

**VIETNAM GENERAL CONFEDERATION OF LABOR  
TON DUC THANG UNIVERSITY  
FACULTY OF INFORMATION TECHNOLOGY**



# **MIDTERM REPORT JAVA TECHNOLOGY**

## **Student Information Management**

*Instructor:* **Mr. MAI VAN MANH**

*Student:* **Ho Huu An – 521H0489**

**Do Minh Quan – 521H0290**

**Nguyen Hoang Phuc – 521H0509**

*Class* : **21H50301**

*Year* : **25**

**HO CHI MINH CITY, 2023**

**VIETNAM GENERAL CONFEDERATION OF LABOR  
TON DUC THANG UNIVERSITY  
FACULTY OF INFORMATION TECHNOLOGY**



# **MIDTERM REPORT JAVA TECHNOLOGY**

## **Student Information Management**

*Instructor:* **Mr. MAI VAN MANH**

*Student:* **Ho Huu An – 521H0489**

**Do Minh Quan – 521H0290**

**Nguyen Hoang Phuc – 521H0509**

*Class* : **21H50301**

*Year* : **25**

**HO CHI MINH CITY, 2023**

## ACKNOWLEDGEMENT

We would like to express our deepest appreciation to the incredible educator, Mai Van Manh, whose unwavering commitment to providing top-notch education has significantly contributed to the success of our group. Without your inspiring guidance and engaging teaching style, the development of our management application would have remained an elusive goal.

A special note of thanks is also extended to Ton Duc Thang University for fostering an environment conducive to learning and innovation. Your institution's commitment to academic excellence has played a pivotal role in shaping our educational experience.

Mai Van Manh, your passion for teaching has not only equipped us with the necessary knowledge but has also fueled our enthusiasm for exploring and implementing creative solutions. Your dedication has been a beacon, guiding us through the challenges of developing this application.

As we reflect on this journey, we are sincerely grateful for your invaluable support and mentorship. Thank you for being a pillar of inspiration and for making our learning adventure at Ton Duc Thang University an enriching and unforgettable experience.

*Ho Chi Minh city, 13<sup>th</sup> December, 2023*

*Author*

*(Sign and write full name)*

  
*Ho Huu An*

*Nguyen Hoang Phuc*



*Do Minh Quan*

## **THIS PROJECT WAS COMPLETED AT TON DUC THANG UNIVERSITY**

I hereby certify that this is my thesis project and was conducted under the guidance of Mr Mai Van Manh. The research content and results in this topic are honest and have not been previously published in any form. The data in the tables are collected by the author from various sources clearly stated in the reference section for analysis, comments, and evaluation.

Furthermore, this thesis includes comments, evaluations, and data from other authors and organizations, all of which are cited and annotated with their sources.

If any fraud is detected, I will fully take responsibility for the content of my thesis. Ton Duc Thang University is not liable for any copyright infringement or violation caused by me during the implementation process (if any).

*Ho Chi Minh city, 13<sup>th</sup> December, 2023*

*Author*


*(Sign and write full name)*



*Ho Huu An*



*Nguyen Hoang Phuc*



*Do Minh Quan*

## CONFIRMATION AND ASSESSMENT SECTION

### Instructor confirmation section

---

---

---

---

---

---

---

---

*Ho Chi Minh    December, 2023*

*(Sign and write full name)*

### Evaluation section for grading instructor

---

---

---

---

---

---

---

---

*Ho Chi Minh    December, 2023*

*(Sign and write full name)*

## SUMMARY

This report, under the guidance of Mr. Mai Van Manh from the Faculty of Information Technology at Ton Duc Thang University, delves into Java Swing and JDBC technologies. Completed at 8:30 AM on December 10th, 2023, this report owes its successful completion to Mr. Mai Van Manh's invaluable lessons.

The primary focus of this report is on Java Swing, a graphical user interface (GUI) framework for building desktop applications in Java. Java Swing facilitates the creation of dynamic and engaging desktop applications through its window, control, and event-driven programming features. Particularly applicable in scenarios like student information management systems, Java Swing provides a robust foundation for crafting intuitive user interfaces.

JDBC (Java Database Connectivity), on the other hand, takes the spotlight as a critical technology for interacting with databases in Java applications. Offering a standardized way to connect and manipulate databases, JDBC empowers developers to execute queries, updates, and transactions seamlessly. In the realm of student information management, JDBC can play a pivotal role in tasks such as retrieving, updating, and managing student-related data within databases.

It's essential to note that while this report endeavors to provide comprehensive insights, there may be potential errors present. The author welcomes constructive contributions from teachers, viewing them as valuable lessons for refining future articles. Continuous learning and improvement are intrinsic to the dynamic field of technology, and any feedback received is appreciated for enhancing the accuracy and quality of the content.

## TABLE OF CONTENTENCE

CONFIRMATION AND ASSESSMENT SECTION .....	iii
CHAPTER 1 – PROJECT INTRODUCTION.....	5
1.1 Management System Introduction .....	5
1.2 System Scope .....	5
1.3 System Description .....	7
CHAPTER 2 – THEORETICAL BASIC .....	8
2.1 JDBC Database Connection .....	8
2.1.1 What Is JDBC Java .....	8
2.1.2 JDBC’s Architecture .....	8
2.1.3 JDBC Driver.....	10
2.1.4 JDBC And ODBC Comparision .....	12
2.1.5 JDBC Implementation.....	13
2.2 Swing (Java) .....	17
2.2.1 What Is Swing (Java) .....	17
2.2.2 Swing Architecture.....	17
2.2.3 Swing And AWT Comparison .....	18
2.1.5 Java Swing Implementation .....	19
2.3 Main Library: .....	24
2.3.1 FlatLaf : .....	24
2.3.2 Timming Framework: .....	24
2.3.3 MigLayout:.....	25
2.3.4 Apache POI: .....	25
CHAPTER 3 – STUDENT INFORMATION MANAGMENT SYSTEM .....	27
3.1 Functionalities of the Project: .....	27

3.2 Uses Cases.....	28
3.3 Entity-Relationship Diagram .....	37
3.4 Demonstration .....	40
CHAPTER 4 – CONCLUSION .....	43
4.1 Result Achieved .....	43
4.2 Drawback .....	43
4.3 Future Improvement.....	44
WORK ASSIGNMENT AND MEMBERS EVALUATION.....	45
Task Assignment .....	45
Member evaluation.....	47
REFERENCE .....	48



## LIST OF TABLES, FIGURES, GRAPHS

### LIST OF FIGURES

Figure 1: Student Information Management Logo.....	5
Figure 2: JDBC Logo .....	8
Figure 3: Graph Showcasing JDBC Architecture .....	9
Figure 4: Graph Showcasing JDBC Driver Type.....	11
Figure 5: JDBC VS ODBC Image .....	12
Figure 6: Connection Config For Handeling Connection String .....	14
Figure 7: Connection Factory For Handeling Connection Factory .....	15
Figure 8: AddUser Method That Used JDBC .....	16
Figure 9: Swing Logo.....	17
Figure 10: Swing VS AWT Image .....	19
Figure 11: Jlabel Implementation.....	20
Figure 12: AWT Label Imprementation.....	21
Figure 13: Jbutton Implementation .....	22
Figure 14: AWT Button Implementation .....	23
Figure 15: Jspinnerbox And Jcombobox Implementation .....	24
Figure 16: General Uses Case Diagram .....	36
Figure 17: Entity Diagram.....	39
Figure 18: Login Interface.....	40
Figure 19: Student Management Interface .....	40
Figure 20: User Management Interface.....	41
Figure 21: Student Detail Interface .....	41
Figure 22: Add A New Student Interface.....	42
Figure 23: User Detail Interface .....	42

**LIST OF TABLES**

Table 1: ODBC VS JDBC Table.....	13
Table 2: AWT VS Swing Table .....	19
Table 3: Description about Interfaces .....	42
Table 4: Task Assignment.....	47
Table 5: Member Evaluation.....	47

## CHAPTER 1 – PROJECT INTRODUCTION

### 1.1 Management System Introduction

Student Information Management (SIM) is a carefully constructed application that makes it easier to manage student data, including grades, certificates, and personal data. Prioritizing effectiveness and ease of use, the system enables users to quickly add and modify student information, control user profiles, and designate specific roles to individuals according to their duties. The system's flexibility is further enhanced by the three-tiered user roles student, Manager, and Admin—which offer customized experiences for each user type. This all-inclusive design guarantees a smooth and intuitive method of handling student data in an educational setting.



*Figure 1: Student Information Management Logo*

### 1.2 System Scope

Student Information Management (SIM) system is comprised of the following parts:

- Login
- User Management
- Student Management
- Import export certificate
- role separation

At its core, SIM incorporates a secure and user-friendly login system, ensuring role-based access for Admin, Manager, and Employee. This foundational feature

establishes the groundwork for the comprehensive user management functionalities within the system.

