCHAR(50), NOT NULL, UNIQUE): Username

password (VARCHAR(255), NOT NULL): Password (hashed)

Confirm (VARCHAR(20), NOT NULL): Confirm Password

****- Table** agent**

****Objective:**** Agent Information Management

**Columns**:****

agent\_id (INT, PK, AI): Agent ID

First\_name (VARCHAR(255), NOT NULL): Agent first name

Last\_name (VARCHAR(255), NOT NULL): Agent last name

address (VARCHAR(255)): Address

city (VARCHAR(100)): City

country (VARCHAR(100)): Country

salary (DECIMAL(15,2)): Salary

****- Table** affiliation**

**Ojective:** Agent Organization Information Management

**Columns:**

Affiliation\_id (INT, PK, AI): Organization ID

Affiliation\_name (VARCHAR(255), NOT NULL): Organization Name

description (TEXT): Organization Name

****- Table** mission**

**Ojective:** Mission Information Management

**Columns:**

mission\_id (INT, PK, AI): Mission ID

mission\_name (VARCHAR(255), NOT NULL): Mission name

location (VARCHAR(255)): Location

agent\_id (INT, FK): Agent ID

access\_id (INT, FK): Security clearance code (referencing the securityclearance table)

team\_id (INT, FK): Team code

mission\_status (VARCHAR(50)): Mission status

****- Table** skill**

**Ojective:** **Spy skills category management**

**Columns:**

id (INT, PK, AI): Skill ID

name (VARCHAR(255), NOT NULL): Skill name

****Table** language**

**Ojective:** Language category management.

**Columns:**

language\_id (INT, PK, AI): language code

language\_name (VARCHAR(255), NOT NULL): language name

****- Table** securityclearance**

**Objective:** Management of agents' security clearance levels.

**Columns:**

sc\_id (INT, PK, AI): Security level code

sc\_level (VARCHAR(50), NOT NULL): Security level name

description (TEXT): Difficulty description

****- Table** team**

**Ojective:** Spy group management.

**Columns:**

id (INT, PK, AI): Team ID

name (VARCHAR(255), NOT NULL): Team name

meeting\_frequency (VARCHAR(50)): Team meeting frequency

****- Table** affiliationrel**

**Objective:** Store information about the relationship between agents and organizations.

**Columns:**

affiliation\_strength (VARCHAR(50)): Level of association

agent\_id (INT, PK, FK): Agent ID

affiliation\_id (INT, FK): Organization ID

****- Table** skillrel**

**Objective:** Store information about the relationship between agents and skills.

**Columns:**

agent\_id (INT, FK): Agent ID

skill\_id (INT, FK): Skill ID

****- Table** languagerel**

**Objective:** Store information about the relationship between agents and languages.

**Columns:**

agent\_id (INT, FK): Agent ID

language\_id (INT, FK): Language ID

****- Table** teamrel**

**Objective:** Store information about the relationship between agents and groups.

**Columns:**

agent\_id (INT, FK): Agent ID

team\_id (INT, FK): Team ID

#### ****2.4.2.**** Constraints and Relationships

Constraints**:**

**Primary Key:**  
Each table has an id column as the primary key (PK).

**Foreign Key:**

agent.security\_clearance\_id → securityclearance.id

affiliationrel.agent\_id → agent.id

affiliationrel.affiliation\_id → affiliation.id

skillrel.agent\_id → agent.id

skillrel.skill\_id → skill.id

languagerel.agent\_id → agent.id

languagerel.language\_id → language.id

teamrel.agent\_id → agent.id

teamrel.team\_id → team.id

**Unique:**

account.username

**Not Null:**

account.username, account.password, account.role, agent.name, affiliation.name, mission.name, skill.name, language.name, securityclearance.level, team.name

Relationships between tables**:**

account independent.

agent — securityclearance: Relationship 1-n

agent — affiliationrel: Relationship 1-n

agent — skillrel: Relationship 1-n

agent — languagerel: Relationship 1-n

agent — teamrel: Relationship1-n

affiliationrel — affiliation: Relationship n-1

skillrel — skill: Relationship n-1

languagerel — language: Relationship n-1

teamrel — team: Relationship n-1

## CHƯƠNG 3: SYSTEM OPERATION MECHANISM

### ****3.1**** Overall Structure

**The Spy Agency Management** software system is designed and developed based on the **Java programming language**, combined with a **MySQL** **database** for centralized data storage and management. The project is organized following **a layered architecture model**, closely integrated with the **MVC (Model-View-Controller)** software design pattern and t**he DAO (Data Access Object)** pattern to separate business logic, user interface, and data processing.

