



NEW YORK INSTITUTE
OF TECHNOLOGY

COURSE NAME: COMPUTER PROGRAMMING II

COURSE CODE: CSCI185 - M03

SEMESTER: SPRING 2025

Final Exam Project

Global Student Network Generator

AUTHORS: JASMINE TUIACHIEVA, RITA CHEN, ARON LIN, HTOO NAING

INSTRUCTOR: AUSTIN STIETZEL

PRESENTATION DATE: 05-12-2025 (MM-DD-YYYY)

TABLE OF CONTENTS

Abstract	2
Introduction	2
System Architecture.....	3
Project Results	4
Main Page.....	4
Instructions Page	5
Network Page	6
Student Form Panel.....	7
University Dropdown Panel.....	8
Map Panel	9
Connection Strength Page	10
CompareConnections Class.....	10
FileManager Class.....	11
Video Demonstration and Github	12
Contributions	12
Rita Chen	12
Aron Lin.....	12
Htoo Naing	13
Jasmine Tuiachieva	13
Conclusion.....	13
Technical Specifications	14

ABSTRACT

International students often struggle to integrate into campus life due to **cultural barriers** and **social anxiety**. These challenges can lead to **isolation** and negatively affect their mental health. The *Global Student Network Generator* addresses this issue by simulating a virtual space where international undergraduates can connect with like-minded peers from around the world.

This Java-based application allows users to create student **profiles** including **interests**, **majors**, and **university** affiliations. Using a similarity-based algorithm, it calculates and displays **connection strength** between users. The graphical user interface (GUI) includes various interactive **panels** and a **world map**, visually representing each student's profile and potential connections.

Our project combines *GUI design, file I/O, object-oriented programming (OOP), and real-time user interaction* to create a socially meaningful software solution.

INTRODUCTION

To provide international students with a platform for forming global social connections, we built a fully interactive Java Swing-based GUI application. After brainstorming several project ideas we decided to focus on a **more purposeful, utility-driven tool**: the *Global Student Network Generator*.

We chose a **GUI-heavy approach** to make the program **intuitive** and **accessible** to users of varying technical skill levels. Rather than relying on text-based input or command-line interaction, we opted for a **clean, modular interface** where students can **visually select** universities, view **world maps**, and **interact** with other profiles via **clickable pins**. The interface design prioritizes *clarity, responsiveness, and logical layout*.

The final interface includes the following key components:

1. **Main Page** – A central hub for navigation
2. **Instructions Page** – A guide on how to use each button and panel
3. **Network Page** – The main operational window containing sub-panels:
 - **Student Form Panel**
 - **University Dropdown Panel**
 - **Map Panel**

4. **Connection Strength Page** – It offers instant feedback on how compatible two users are.

Our design decisions were grounded in object-oriented programming (OOP) best practices. We modularized the interface by separating major functions into different GUI classes and used *inheritance*, *encapsulation*, and *polymorphism* across domain classes like Student, Person, and University. The application also implements file I/O to save and load user profiles and uses visual feedback (like progress bars) to improve user interactivity.

SYSTEM ARCHITECTURE

The architecture of this application can be best understood through two diagrams: a **UML Class Diagram** and a **Flow Diagram**.

The **UML Class Diagram** shows the relationship among the application's **core data structures** and **GUI components**. It illustrates how user profile data flows through the system, how the GUI panels interact with those data models, and how utility classes like FileManager and CompareConnections support the application's functionality.