Admin has access to a variety of powerful tools for user management. They can log in and navigate the system, changing profile pictures for all users and managing a comprehensive list of system users. The ability to add new user and capture relevant details such as name, age, phone number, and status (normal/locked) allows for greater flexibility in managing the user information. Furthermore, Admin has the authority to delete and modify user information, fostering a dynamic and adaptive user management environment.

When it comes to student management, SIM provides tools for both Admin and Managers to easily manipulate student data. SIM offers a comprehensive approach to student data management, from viewing a detailed list of students to seamlessly adding new entries and updating existing information. Sorting the student list based on different criteria and conducting searches with multiple parameters all contribute to a more efficient and organized workflow. Authorized users can access individual student information, providing a comprehensive view of their academic certificate.

SIM facilitates the import and export of student data and academic certificates, in addition to student management. This feature improves data accessibility and compatibility by allowing administrators and managers to import student and academic certificate lists from the database. The export feature ensures that data is presented in Excel/CSV format.

SIM includes a robust role separation feature that distinguishes three distinct user roles: Admin, Manager, and Employee. Admin accounts integrate into the system without any additional configuration. Administrators have complete control over all system functions, whereas Managers can perform actions related to student management. Employee accounts, which are intended for content viewing, are not allowed to edit data except to change their profile picture.

## 1.3 System Description

### Student Information Management System (SIM)

The Student Information Management System (SIM) is a useful tool carefully designed to facilitate student information Administration. This comprehensive system, which embodies a dedication to efficient and user-friendly operation, handles the management of grades, certificates, and personal data.

#### Admin:

Admin using SIM has total authority over how the system operates. Admin can do many different operations for managing users and students. This involves monitoring user logins, and doing necessary tasks like updating, deleting, or adding user and student information. Admin can modify their own profile pictures and import and export student data and academic qualifications

#### Manager:

Managers using SIM are equipped with a range of tools designed to make managing student affairs easier. Managers whose accounts are created by Admin, Managers can import and export student data and academic credentials, add and remove student records, and edit student information. Manager also can modify their profile image

#### Employee:

Employees using SIM are focused on simplicity and restriction. Employees whose accounts are typically created by Admin are limited to viewing system content. Their operations revolve around gaining access to information rather than manipulating it. Also, Employees can still change their profile picture

## CHAPTER 2 – THEORETICAL BASIC

### 2.1 JDBC Database Connection

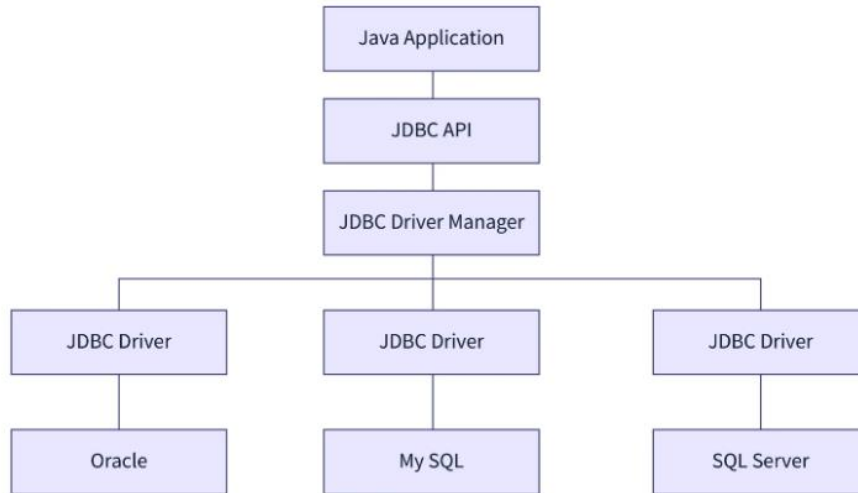
#### 2.1.1 What Is JDBC Java

Java Database Connectivity (JDBC) is the standardized application interface facilitating Java applications' interaction with Database Management Systems (DBMS). JDBC is adept at managing operations such as formulating queries and commands and processing result sets from the database. Initially confined to client-side functionalities, encompassing the handling of database interfaces, executing database calls, and retrieving data for client applications, JDBC has since evolved to encompass server-side capabilities. This extension involves the orchestration of database requests and the management of information transactions on the server. JDBC provides users with a diverse set of interfaces and classes, enabling them to manipulate and engage with the database seamlessly.



*Figure 2: JDBC Logo*

#### 2.1.2 JDBC's Architecture



*Figure 3: Graph Showcasing JDBC Architecture*

### **The JDBC interface consists of 5 major components:**

application: Applications assume the form of Java-based entities, encompassing applets or servlets, and engaging in communication protocols with databases.

the JDBC (Java Database Connectivity) API: Constitutes an integral application programming interface employed in the establishment of databases. Within the JDBC API framework, an array of classes and interfaces is employed to facilitate seamless connections with databases. Noteworthy entities within the JDBC architecture encompass the driver Manager class, the connection interface, among others.

The driver Manager class: Assumes a pivotal role in facilitating the establishment of connections between Java applications and databases. This class leverages the “getConnection” method, wherein the initiation of a connection between the Java application and data sources is realized.

JDBC drivers: Play a crucial role in enabling connectivity with diverse data sources. Databases such as Oracle, MSSQL, MYSQL, among others, possess unique drivers tailored for establishing connections. To engage with these databases, it becomes imperative to load their specific drivers. The instantiation of a Java class, known as "class," serves as a mechanism for loading these drivers within the JDBC architecture.

data sources: Refer to the databases accessible through JDBC API. These repositories serve as reservoirs where data is stored and subsequently utilized by Java applications. The JDBC API provides a standardized mechanism for establishing connections with diverse databases, including Oracle, MYSQL, MSSQL, PostgreSQL, and others. This ensures a uniform approach to interfacing with a variety of data sources within Java applications.

### *2.1.3 JDBC Driver*

The JDBC (Java Database Connectivity) API includes a diverse set of interfaces and classes designed to establish database connections and execute queries. A critical feature of the JDBC architecture is its adaptability, wherein a driver can implement these interfaces and classes to align with the requirements of a particular database management system.

In the operational sequence, the JDBC API initiates the loading of a specific database driver before the establishment of a database connection. This loading process is a prerequisite, allowing the subsequent interaction with the database Manager to relay all API calls to the loaded driver. This dynamic loading mechanism ensures the seamless integration and compatibility of the JDBC API with various database management systems by accommodating specific driver implementations tailored to each system's nuances.



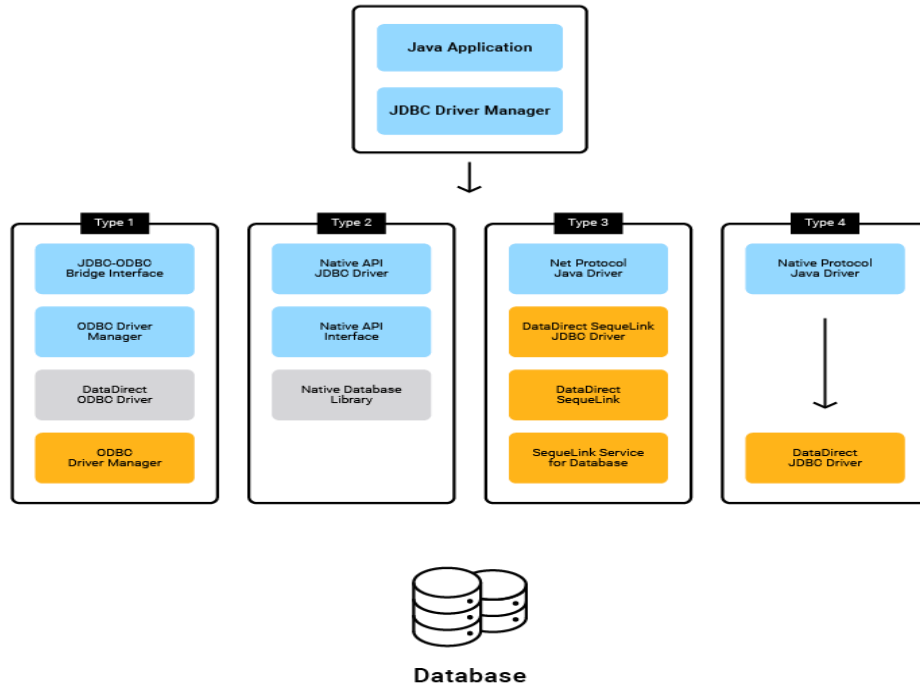


Figure 4: Graph Showcasing JDBC Driver Type

There are four types of JDBC drivers:

**Type 1 driver:** The JDBC-ODBC bridge, coupled with an ODBC driver (Type 1 Driver), functions as an intermediary for the translation of JDBC API calls into corresponding ODBC API calls. This process mandates loading binary code onto client machines employing this driver. Primarily, this driver finds its common application in the testing phase of JDBC applications, particularly when assessing their functionality against an ODBC data source.

**Type 2 driver:** Native-API, partly Java driver converts the API call of the JDBC driver into specific databases management system (DBMS) specific API call resembling Type 1 driver. This requires the binary code of the DBMS API to be loaded into each client computer for testing JDBC applications against a DBMS API data source.