The overall system is divided into separate **packages**, each responsible for a distinct role or function such as managing models, views, controllers, DAOs, configurations, and charts. This organization ensures the source code is managed scientifically, easy to expand, maintain, and upgrade later

The system structure includes:

**Model:** Classes representing data objects.

**DAO:** Classes for data operations with MySQL.

**Controller:** Handles events and coordinates between View and DAO.

**View:** User interface classes built with Java Swing.

**Chart & ChartDAO:** · Handle data processing and visualization via charts.

**Config:** System configuration and database connection

**Icon:** Management of icons and UI resources.

This design guarantees that the system can easily develop new features such as statistics, advanced access control, or alerts without affecting the existing structure.

### ****3.2**** Model Layer

**The Model layer** is responsible for representing actual entities within the software system, such as **Agent, Mission, Affiliation, Account, Skill, Team, etc**. Each main Model class is a **Plain Old Java Object (POJO)** with private attributes and public getter/setter methods.

Building the Model according to the POJO standard helps to:

Ensure encapsulation.

Easily map data to corresponding tables in the database.

Provide a foundation for flexible data transformation between DAO, Controller, and View layers.

This organization helps maintain consistency of data between the program and the database, facilitating easy addition, update, and deletion of data.

.

****3.3** DAO Layer (Data Access Object)**

**The DAO layer** acts as an intermediary between the system and the database, specializing in data operations such as adding, deleting, updating, and searching. DAO classes like AgentDAO, AffiliationDAO, and MissionDAO use JDBC to execute SQL statements directly and return results to the Controller for further processing.

Advantages:

+ Separates SQL queries from the Controller, improving maintainability.

+ Easily allows changes to the database or table structure without affecting other layers.

+ Reuses methods to avoid code duplication.

DAO also helps handle exceptions and manage connections efficiently, preventing resource leaks.

### ****3.4**** Controller Layer

**The Controller acts** as a coordinator between the **View and DAO**. It receives user events from the interface, calls DAO methods to perform business operations, and then returns the results to the View for display..

Example:

- When the user clicks the “**Add Agent**” button, the Controller gathers data from the input form, creates a new Agent object, calls AgentDAO.insertAgent(agent), and then updates the data table display.

Advantages:

+ Separates business logic from the UI layer.

+ Provides better control over the processing flow and is easier to extend functionality when needed.

Additionally, the Controller performs data validation before sending it to the DAO, ensuring data integrity across the system.

### ****3.5**** View Layer (Java Swing)

**The View** in the system is built using **Java Swing**, consisting of individual **JPanel components** like **AgentManagementJPanel, MissionManagementJPanel, and AffiliationManagementJPane**l. Each View is designed with a modern layout and includes:

**+ Labels** for titles and descriptions.

**+ TextField** for user input.

**+ ComboBox** for quick selection.

**+ Button** for executing action

**+ JTable** for displaying data in tabular format.

**+ Icon** **and visual effects** for user-friendliness

The UI components are arranged logically, with attractive icons and optimized colors to help users navigate easily and minimize errors during operation.

### ****3.6**** Chart and ChartDAO Layers

The system uses the JFreeChart library to visualize data through pie charts, bar charts, and line charts. The Chart class handles chart configuration and rendering, while the ChartDAO class fetches data from MySQL with appropriate queries.

Examples:

- Count the number of agents by nationality using the method AgentChartDAO.countByNationality() and return a dataset for the pie chart.

- Display the number of missions by status (success, failure) in a bar chart..

Advantages:

- Allows managers to monitor data in a visual format.

- Supports trend analysis and efficient organizational performance evaluation.

### ****3.7**** Connection Configuration (Config)

**The Config class** manages database connection parameters such as:

+ Server address.

+ Database name.

+ Username.

+ Password.

+ Port number.

T**he DBConnection class** is designed following the Singleton pattern to ensure only one connection is created and reused across the entire system. This saves resources and prevents errors during simultaneous database access.

Advantages:

+ Easy adjustment of parameters during deployment.

+ Secure connection information management.

+ Avoids repeating connection setup code in each DAO.

### ****3.8**** User Authentication Handling

The system includes a user authentication mechanism at the login screen. The authentication process involves:

Entering username and password.