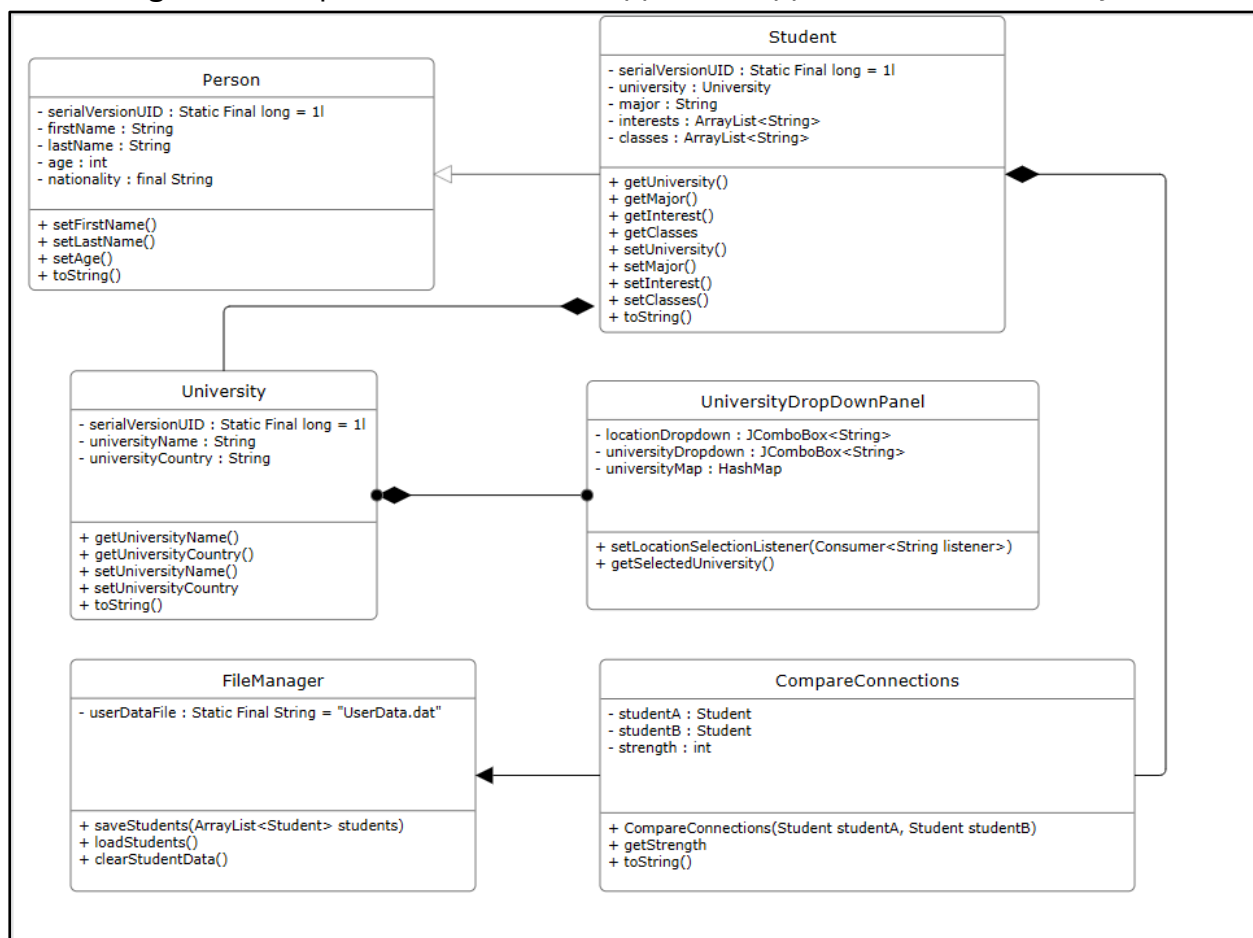


DIAGRAM 1. UML Class Diagram - Project Architecture.

The **Flow Diagram** explains the high-level behaviour of the application from a **user's perspective**. It shows how the system **initializes**, **responds to inputs**, **updates the GUI**, and **saves or retrieves data**.