**Type 3 driver:** JDBC-Net, is a pure-Java implementation that manages the translation of API calls by routing them through an intermediary middle management

server. Subsequently, this intermediary server undertakes an additional translation of the calls into a protocol specific to the management of the targeted database. An illustrative example of a Type 3 JDBC driver is the DataDirect Sequel ink JDBC driver.

Type 4 driver: The Native-Protocol, Pure-Java driver, executes the translation of API calls directly into a management-specific protocol, eliminating the need for an intermediary middle server. This configuration enables the client application to establish a direct connection with the database server. Type 4 drivers are notably prevalent and are typically tailored for a specific vendor's database system.

#### *2.1.4 JDBC And ODBC Comparision*

JDBC and ODBC share a historical connection, as JDBC was originally influenced by the Open Database Connectivity (ODBC) standard. ODBC, a language-agnostic approach to database connectivity, allowed interaction with relational database management systems using various programming languages, each with its own syntax. In contrast, JDBC departed from this approach, concentrating exclusively on the Java programming language. Despite their historical similarities, there are notable differences between JDBC and ODBC.



*Figure 5: JDBC VS ODBC Image*

ODBC	JDBC
Develop and introduce by Microsoft back in the 1992	Develop and introduce by Introduced by SUN Micro Systems in 1997.
. ODBC can only be used on window bases platform	JDBC can be used on all platform
Using ODBC on java application will result in it becoming slower due to internal conversion and the application will become platform dependance	It is highly advised to used JDBC for java application because there are no internal conversion and platform dependance
ODBC is procedure oriented	JDBC is object oriented

*Table 1: ODBC VS JDBC Table*

### *2.1.5 JDBC Implementation*

In the implementation of our Student Information Management application, our group has chosen the Type 4 driver, commonly known as the Native-Protocol driver. The rationale behind this decision lies in the inherent capability of the Type 4 driver to establish a direct connection between our client application and the MySQL server without the need for additional binary code or an intermediary server. In this configuration, the database seamlessly converts JDBC API calls to a protocol specific to the MySQL Database Management System (DBMS), ensuring an efficient and straightforward communication channel between our client application and the MySQL server. This choice contributes to the overall simplicity and effectiveness of our database connectivity approach.

#### **“ConnectionConfig” Class:**



```
1 // ConnectionConfig.java
2 public class ConnectionConfig {
3
4     public static final String DRIVER_CLASS_NAME = "com.mysql.cj.jdbc.Driver";
5     public static final String CONNECTION_URL = "jdbc:mysql://localhost:3306/studentmanagementinformation?user=root";
6     public static final String DATABASE_NAME = "lab2";
7     public static final String DB_USER = "root";
8     public static final String DB_PASSWORD = "";
9
10 }
11
```

*Figure 6: Connection Config For Handling Connection String*

The "ConnectionConfig" class serves as a configuration container for database connection parameters. It defines several constants related to the MySQL database, such as the JDBC driver class name ("DRIVER\_CLASS\_NAME"), the connection URL ("CONNECTION\_URL"), the name of the database ("DATABASE\_NAME"), and the database user credentials ("DB\_USER" and "DB\_PASSWORD"). These constants provide a centralized place for managing and modifying database connection details.

### **“ConnectionFactory” Class:**



```

1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.SQLException;
4
5  public class ConnectionFactory {
6      private static ConnectionFactory connectionFactory = null;
7
8      private ConnectionFactory() {
9          try {
10             Class.forName(ConnectionConfig.DRIVER_CLASS_NAME);
11         } catch (ClassNotFoundException e) {
12             e.printStackTrace();
13         }
14     }
15     public static Connection getConnection() throws SQLException {
16         Connection conn = null;
17         String connectionUrl = ConnectionConfig.CONNECTION_URL + ConnectionConfig.DATABASE_NAME;
18         conn = DriverManager.getConnection(connectionUrl, ConnectionConfig.DB_USER, ConnectionConfig.DB_PASSWORD);
19         return conn;
20     }
21     public ConnectionFactory getInstance() {
22         if (connectionFactory == null) {
23             connectionFactory = new ConnectionFactory();
24         }
25         return connectionFactory;
26     }
27 }

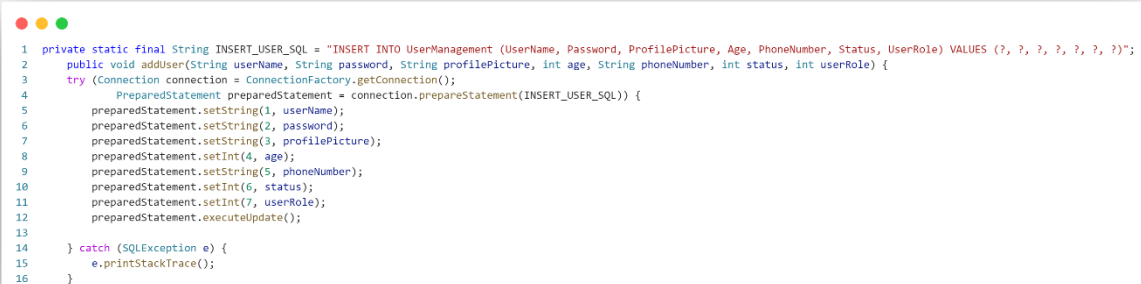
```

*Figure 7: Connection Factory For Handling Connection Factory*

The "ConnectionFactory" class is designed to handle the creation of database connections. It employs the Singleton pattern to ensure that only one instance of the "ConnectionFactory" is created. In its constructor, the class dynamically loads the MySQL JDBC driver using "Class.forName()". The "getInstance" method ensures that only a single instance of the "ConnectionFactory" is created and returned.

The "getConnection" method is responsible for obtaining a "Connection" object to the database. It constructs the connection URL by combining the base URL from "CONNECTION\_URL" with the specific database name from "DATABASE\_NAME". Then, it uses "DriverManager.getConnection()" to establish a connection, utilizing the database user credentials from "DB\_USER" and "DB\_PASSWORD".

### **“UserDao” Class With “addUser” Method:**



```

1 private static final String INSERT_USER_SQL = "INSERT INTO UserManagement (UserName, Password, ProfilePicture, Age, PhoneNumber, Status, UserRole) VALUES (?, ?, ?, ?, ?, ?, ?)";
2 public void addUser(String userName, String password, String profilePicture, int age, String phoneNumber, int status, int userRole) {
3     try (Connection connection = ConnectionFactory.getConnection();
4         PreparedStatement preparedStatement = connection.prepareStatement(INSERT_USER_SQL)) {
5         preparedStatement.setString(1, userName);
6         preparedStatement.setString(2, password);
7         preparedStatement.setString(3, profilePicture);
8         preparedStatement.setInt(4, age);
9         preparedStatement.setString(5, phoneNumber);
10        preparedStatement.setInt(6, status);
11        preparedStatement.setInt(7, userRole);
12        preparedStatement.executeUpdate();
13    }
14    } catch (SQLException e) {
15        e.printStackTrace();
16    }

```

*Figure 8: AddUser Method That Used JDBC*

The "addUser" method within the main class facilitates the insertion of a new user into the database. It takes various user-related parameters such as username, password, profile picture, age, phone number, status, and user role. Inside the method, a "Connection" object is obtained by invoking the "ConnectionFactory.getConnection()" method.

Subsequently, a "PreparedStatement" is created using the SQL query stored in the constant "INSERT\_USER\_SQL". The parameters for the query are then set using the provided user information. The "executeUpdate()" method is used to execute the SQL update query, which inserts the user information into the "UserManagement" table.

To ensure proper resource management, a try-with-resources block is employed for both the "Connection" and "PreparedStatement". This guarantees that these resources are closed appropriately, even in the event of an exception. If a "SQLException" occurs during the database interaction, it is caught, and the stack trace is printed.

The SQL query, represented by the constant "INSERT\_USER\_SQL", specifies the structure for inserting a new user into the "UserManagement" table with the appropriate column names.

### **Overall Flow:**

The process begins with the "ConnectionFactory" handling the database connection setup. The "addUser" method utilizes this connection to create and execute

a SQL query for inserting a new user into the database. The entire operation is encapsulated within a try-with-resources block for effective resource management, and any potential SQL exceptions are caught and printed for debugging purposes. This structure promotes code modularity and separation of concerns for managing database connections and user data insertion.

## 2.2 Swing (Java)

### 2.2.1 What Is Swing (Java)

Swing is a graphical user interface (GUI) toolkit developed by Oracle as part of the Java Foundation Classes (JFC). Its application programming interface (API) serves as a framework for the creation and development of graphical user interfaces in Java applications. Swing constitutes a platform-independent Model-View-Controller (MVC) framework in Java, characterized by a single-threaded programming model. This framework highlights the abstraction of both the code structure and the graphical presentation within a Swing-based graphical user interface (GUI).



Figure 9: Swing Logo

### 2.2.2 Swing Architecture

**Foundations:** Swing, being exclusively implemented in Java, exemplifies platform agnosticism. Platform agnosticism means that it can run on a system independent of the underlying hardware or software platform.