Encrypting the password using the **SHA-256 algorithm**.

Comparing the encrypted password with the one stored in the Account table.

## CHAPTER 4: EXPERIMENTS AND EVALUATION

### 4.1 Testing Environment and Procedure

To ensure the accuracy and reliability of the system, the testing process was conducted in a specific hardware and software environment, closely aligned with the actual conditions under which modern information management applications operate. The use of an up-to-date testing environment ensures the system meets performance and stability requirements during real-world operation.

**Testing environment configuration:**

**Operating System:** Windows 11 Pro 64-bit

**JDK Version**:**** Java Development Kit 17 (LTS)

**Database Management System**:**** MySQL Community Server 8.0

**Development Tool**:**** IntelliJ IDEA version 2024.1 Ultimate

**Chart Library**:**** JFreeChart version 1.5.4

**The test data set** was constructed with over **1000 records**, evenly distributed across the system’s key database tables such as **Agent, Mission, Affiliation, Affiliationrel, Skill, Skillrel, Team, Teamrel, Securityclearance, and Account**. These records were simulated based on realistic scenarios to ensure diversity and complexity, aiming to test the system’s processing capabilities.

T**he testing process** was carried out according to standardized steps as follows:

- **Login Functionality Testing**: Conducted using various user accounts to verify access control, credential validation, and appropriate interface navigation based on user roles.

- **Data Management Testing:** Performed operations such as adding new records, editing, deleting, and searching across all data tables to ensure data integrity and synchronization between the user interface and the database.

**- Statistical Chart Testing:** Created charts in various formats such as bar charts, pie charts, and line charts to verify accurate data visualization and support for dynamic updates when data changes.

**- Performance Testing:** Executed continuous operations with a large dataset to evaluate the application’s responsiveness, concurrent processing capability, and identify any errors or system freezes.

The testing was carried out across multiple sessions using simulated error scenarios such as incorrect login attempts, duplicate primary keys, and deletion of records being referenced—this was done to evaluate the system’s exception handling and stability.

### 4.2 Testing Results

After conducting comprehensive testing based on the established plan, the results are as follows:

**Login Functionality**:****  
The system successfully validates login credentials. Cases such as non-existent accounts, incorrect passwords, or unauthorized access are correctly handled. Upon successful login, the system displays the interface corresponding to the user's role, clearly distinguishing between administrative and regular user features.

**Data Management**:****All CRUD functions for each table performed correctly as specified. The user interface updates in real-time, ensuring data displayed matches the actual database records. The system supports fast keyword searches and filtering based on specific criteria, offering convenience and time efficiency.

**Chart Visualization**:****  
The system generates accurate and visually intuitive charts across multiple types. Charts dynamically update upon data changes without requiring a program restart. Rendering speed is fast, with clear colors, labels, and complete information, enhancing monitoring and analytical capabilities.

**Performance**:****  
During tests involving over 1000 records and prolonged usage, the system remained stable with no runtime errors or application freezes. CRUD operations responded within an average of 1 second, indicating satisfactory real-world performance.

Additionally, the system was tested with simulated exceptions such as inserting duplicate primary keys, deleting records referenced by foreign keys, and repeated incorrect login attempts. All exceptions were handled correctly, and informative messages were displayed to users.

### 4.3 Product Evaluation

After a comprehensive testing process, the system has demonstrated strong capability in meeting the initial functional requirements. The product can be evaluated based on the following criteria:

**Strengths**:****

**Adoption of a Standardized MVC Architecture:** - The system is built upon the **Model-View-Controller (MVC)** architecture, which clearly separates concerns among data models, business logic, and user interfaces. This structure enhances maintainability and scalability while making future development more manageable and controlled.

**Modern and User-Friendly Interface**:****- The user interface is designed with a modern aesthetic, using harmonious colors and logically arranged features. This ensures users can quickly become familiar with the system and perform management tasks efficiently..

**Intuitive Chart Integration for Data Analysis**:****- By integrating data visualization charts, the system serves not only as a data management tool but also as a platform for analytical insights, supporting better decision-making and reporting.

**Stable Performance with Large Data Volumes**:****- The system handles more than 1,000 records smoothly, maintaining fast response times and error-free operation even under multiple concurrent actions.