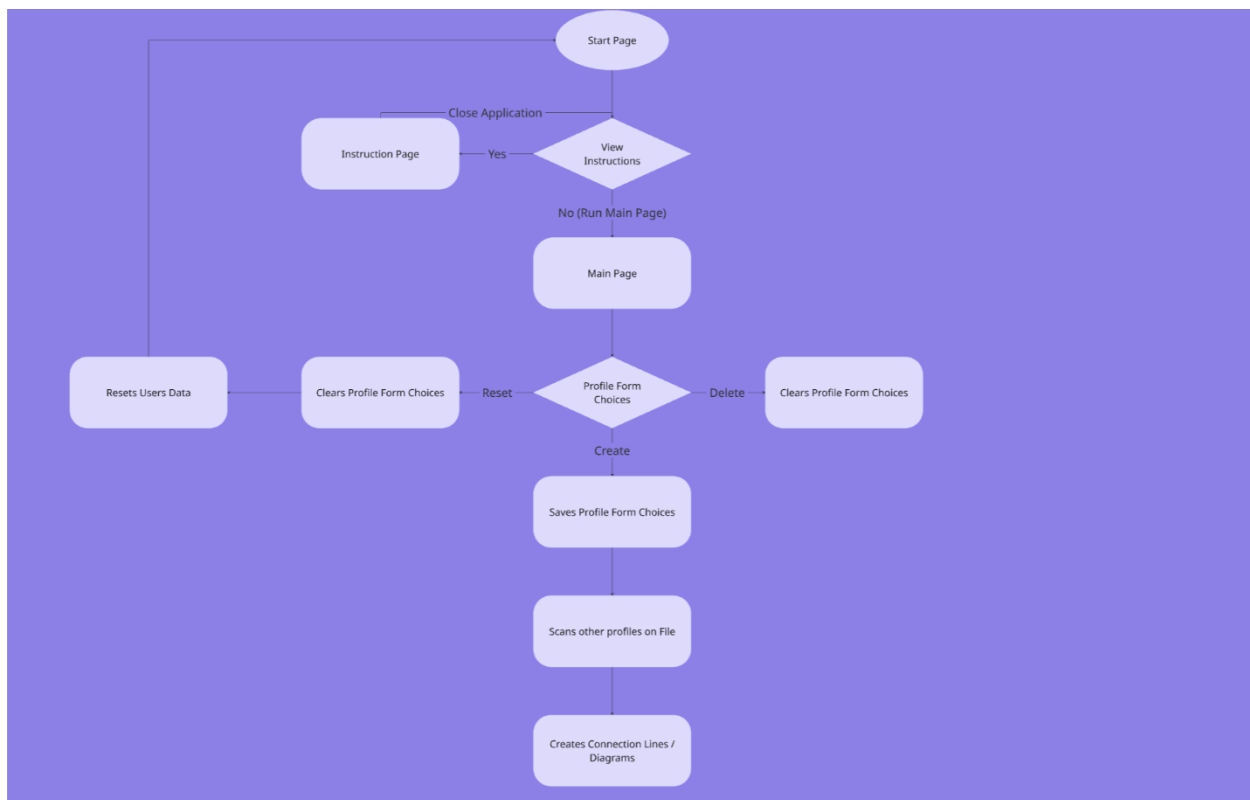


DIAGRAM 2. Flow Diagram - GUI and Data Interaction.