**Extensible:** One of Swing's unique features is its modular-based architecture, which enables the smooth integration of several custom implementations specified by framework interfaces. This built-in flexibility is further highlighted by the ability for

users to provide their own implementations, going beyond default settings by utilizing Java's inheritance process made possible by Look And Feel.

**Lightweight UI:** Swing is characterized as a lightweight graphical user interface (GUI) toolkit due to its complete implementation in the Java programming language, removing the necessity to invoke the native operating system for rendering graphical interface components. In contrast, the Abstract Window Toolkit (AWT) is deemed a heavyweight toolkit, as it relies on direct calls to the operating system to generate its GUI components.

**Configurable:** The Swing framework's configurability highlights its capacity to flexibly adjust to runtime changes in its configuration. Swing relies heavily on indirect composition patterns and runtime techniques to accomplish this flexibility. Notably, a Swing program can change its user interface while it is still running; this is known as "hot swapping" during runtime. Furthermore, users can present their own customized versions of Swing applications' appearance and feel. This suggests that users can alter the aesthetic look and feel of pre-existing Swing apps without having to modify the application code.

### *2.2.3 Swing And AWT Comparison*

User interface component platform-independent APIs have been provided by the Abstract Window Toolkit (AWT) from the early versions of Java. A native peer component customized for the underlying windowing system is responsible for managing and displaying the visual representation of every component inside the AWT framework. The Swing API is not a straight replacement for the AWT; rather, it is an additional extension of it. It is interesting to note that every Swing light interface is contained within an AWT heavyweight component. As can be seen in the inheritance structure, an AWT top-level container is extended by all top-level Swing components, including JApplet, JDialog, JFrame, and JWindow.





*Figure 10: Swing VS AWT Image*

differences between AWT and Swing:

AWT	Swing
Java AWT have heavy weight component	Swing has light weight component
Model view controller is not supported by AWT	Model view controller is supported by Swing
AWT take up more running time than Swing	Swing takes up less running time than AWT
AWT is platform dependance	Swing is platform independence
AWT have less component than Swing	Swing have much more component

*Table 2: AWT VS Swing Table*

### *2.1.5 Java Swing Implementation*

While developing our Student Information Management system, our group observed a notable disparity in functionality between the JLabel in AWT and Java Swing. The JLabel in Java Swing afforded us significantly greater control over the content type, allowing for the incorporation of icons and the utilization of background images, in contrast to the more limited capabilities of its AWT counterpart, which primarily supports text manipulation and font adjustments. Leveraging the built-in components of Java Swing facilitated the seamless integration of additional elements, such as JSpinner and JComboBox, enhancing the overall functionality of the application. This was particularly advantageous when implementing features like updating and

adding information through the JDialogs. Choosing to exclusively employ AWT would have resulted in the omission of these advanced functionalities from our application.

### Java Swing Label:

In Java Swing, the "Properties" tab of a label offers a feature that allows the addition of an icon to the label. This functionality enables me to incorporate a background image, such as "certificate.jpg," to enhance the appearance of my Certificate View JDialog.

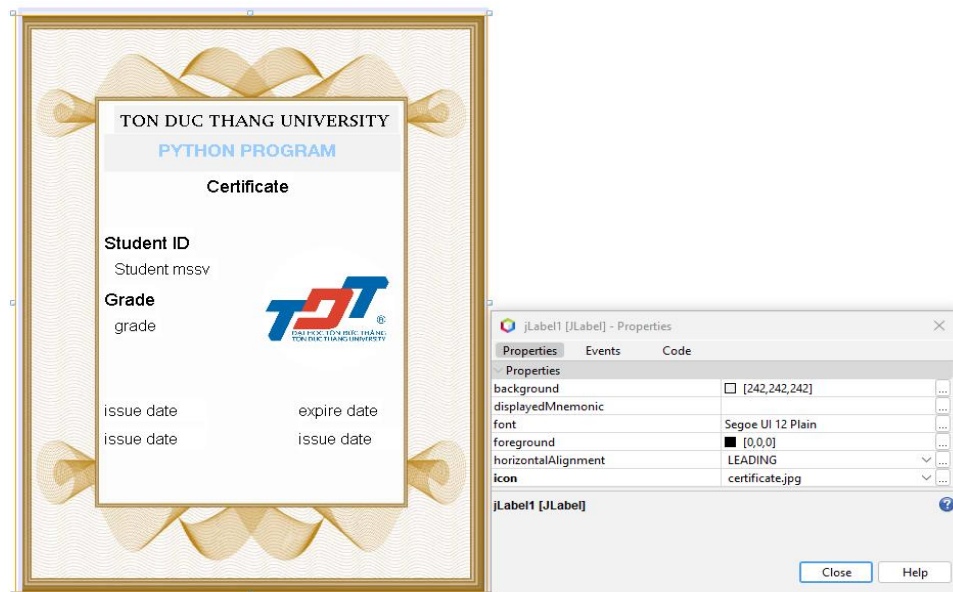


Figure 11: JLabel Implementation

In contrast to labels in AWT, which are constrained to displaying text with limited formatting options, such as adjusting text content, font, and size, there is a notable drawback. AWT labels lack the capability to seamlessly incorporate images or icons, limiting their versatility in creating visually enriched components.

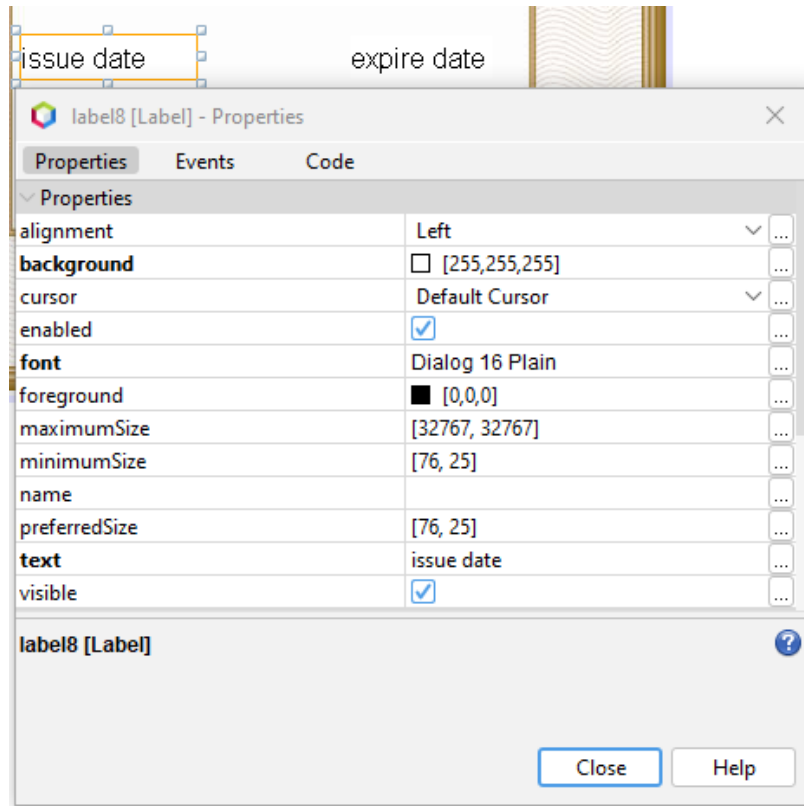


Figure 12: AWT Label Implementation

### Java Swing JButton:

The custom Actionbutton, inheriting its functionality from JButton in Java Swing, significantly simplifies the process of incorporating icons. Leveraging the inherent icon functionality of JButton has notably facilitated the creation of a more intuitive and visually appealing user interface, allowing me to seamlessly add icons such as TableViewAction.png.