**Limitations:**

**Lack of Multi-Language Support:**  
- Currently, the interface supports only English. This limitation may affect usage in international or multilingual organizations. Future versions should include language selection or bilingual support to expand the system’s usability.

**Absence of Advanced Analytical Features**:****- The current system offers basic statistical and charting features. Advanced capabilities such as **agent activity prediction, relationship analysis between agents and missions, or team performance evaluation have not yet been implemented**. These features represent potential areas for future development to increase the product’s value.

**Conclusion**:****  
Through thorough testing and evaluation, the system has proven to meet expectations in terms of functionality, performance, and user interface. It provides a solid foundation for further feature expansion, enhanced analytics, and integration with other systems in future iterations..

**Conclusion**

After a serious process of research, design, and development, the project **“SpyAgency2024 – Agent Data Management and Analysis System”** has successfully achieved the initial goals. A desktop application was developed to manage agent information, missions, and related entities, with robust data visualization features, a user-friendly interface, and strong security. The key outcomes of the project include:

## ****1.**** Successfully implemented User Management and Authentication

The application provides full login and logout features, along with secure session management. User passwords are encrypted using **the SHA-256 algorithm** before being stored in the **MySQL database** to ensure security. Additionally, users can update their personal information, and an administrator interface is available for managing user accounts, assigning permissions, editing information, and deactivating accounts when necessary. This ensures secure and appropriate access control..

## ****2.**** Developed a comprehensive module for managing agents and related data entities

The system features a user-friendly and well-structured interface that supports full CRUD (Create, Read, Update, Delete) operations on key data tables such as **Agent, Mission, Affiliation, Skill, Language, Team, and relationship tables including AffiliationRel, LanguageRel, SkillRel, and TeamRel**. The MySQL database is designed with a normalized structure, ensuring data integrity and high scalability.

Integration with **MySQL tools** allows for efficient data inspection and management. Sample data were created based on realistic intelligence agency scenarios, including comprehensive details on agents, missions, organizations, and access permissions, making the data handling process more intuitive.

## ****3.**** Implemented data visualization features using charts

The project successfully integrates the **JFreeChart library** to visualize agent and mission data in dynamic charts. The implemented chart types include:

**Bar Chart**: Visualizes agent distribution by nationality or by team.

**Pie Chart**: Displays the proportion of agents by security clearance level.

**Line Chart**: Shows the trend of mission completion over months or years..

Users can select parameters such as country, team, or time period to customize the visualizations for specific analytical needs, thereby enhancing the effectiveness of agent management and monitoring.

## ****4.**** Effectively applied Object-Oriented Programming (OOP) techniques and Java technologies

During the design and development process, the project made full use of **OOP** principles such as encapsulation, inheritance, and polymorphism in designing the **POJO** classes (Agent, Mission, Affiliation, Team, Skill, Language), **DAO classes** (AgentDAO, MissionDAO, AffiliationDAO, etc.), and **service classes** that handle business logic (AuthService, ChartService, MissionService).

The project also applied modern technologies and tools:

**Java Swing**: Used for building the intuitive desktop graphical user interface (GUI)..

## ****5.**** Completed a well-structured project and documentation system

All source code, resources (icons, images, CSS), FXML configuration files, and SQL scripts (for table creation and sample data insertion) are organized clearly and professionally following real-world project standards. In addition, the project includes **comprehensive technical documentation, a user manual, and a design analysis report**, supporting future deployment, operation, and expansion.

**Future Development Directions**

Although the **SpyAgency2024 application** has achieved many positive results, there are several promising directions for future development:

Add **notification and security alert features** to monitor suspicious account activities..

**Develop AI-based analytics to assess risk levels** in mission assignments.

**Design a real-time dashboard** to provide a comprehensive overview of agents, missions, and organizational data.

**Integrate multi-language** support to serve international users.

With the results achieved, it can be affirmed that the **SpyAgency2024 project** not only helped the developer consolidate and apply knowledge of Java programming, databases, and data visualization but also resulted in a practical, high-utility software product. This provides a solid foundation for further research and development of advanced features in the field of intelligence data management and analysis, supporting supervision, administration, and information processing in security organizations.

# References

1. Slides of the Object-Oriented Programming (OOP) course by Dr. Lê Ngọc Hiếu
2. JFreeChart Project. JFreeChart Developer Guide. Truy cập tại: <http://www.jfree.org/jfreechart/.>
3. git clone <https://github.com/hgbaodev/QuanLyKhoDienThoai.git>.