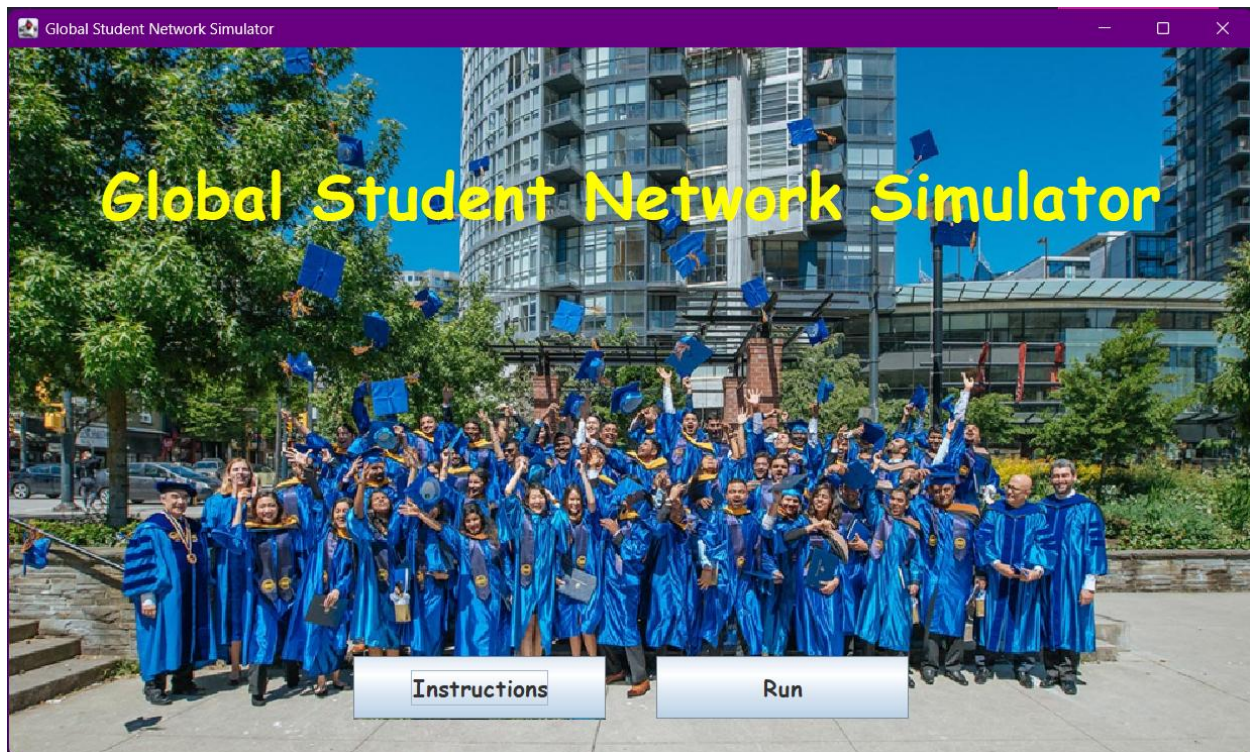
PROJECT RESULTS

The *Global Student Network Generator* consists of modular architecture, with each class and GUI panel fulfilling a distinct responsibility. This separation of concerns ensures a maintainable, readable, and scalable system. Below, we outline each key component and its connection to the project's goals and course concepts.

MAIN PAGE

The **Main Page** is the starting point of the application. It displays two primary navigation buttons:

- **Instructions**
- **Run**



SCREENSHOT 1. *Main Page – first window to open when you run the program.*

It utilizes Java's `ActionListener` interface to listen for button events and trigger transitions between screens. This page demonstrates event-driven programming and GUI control flow.

INSTRUCTIONS PAGE

The **Instructions Page** serves as a user manual. When selected from the main page, it presents detailed instructions on how to use each part of the interface. Its implementation includes:

- **JTextArea** for multiline text
- **JScrollPane** for vertical scrolling

The design ensures that first-time users can quickly understand how to interact with the program's features without external documentation.

Instructions

Hill

This application helps you discover what you have in common with other students who have submitted their information.

Connections Strength Table

Feature	Points if Same
First Name	5
Last Name	5
Age	10
Nationality	10
University Name	15
University Location	15

Previous Run

Instructions

Nationality	10
University Name	15
University Location	15
Major	10
Interests	$1 \times 10 = 10$ (max 10)
Classes	$4 \times 5 = 20$ (max 5)

After you fill out the form and press "Create Student", a pin with your name will appear on the map at your university location.

If at least two students are created, select their pins and click "Show Connections".

A new window will display your similarities (classes, interests, etc).

To delete all students and reset, press the "Reset" button below the map.

Previous Run

SCREENSHOTS 2&3. Instructions Page.

NETWORK PAGE

The **Network Page** is the central window for user interaction. It combines three major components:

- Student Form Panel (left)
- University Dropdown Panel (left)
- Map Panel (center)

Current Network

Student Profile Form

First Name:

Last Name:

Age:

Nationality:

University Country of Location:

University's Name:

Major:

Interests:

Classes:

Clear Form

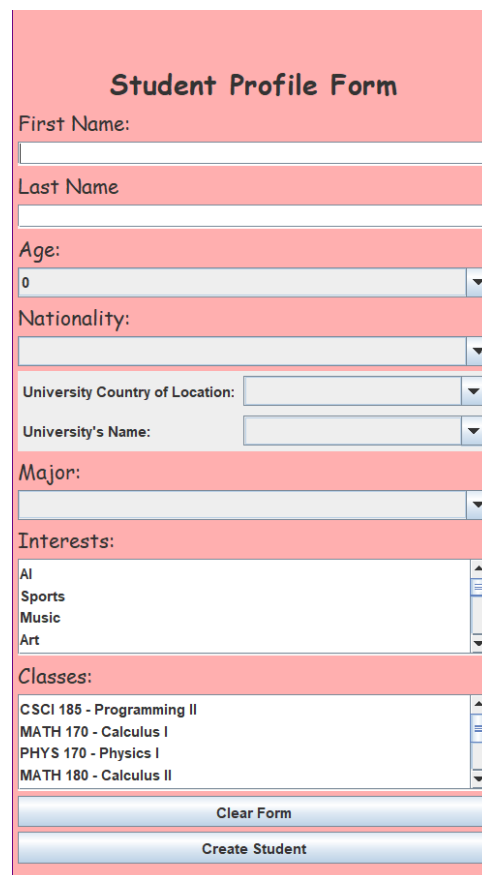
Create Student

SCREENSHOT 4. Network Page

This page is constructed using a combination of `JPanel` containers and layout managers. It demonstrates modular GUI design and inter-panel communication through shared state and event listeners.