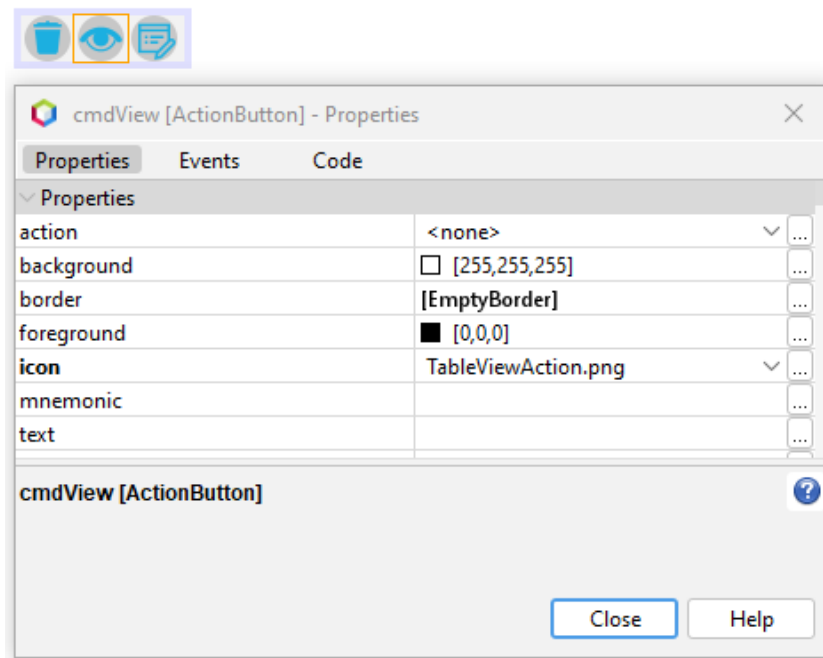
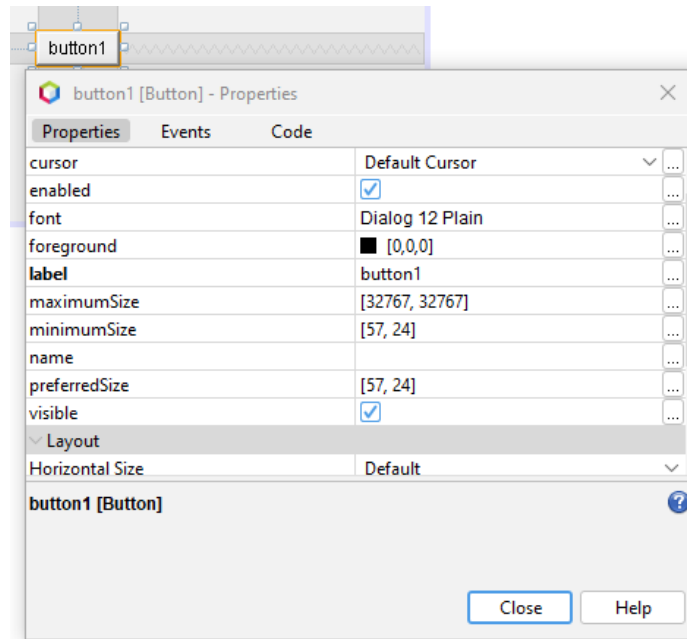


Figure 13: JButton Implementation

This stands in contrast to AWT buttons, limiting editing capabilities to the button text, with functionality restricted to adjusting text font and layout. The absence of built-in support for versatile icon integration in AWT buttons further distinguishes their limitations compared to Java Swing components.



*Figure 14: AWT Button Implementation*

### **Java Swing Jspinnerbox And Jcombobox :**

Java Swing, in addition to its enhanced support for incorporating images or icons in labels, offers a diverse set of components tailored to meet project-specific requirements. For example, components like JComboBox, suitable for storing years, and JSpinner, designed for selecting the age of the user, provide increased flexibility and functionality for your project needs. These components contribute to a more comprehensive and tailored user interface.

The image shows a Java Swing window with a light gray background. It contains four components arranged in a 2x2 grid. The top-left component is labeled 'PhoneNumber' in blue text and is a text field. The top-right component is labeled 'Age' in blue text and is a spinner box showing the value '5'. The bottom-left component is labeled 'Status' in blue text and is a dropdown menu showing 'Normal'. The bottom-right component is labeled 'UserRole' in blue text and is a dropdown menu with 'Manager' selected and 'Employee' as an option.

Figure 15: Jspinnerbox And Jcombobox Implementation

## 2.3 Main Library:

### 2.3.1 FlatLaf :

FlatLaf is a modern open-source cross-platform Look and Feel for Java Swing desktop applications. It appears flat (no gradients or shadows), clean, simple, and elegant. FlatLaf includes Light, Dark, IntelliJ, and Darcula themes, scales on HiDPI displays, and requires Java 8 or later to run. The design is heavily influenced by the Darcula and IntelliJ themes from IntelliJ IDEA 2019.2+, with identical colors and icons.

In the context of our group Student Information Management application, the utilization of FlatLaf has been instrumental in enhancing the user interface (UI). Absent the integration of FlatLaf, the application would default to the standard Java Swing user interface.

### 2.3.2 Timming Framework:

The Timing Framework Swing is a specialized library designed to make it easier to implement Java animations and timing-based controls in Swing, Java's graphical user interface toolkit, and SWT (Standard Widget Toolkit). Its primary goal is to simplify the complex process of animating components and managing time in Swing applications. This framework enables developers to create fluid animations, smooth transitions, and engaging visual effects with ease. The ability to define animations based on time intervals, seamless integration with Swing for incorporating animations into user

interfaces, and the provision of sample code for a quick start in animation development are key features.

In our Student Information Management Application, the Timing Framework has proven invaluable for precisely timing animations and preventing visual errors. For instance, when the sidebar shrinks, it intelligently waits for the ongoing animation to finish before readjusting, effectively averting animation-related issues and ensuring a seamless user experience.

### *2.3.3 MigLayout:*

MigLayout is a powerful Java/Swing layout manager that streamlines user interfaces with grid-based layouts, docking, grouping, precise spacing, in-cell flow, and visual bounds. It integrates seamlessly with Swing, SWT, and JavaFX, making it ideal for developers looking for simplicity, power, and platform fidelity. MigLayout provides simple maintenance, clear source code comprehension, and a wide range of layout options, allowing you to create flowing, grid-based, absolute, grouped, and docking layouts with concise and straightforward code.

In our Student Information Management Application, MigLayout significantly enhances the formatting of the main panel's layout. It efficiently allocates space for each element, ensuring proper placement for an aesthetically pleasing user interface. For instance, I can position the header at the top and the sidebar on the right, creating an organized layout. Additionally, MigLayout simplifies animation tasks by automatically adjusting the content panel's size based on percentages rather than manually resizing it. This adaptive approach ensures that if the sidebar shrinks, the content panel dynamically adjusts to fit the available space.

### *2.3.4 Apache POI:*

Apache POI is a powerful Java API created by the Apache Foundation for handling various file formats found in Microsoft Office applications. It makes it easier to read, write, and manipulate Excel spreadsheets, Word documents, and PowerPoint presentations. Support for both Office Open XML (OOXML) and OLE 2 Compound Document formats is important, as is the ability to work with cell data, formulas, and styles in Excel, create/modify Word documents, and interact with PowerPoint

presentations. The project's goal is to keep Java APIs for various file formats used in Microsoft Office up to date.

In our Student Information Management Application, Apache POI simplifies the import and export of Excel files. Instead of manually coding these functions, we leverage the library to handle Excel file formats. This allows us to focus on the logic aspects, streamlining the process of handling Excel data seamlessly



## **CHAPTER 3 – STUDENT INFORMATION MANAGMENT SYSTEM**

### **3.1 Functionalities of the Project:**

The system shall support user login functionality for all registered users. The user role will be assigned based on the user login password and name.

Users can update their profile picture.

#### **User Management (Admin):**

Admins can change the profile picture for all users.

Admins will be able to view the list of system users.

Admins will be able to add new users, specifying details such as Name, Age, Phone Number, and Status (Normal/Locked).

Admins shall be able to delete user accounts.

#### **Student Management:**

Admins and Managers, shall be able to view the list of students

Admin and Managers can add new students to the system, providing details such as Name, Age, and Phone Number

Admins and Managers can delete student records from the system

Admins and Managers shall have the ability to update student information, including modifying details like Name and Age

Admins and Managers shall be able to sort the list of students based on various criteria

Admins and Managers can search for students using multiple criteria

Admins and Managers can access a detailed student details page, viewing complete student information

Admins and Managers can manage a list of certificates for students, including viewing the list, adding new certificates, deleting certificates, and updating certificate information

### **Data Import/Export**

Admins and Managers shall have the ability to import a list of students from a file

Admins and Managers shall be able to export the list of students to Excel/CSV files

Admins and Managers have the capability to import a list of academic qualifications for students from a file

Admins and Managers be able to export the list of academic qualifications to Excel/CSV files

### **User Roles:**

The system shall provide three user roles: Admin, Manager, and Employee

The Admin account is integrated with the system and does not require creation

Admins shall have full access to system functions, Managers can perform all functions related to students, and Employees can only view content Employees are restricted from editing any content other than updating their profile picture

## **3.2 Uses Cases**

### **Use Case 1: User Login**

Actors: Admin, Manager, Employee

Description: Users log in to the system.

Preconditions: The system is accessible, and the user has valid credentials.

Postconditions: Users gain access to the system.

Main Flow:

- Users initiate the login process.
- The system prompts for username and password.
- Users provide valid credentials.

- The system verifies the credentials.
- If authentication is successful, users are granted access; otherwise, an error message is displayed.

### **Use Case 2: Change Profile Picture**

Actors: Admin, Manager, Employee

Description: Users change their profile pictures.

Preconditions: Users are logged into the system.

Postconditions: The user's profile picture is updated.

Main Flow:

- Users navigate to the profile settings.
- The system displays the current profile picture.
- Users select an option to upload a new picture.
- Users choose a new image file.
- The system updates the profile picture.

### **Use Case 3: View List of Students**

Actors: Admin, Manager, Employee

Description: Users view the list of all students in the system.

Preconditions: Users are logged into the system.

Postconditions: Users have an overview of all students.