STUDENT FORM PANEL

The **Student Form Panel** is a critical GUI component that allows users to **input their profile data**. This form serves as the foundation for creating a `Student` object, which is later visualized on the map and used in connection comparisons.



The screenshot displays a 'Student Profile Form' with a pink header. The form contains several input fields and dropdown menus: 'First Name:' and 'Last Name:' are text boxes; 'Age:' is a dropdown menu with '0' selected; 'Nationality:' is a dropdown menu; 'University Country of Location:' and 'University's Name:' are dropdown menus; 'Major:' is a dropdown menu; 'Interests:' is a list box containing 'AI', 'Sports', 'Music', and 'Art'; 'Classes:' is a list box containing 'CSCI 185 - Programming II', 'MATH 170 - Calculus I', 'PHYS 170 - Physics I', and 'MATH 180 - Calculus II'. At the bottom, there are two buttons: 'Clear Form' and 'Create Student'.

SCREENSHOT 5. *Student Form Panel.*

Key Features:

Input Fields:

- `TextField` for first and last names
- `ComboBox` for age, nationality, and major
- `MultiSelectionList` for interests and classes (custom panel)
- Embedded `UniversityDropdownPanel` for selecting university and location

Buttons:

- **Create** Button triggers student object creation and updates map
- **Clear** Button resets the form

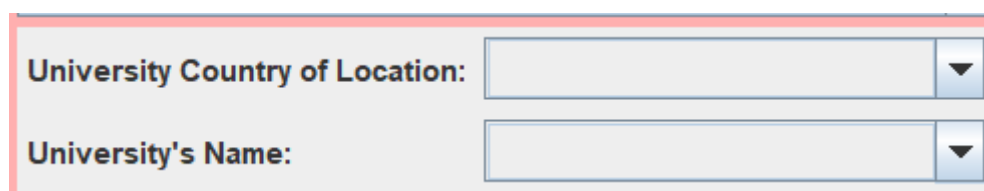
Design Concepts Applied:

- Uses GridBagLayout for precise and flexible form structuring
- Demonstrates modular GUI design by embedding other panels (like UniversityDropdownPanel)
- Applies encapsulation by keeping most fields private and linking via well-defined actions
- Triggers the FileManager to save student data once a profile is created

This panel plays a central role in connecting the GUI with the backend logic. It gathers user input, packages it into structured data (Student), and hands it off to other systems like MapPanel and CompareConnections.

UNIVERSITY DROPDOWN PANEL

This panel allows users to select a sample university by country. When a university is chosen, its geographic coordinates are assigned to the student's profile, and a pin is rendered on the map.

A screenshot of a Java Swing window titled "University Dropdown Panel". The window has a light gray background and a red border. It contains two rows of labels and text fields. The first row has the label "University Country of Location:" followed by a text field with a dropdown arrow on the right. The second row has the label "University's Name:" followed by a text field with a dropdown arrow on the right.

SCREENSHOT 5. *University dropdown panel.*

Technologies used:

- `JList<String>` for listing universities
- `ListSelectionListener` to detect changes
- Dynamic linking to `University` objects, which store name and location data

This panel is essential for generating the visual world map of student locations.

MAP PANEL

The **Map Panel** renders a world map and uses pins to represent student profiles. Each **pin** is **clickable** and displays relevant data about the selected student.

Implemented concepts:

- `paintComponent(Graphics g)` for custom drawing
- `HashMap<String, Point>` for country-to-coordinate mapping
- `MouseListener` and `MouseEvent` for click detection
- Visual indicators of selected pins



SCREENSHOT 6. Map panel.

When a pin is clicked, the map dynamically updates the Connection Strength Page using data from `CompareConnections`.

CONNECTION STRENGTH PAGE

The **Connection Strength Page** provides a graphical visualization of the similarity between two students. It changes dynamically based on a numeric score calculated by `CompareConnections`.

Key features:

- Uses `JProgressBar` to display similarity score as a percentage
- Encapsulation ensures only designated methods can modify the value
- Color changes: **Red** (low match), **Yellow** (moderate match), **Green** (strong match)

This component demonstrates how backend logic can be tightly integrated with GUI output for responsive user feedback.

CompareConnections Class

This utility class determines the **connection strength between two student profiles**. It compares values such as:

- First Name
- Last Name
- Age
- Nationality
- University Name
- University Location
- Major
- Interests
- Classes

Each **matching** field **increases** the **connection score**. The final value is returned as an **integer** and passed to the **progress bar**.

This class applies:

- OOP (class-based comparison)

- Abstraction (scores calculated behind the scenes)
- Clean method encapsulation



SCREENSHOT 7. *Sample Output from CompareConnections*

FileManager Class

The **FileManager** is responsible for **saving and loading student profiles**. It enables **long-term data storage** and project **persistence** using:

- `ObjectOutputStream` and `FileOutputStream` for saving profiles
- `ObjectInputStream` for loading data
- `try-catch` blocks for error handling and user feedback

Each student profile is **serialized** into a **.dat** file and can be reloaded even after the application closes.

VIDEO DEMONSTRATION AND GITHUB

A full video demonstration of the application, including the interface, features, and sample use cases is available on YouTube:

<https://youtu.be/oZV9fMl1vlw?si=X1T8N4vMggpOIXc5>

The complete source code and project files are available on GitHub:

https://github.com/mikaisloyal/Global-Student-Network_Smulator-Jasmines-Branch

CONTRIBUTIONS

This project was developed collaboratively by four team members, each of whom contributed distinct portions of the codebase, documentation, and design elements. Tasks were assigned based on technical strengths and project planning discussions.

RITA CHEN

- Designed and implemented the `MapPanel` class responsible for drawing and interacting with student pins on the world map.
- Developed the `UniversityDropdownPanel` to allow users to select a university by country, which affects pin placement.
- Constructed the internal structure for country-to-coordinate mapping using `HashMap` and Java `AWT`.
- Enabled mouse interaction with student pins, allowing users to view profile details and connection scores.
- Collaborated in mapping student locations to pins for global visualization.

ARON LIN

- Managed the GitHub repository, handled version control, and resolved all merge conflicts across branches.
- Created the `MainPage` and `InstructionsWindow` components, including layout, navigation, and user guidance.
- Designed and produced both the UML Diagram and Flow Diagram using external tools.

- Created a polished PowerPoint slide deck and final video walkthrough.
- Assisted with presentation graphics and overall visual cohesion.

HTOO NAING

- Wrote the full first draft of the formal project report and coordinated updates with the team.
- Drafted and revised all slide content for the presentation.
- Designed the `ConnectionStrengthBar`, linking backend connection scores to visual feedback in the GUI.
- Was originally tasked with the `ConnectionEngine`, a prototype that evolved into the `CompareConnections` class.
- Applied color-coded logic to the strength bar based on numerical thresholds.

JASMINE TUIACHIEVA

- Proposed and outlined three project ideas and led the planning process by drafting the Final Project Plan.
- Designed visual mockups of the GUI layout to guide development.
- Implemented the `StudentFormPanel` which gathers input and integrates the university panel, interests, and classes.
- Created the foundational `Person` and `Student` classes, and modified `University` to support serialization.
- Wrote the `CompareConnections` class to compute profile similarity scores.
- Integrated File I/O into the `FileManager` for saving/loading profiles.

CONCLUSION

The *Global Student Network Generator* successfully met its core objectives: allowing international students to create profiles, select universities, and visualize potential connections with others through an interactive map interface. Despite **time constraints** and **initial uncertainty** about the project direction, our team delivered a **minimum viable product (MVP)** that includes a modular **GUI**, **file persistence**, and a **meaningful algorithm** for calculating user compatibility.

Throughout development, we faced **challenges with project scoping, timeline compression, and integration** across different team members' components. In particular, working with Java Swing layouts and **merging GUI logic with backend** processes required careful coordination and multiple iterations.

Still, the experience gave us valuable technical and collaborative growth. We learned to:

- Use object-oriented programming principles effectively in a team setting
- Employ file I/O and error handling for robust user experiences
- Visualize data dynamically using Java's graphics tools
- Manage contributions using GitHub and resolve real-world merge conflicts

If given more time, we would expand the project by implementing:

- Multi-profile comparison logic
- Randomized student events or profile evolution
- Chat or message simulation features
- A database integration for persistent cloud-based storage

This project not only helped us reinforce everything we've learned in Programming II but also showed us how to scope, design, and execute a real-world software solution with meaningful purpose.

TECHNICAL SPECIFICATIONS

Below are the libraries, APIs, and documentation sources we consulted while building this application. All references are formatted in IEEE citation style.