Main Flow:

- Users navigate to the student management section.
- The system retrieves and displays the student list in a table.

### **Use Case 4: Access Student Details Page**

Actors: Admin, Manager, Employee

Description: Users

access a detailed page containing information about a specific student.

Preconditions: Users are logged into the system.

Postconditions: Users view comprehensive details about the selected student.

Main Flow:

- Users navigate to the student management section.
- Users select the option to access student details.
- Users choose the student to view.
- The system displays a detailed page with complete student information.

### **Use Case 5: Student Management**

Actors: Manager, Admin

Description: Users are responsible for managing student information, including updating, deleting, adding new students, importing student data from an Excel file, and exporting student data to an Excel file.

Preconditions: The Manager or Admin is authenticated and logged into the Student Management application.

Postconditions: The student data in the system is appropriately modified, added, or exported based on the actions performed by the Manager or Admin.

Main Flow:

#### **Update Student Information:**

- The Manager or Admin navigates to the “Student” section.
- Selects the student to be updated.
- Modifies the relevant student information (e.g., Name, Age, Phone Number).
- Submits the updated information.

- The system validates and updates the student details.

#### **Delete Student:**

- The Manager or Admin navigates to the “Student” section.
- Chooses the student to be deleted.
- Confirms the deletion.
- The system removes the selected student from the database.

#### **Add New Student:**

- The Manager or Admin navigates to the “Student” section.
- Fill in the required information for the new student (Name, Age, Phone Number).
- Submit the form.
- The system validates the information and adds the new student to the system.

#### **Import Excel File:**

- The Manager or Admin goes to the “Report” section.
- Selects the option to import data from an Excel file.
- Uploads the Excel file containing student information.
- The system validates and adds the new student data from the file.

#### **Export Excel File:**

- The Manager or Admin goes to the " Report " section.
- Selects the option to export student data to an Excel file.
- Choose the destination for the exported file.
- The system generates the Excel file with the student data.

### **Use Case 6: Certificate Management**

Actors: Manager, Admin

Description: Users are responsible for managing certificate information, including updating, deleting, adding new certificates, importing certificate data from an Excel file, and exporting certificate data to an Excel file.

**Preconditions:**

The Manager or Admin is authenticated and logged into the Certificate Management application.

The user must have access to the student detail page.

Postconditions: The certificate data in the system is appropriately modified, added, or exported based on the actions performed by the Manager or Admin.

**Main Flow:****Update Certificate Information:**

- The Manager or Admin navigates to the "Student detail" section.
- Selects the certificate to be updated.
- Modifies the relevant certificate information (e.g., Certificate Name, Issuing Authority).
- Submits the updated information.
- The system validates and updates the certificate details.

**Delete Certificate:**

- The Manager or Admin navigates to the " Student detail" section.
- Choose the certificate to be deleted.
- Confirms the deletion.
- The system removes the selected certificate from the database.

**Add New Certificate:**

- The Manager or Admin navigates to the " Student detail " section.
- Fills in the required information for the new certificate (Certificate Name, Issuing Authority).
- Submit the form.
- The system validates the information and adds the new certificate to the system.

**Import Excel File:**

- The Manager or Admin goes to the " Report " section.
- Selects the option to import data from an Excel file.

- Uploads the Excel file containing certificate information.
- The system validates and adds the new certificate data from the file.

#### **Export Excel File:**

- The Manager or Admin goes to the " Report" section.
- Selects the option to export certificate data to an Excel file.
- Choose the destination for the exported file.
- The system generates the Excel file with the certificate data.

### **Use Case 7: User Management**

Actor: Admin

Description Users is responsible for managing user information, including updating, deleting, adding new users, and viewing the list of system users. The User Management system allows for the additional functionality of viewing login history.

Preconditions: The Admin is authenticated and logged into the User Management application.

Postconditions: The user data in the system is appropriately modified, added, or deleted based on the actions performed by the Admin.

Main Flow:

#### **Update User Information:**

- The Admin navigates to the "User" section.
- Selects the user to be updated.
- Modifies the relevant user information (e.g., Name, Age, Phone Number, Status).
- Submits the updated information.
- The system validates and updates the user details.

#### **Delete User:**

- The Admin navigates to the "User" section.
- Chooses the user to be deleted.

- Confirms the deletion.
- The system removes the selected user from the system.

#### **Add New User:**

- The Admin navigates to the " User " section.
- Fills in the required information for the new user (Name, Age, Phone Number, Status).
- Submits the form.
- The system validates the information and adds the new user to the system.

#### **View List of System Users:**

- The Admin navigates to the " User " section.
- The system retrieves and displays the list of all system users.

#### **View Login History of a User:**

- The Admin navigates to the " User " section.
- Selects the user for whom the login history needs to be viewed.
- The system displays the login history for the selected user.

### **Use Case 8: Search for Students Using Multiple Criteria**

Actors: Admin, Manager, Employee

Description Users initiate a search for students based on multiple criteria.

Preconditions: Users are logged into the system.

Postconditions: Users receive the search results.

Main Flow:

- Users navigate to the student management section.
- input the search query with multiple criteria.
- The system displays the search results in a table.



**Use Case 9: Sort Students Using Multiple Criteria**

**Actors:** Admin,Manager,Employee

**Description:** Users initiate sorting of students by selecting sort criteria from a combo box. The system displays the sorted results on the system table.

**Preconditions:** Users are logged into the system.

**Post Condition:** Users receive the sorted results based on the selected criteria.

**Main Flow:**

- Users navigate to the student management section.
- Users select the sorting criteria from the combo box.
- The system sorts the list of students based on the selected criteria.
- The system displays the sorted results in a table.

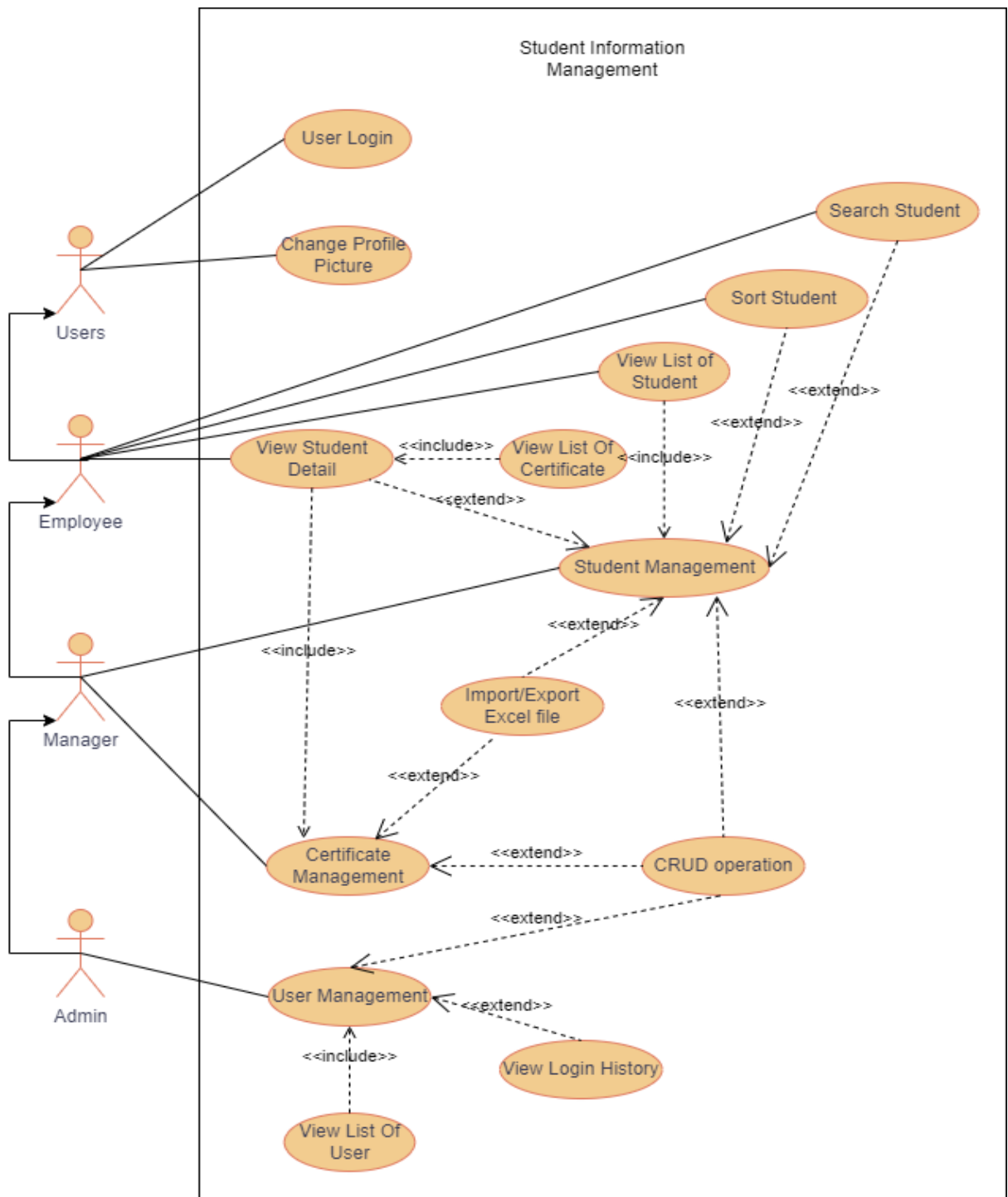


Figure 16: General Uses Case Diagram

### 3.3 Entity-Relationship Diagram

Student Table:

Attributes:

- ID: Primary Key, uniquely identifies each student.
- Name: Stores the name of the student.
- BeginningYear and EndYear: Capture the academic years of the student.
- Major: Represents the major of the student.
- EducationQuality: Indicates the quality of education.
- Phone: Stores the phone number of the student.

Reasoning:

The Student table serves as the core entity to store comprehensive information about each student. The ID acts as the primary key to uniquely identify each student. The cascading delete constraint ensures that when a student is deleted, all associated certificates are also deleted, maintaining data integrity.

UserManagement Table:

Attributes:

- UserID: Primary Key, uniquely identifies each user.
- UserName: Stores the username of the user.
- Password: Stores the hashed password for security.
- ProfilePicture: Stores the path to the user's profile picture.
- Age and PhoneNumber: Capture additional user details.
- Status: Represents whether the user is blocked or not (0: Normal, 1: Blocked).
- UserRole: Represents the role of the user (0: Admin, 1: Manager, 2: Employee).

Reasoning:

The UserManagement table is the central repository for handling user-related information, where each user is uniquely identified by their UserID, designated as the primary key. Ensuring data integrity, a cascading delete constraint is implemented. This

constraint guarantees that when a user is deleted, all associated login history records are also removed.

Two essential attributes, Status and UserRole, play crucial roles in user management. The Status attribute allows Admin to block users, controlling their access to the system. On the other hand, the UserRole attribute facilitates the assignment of different roles to users, dictating their access levels and permissions within the system.

LoginHistory Table:

Attributes:

- LoginHistoryID: Primary Key, uniquely identifies each login history record.
- UserID: Foreign Key, referencing the UserID in the UserManagement table.
- TimeLogin: Records the timestamp of user logins.

Reasoning:

The LoginHistory table is responsible for tracking the login history of users. Each login event is uniquely identified by the LoginHistoryID, serving as the primary key for these records. Establishing a foreign key relationship with the UserID allows for a one-to-many connection, enabling multiple login history records for each user.

In the context of data consistency, a crucial mechanism is in place: when a user is deleted, the associated login history records are also deleted. This ensures that the database maintains a coherent and consistent state, avoiding orphaned records and preserving the interconnected relationship between user data and login history.

CertificateManagement Table:

Attributes:

- CertificateID: Primary Key, uniquely identifies each certificate.
- StudentID: Foreign Key, referencing the ID in the Student table.
- CertificateName: Stores the name of the certificate.
- IssueDate,: The date in which the certificate is issue
- ExpiryDate: The date in which the certificate is expired

- Grade: The grade of the certificate

Reasoning:

The CertificateManagement table serves as the repository for managing information related to certificates. Each certificate is uniquely identified by the CertificateID, which acts as the primary key for these records. Creating a foreign key relationship with StudentID establishes a link to the Student table, ensuring that certificates are associated with specific students.

To uphold data integrity, a crucial mechanism is implemented: when a student is deleted, the associated certificates are also deleted. This ensures that the Database maintains data consistency by preventing orphaned records and preserving the interconnected relationship between student data and certificates.

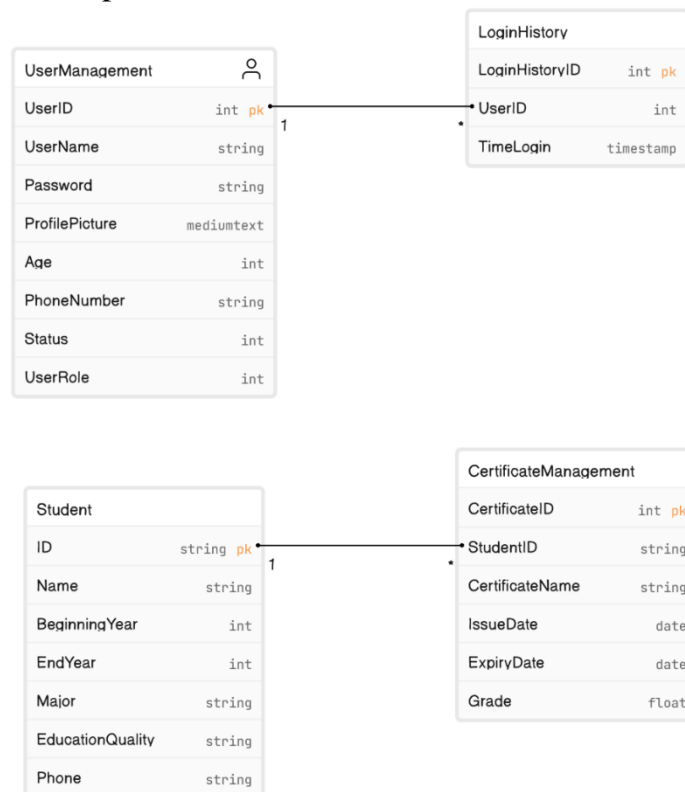
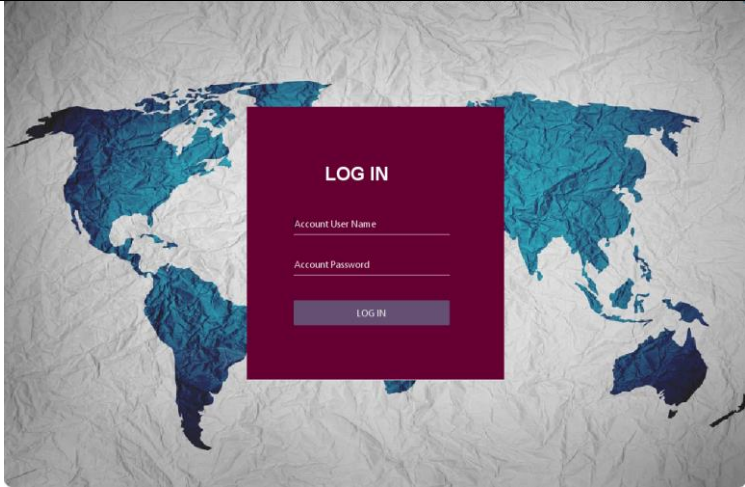
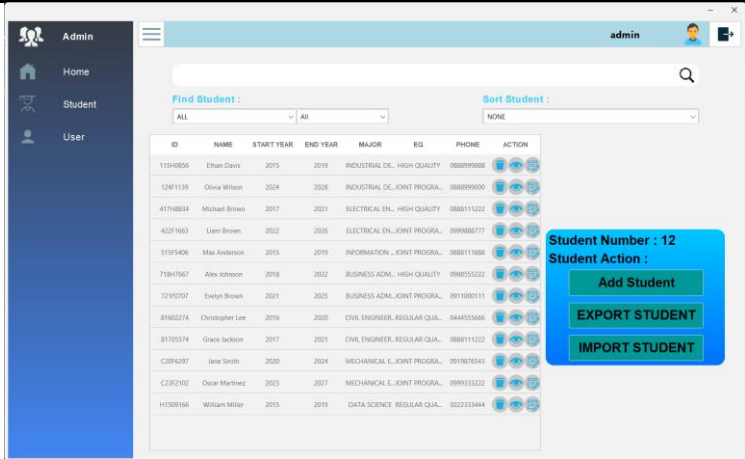


Figure 17: Entity Diagram

### 3.4 Demonstration

Specific Description	User Interface
<p><b>Login Interface:</b></p> <p>This interface serves as the authentication gateway, facilitating user access to the system through the provision of individualized credentials. To gain entry, users are required to enter both their designated username and password</p>	 <p><i>Figure 18: Login Interface</i></p>
<p><b>Students Management Interface:</b></p> <p>In this section, the management student operation is used. Various crud operations can be used to modify student information, and users can search for students using various criteria. Beside that, Export or import a list of student is a approach handle data student</p>	 <p><i>Figure 19: Student Management Interface</i></p>

### User Management Interface:

In this section, the management user operation is used. Various crud operations can be used to modify user information, and users can found bases on different criterial

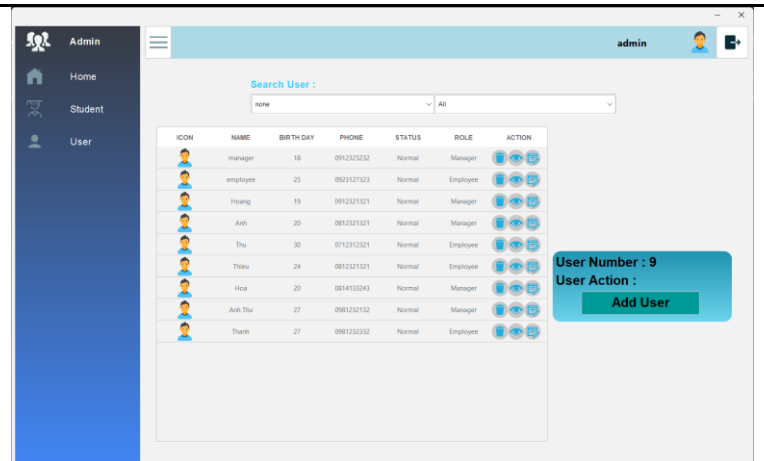


Figure 20: User Management Interface

### Student detail Management Interface:

In the student detail section, the user can see the student's details and the student certificate. In this section, the user can perform CRUD operations on the student certificate.

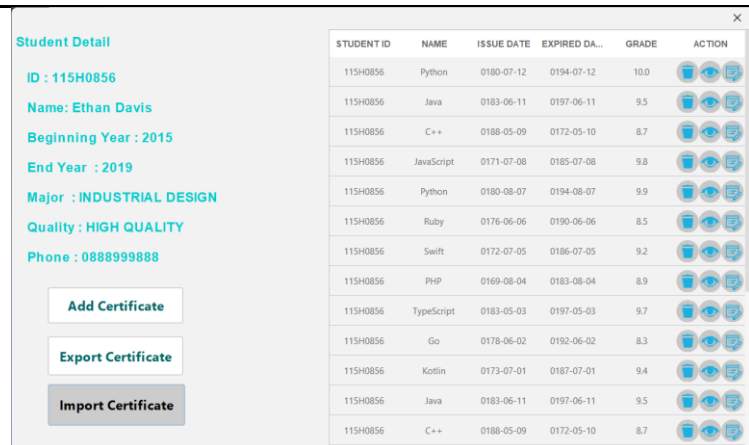


Figure 21: Student Detail Interface


<p>Add A New Student Interface:</p> <p>The user can enter all fields of student such name of student, phone, major, .. and enter submit button in order to add a new student</p>	<div><div><div>Student ID : None</div><div><div>StudentName</div><div>Phone</div></div><div><div>Major</div><div>Education Quality</div></div><div><div>Beginning Year</div><div>End Year</div></div><div>Submit</div></div></div> <p>Figure 22: Add A New Student Interface</p>
<p>User Detail Interface:</p> <p>The user can be added after entering full of fields of data relevanting to user or saved after editing.</p>	<div><div><div></div><div><div>UserName</div><div>Password</div></div><div><div>PhoneNumber</div><div>Age</div></div><div><div>Status</div><div>UserRole</div></div><div>Submit</div></div></div> <p>Figure 23: User Detail Interface</p>

Table 3: Description about Interfaces



## **CHAPTER 4 – CONCLUSION**

### **4.1 Result Achieved**

The system has effectively implemented all specified requirements, facilitating the comprehensive management of user information. Managers can proficiently oversee and manipulate relevant user data, Admin can manage all aspects of user information, and Employees are constrained to view and update their profile pictures exclusively. The seamless integration of a role-based authentication mechanism has been successfully implemented, ensuring a secure and precisely defined access structure within the system.

### **4.2 Drawback**

Java Swing is known for its traditional and somewhat outdated look and feel. The UI components provided by Swing may not meet modern design expectations, making it challenging to create visually appealing and contemporary user interfaces. This can result in a less engaging experience for users accustomed to more modern application designs.

In the era of web-based applications, Swing is primarily designed for desktop applications. If there is a requirement to integrate the student management system with web technologies or provide online access to users, Swing may not be the most suitable choice. Alternative technologies, such as JavaFX or web-based frameworks, might offer better support for these scenarios.

As the student management system grows in terms of data and functionalities, scalability becomes a concern. Swing applications might face challenges in handling large datasets or providing efficient data processing capabilities compared to more modern frameworks designed with scalability in mind.

The system's limitation lies in its relative simplicity, lacking additional functionalities that could enhance the management system. There is a notable absence of advanced features, such as a sophisticated search bar to facilitate the retrieval of student information based on complex criteria. Additionally, the system lacks a dedicated section for presenting visual representations, such as charts or graphs, to effectively showcase and analyze student data. These omissions hinder the system's ability to provide a more

comprehensive and advanced user experience in managing and visualizing student information.

### **4.3 Future Improvement**

In the realm of developing a desktop software for student information management utilizing Java Swing and JDBC, numerous prospective avenues can be explored to augment the software's capabilities and address emerging requirements.

One avenue for future development entails the integration of cloud technologies. Through harnessing cloud services and storage, the software can offer heightened accessibility and data availability. This facilitates users in accessing student information from any location, engaging in real-time collaboration, and securely storing and backing up data. The integration with cloud-based authentication services further amplifies security measures and simplifies user management.

Lastly, the exploration of integrating artificial intelligence (AI) and machine learning (ML) presents opportunities for valuable automation and predictive capabilities in student information management. AI-powered functionalities such as intelligent data entry, automated grading systems, or personalized learning recommendations can optimize administrative tasks and enrich the learning experience.

Another area for potential enhancement is the implementation of data analytics and reporting features. By incorporating data visualization tools and analytics algorithms, the software can furnish valuable insights into student performance, trends, and patterns. This empowers educators and administrators to make informed, data-driven decisions and identify areas for improvement. Additionally, the incorporation of customizable reporting capabilities enables the generation of comprehensive reports tailored to specific needs.

In summary, the future directions for a Java Swing and JDBC-based student information management software encompass leveraging cloud technologies, refining data analytics and reporting capabilities, enhancing the user interface, fostering communication and collaboration, ensuring data privacy and security, and delving into the potential of AI and ML integration. Embracing these advancements allows the software to evolve and align with the evolving needs and expectations of educational institutions and stakeholders.

## WORK ASSIGNMENT AND MEMBERS EVALUATION

### Task Assignment

Name	The content of work
Ho Huu An	<ul style="list-style-type: none"> <li>• Develop user login functionality to cater to all user accounts.</li> <li>• Create a view and functionality to display the login history of a user.</li> <li>• Implement features for modifying user information as needed.</li> <li>• Design and implement a feature to change profile pictures for users.</li> <li>• Integrate functionality to add a new user, including fields such as name, age, phone number, and status (normal/locked).</li> <li>• Implement the functionality to delete a user from the system.</li> <li>• Develop an import feature to bring in a list of students from an external file.</li> <li>• Implement an import functionality to bring in a list of academic qualifications for a student from an external file.</li> <li>• Develop a view and functionality to showcase the list of system users.</li> <li>• Develop the export functionality to export the list of students to Excel/CSV formats.</li> <li>• Develop the export functionality to export the list of academic qualifications to Excel/CSV formats.</li> </ul>

Do Minh Quan	<ul style="list-style-type: none"> <li>• Create the view and functionality to display the list of students.</li> <li>• Develop the functionality to add a new student to the system.</li> <li>• Implement the functionality to delete a student record.</li> <li>• Enhance the system with the functionality to update student information as needed.</li> <li>• Develop sorting functionality for the list of students based on various criteria.</li> <li>• Implement a robust search functionality to locate students using multiple criteria.</li> <li>• Create a dedicated view and functionality to access the student details page.</li> <li>• Implement the functionality to view comprehensive student information on the details page.</li> <li>• Extend the system to manage a list of certificates for students, including viewing, adding, deleting, and updating certificate information.</li> </ul>
Nguyen Hoang Phuc	<ul style="list-style-type: none"> <li>• Implement user roles functionality (admin, manager, employee) in the Java Swing application using JDBC.</li> <li>• Create the view and functionality to manage user accounts, allowing the creation, editing, and deletion of accounts.</li> <li>• Implement functionality to restrict employee accounts from editing content other than their profile picture in the Java Swing application with JDBC.</li> <li>• Develop the profile picture update functionality specifically for employees using Java Swing and JDBC.</li> <li>• Assist in integrating the admin account seamlessly into the system without the need for manual creation, leveraging Java Swing and JDBC.</li> </ul>

	<ul style="list-style-type: none"> <li>• Provide support for the manager role to perform all functions related to students within the Java Swing application using JDBC.</li> <li>• Assist in creating the report (pdf) that introduces Java Swing and JDBC, explaining the development process of the student management application.</li> <li>• Assist in creating a demo video (with audio) that showcases the team's research findings and provides valuable information on the topic of Java Swing and JDBC in student management applications.</li> </ul>
--	---

*Table 4: Task Assignment*

### **Member evaluation**

Full name	Assigned tasks (100%)	Completion level
Ho Huu An	33.3%	On schedule as assigned
Nguyen Hoang Phuc	33.4%	On schedule as assigned
Do Minh Quan	33.3%	On schedule as assigned

*Table 5: Member Evaluation*

## **REFERENCE**

1. Friesen, J. (2006). Java Swing (2nd ed.). O'Reilly Media.
2. Lanchester, S. (2017). Java Swing (2nd ed.). Packt Publishing.
3. Vohra, D. (2012). JDBC: A Beginner's Guide. McGraw-Hill Education.