[1] "HashMap in Java with Examples," *GeeksforGeeks*, Apr. 28, 2017.

<https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/>

[2] "Java: create a list of HashMaps," *Stack Overflow*.

<https://stackoverflow.com/questions/4100486/java-create-a-list-of-hashmaps>

[3] “Stack Overflow,” *Stack Overflow*, Nov. 13, 2011.

<https://stackoverflow.com/questions/8114406/adding-image-to-jpanel-through-imageio-read>

[4] Overriding paintComponent, “Stack Overflow,” *Stack Overflow*, Mar. 24, 2013.

<https://stackoverflow.com/questions/15602235/overriding-paintcomponent>

[5] “Graphics (Java Platform SE 8),” *Oracle.com*, Apr. 05, 2025.

<https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html#drawImage>

[6] “Graphics2D (Java Platform SE 8),” *docs.oracle.com*.

<https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics2D.html>

[6] “paint() and repaint() in Java,” *Stack Overflow*, May 26, 2012.

<https://stackoverflow.com/questions/10768619/paint-and-repaint-in-java>

[7] “How can I use a custom font in Java?” *Stack Overflow*.

<https://stackoverflow.com/questions/5652344/how-can-i-use-a-custom-font-in-java>

[8] “Stack Overflow,” *Stack Overflow*, Oct. 04, 2019.

<https://stackoverflow.com/questions/58240025/how-to-draw-any-text-using-drawstring-in-java>

[9] “Java 8 | Consumer Interface in Java with Examples,” *GeeksforGeeks*, Sep. 24, 2018. <https://www.geeksforgeeks.org/java-8-consumer-interface-in-java-with-examples/>

[10] “Consumer (Java Platform SE 8),” *docs.oracle.com*.

<https://docs.oracle.com/javase/8/docs/api/java/util/function/Consumer.html>

[11] “How to Write a Mouse Listener (The Java™ Tutorials > Creating a GUI With JFC/Swing > Writing Event Listeners),” *docs.oracle.com*.

<https://docs.oracle.com/javase/tutorial/uiswing/events/mouselistener.html>

[12] “MouseListener and MouseMotionListener in Java,” *GeeksforGeeks*, May 02, 2018.

<https://www.geeksforgeeks.org/mouselistener-mousemotionlistener-java/>

[13] “Java MouseListener,” *Stack Overflow*.

<https://stackoverflow.com/questions/2668718/java-mouselistener>

[14] Oracle, “List (Java Platform SE 8),” *docs.oracle.com*.

<https://docs.oracle.com/javase/8/docs/api/java/util/List.html>

[15] “Rectangle (Java Platform SE 8),” *Oracle.com*, Dec. 04, 2024.

<https://docs.oracle.com/javase/8/docs/api/java/awt/Rectangle.html>

[16] “Detecting a mouse click inside a box painted in Frame? [Solved] (Swing / AWT / SWT forum at Coderanch),” *Coderanch.com*, 2025.

<https://coderanch.com/t/645861/java/Detecting-mouse-click-box-painted>

[17] “How to Use GridBagLayout (The Java™ Tutorials > Creating a GUI With JFC/Swing > Laying Out Components Within a Container),” *Oracle.com*, 2017.

<https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html>

[18] “Color (Java Platform SE 7),” *docs.oracle.com*.

<https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html>

[19] “JProgressBar (Java Platform SE 8),” *Oracle.com*, Dec. 04, 2024.

<https://docs.oracle.com/javase/8/docs/api/javax/swing/JProgressBar.html>

- [20] “How to set an image as a background for Frame in Swing GUI of java?,” *Stack Overflow*, Sep. 23, 2009. <https://stackoverflow.com/questions/1466240/how-to-set-an-image-as-a-background-for-frame-in-swing-gui-of-java>
- [21] “Serialization and Deserialization in Java with Example” *GeeksforGeeks*, Feb. 22, 2019. <https://www.geeksforgeeks.org/serialization-in-java/>
- [22] “How to Align Text in HTML?,” *GeeksforGeeks*, Jun. 20, 2021. <https://www.geeksforgeeks.org/how-to-align-text-in-html/>
- [23] “How to center the text in a JLabel?,” *Stack Overflow*, Jul. 25, 2011. <https://stackoverflow.com/questions/6810581/how-to-center-the-text-in-a-jlabel>
- [24] “Git Commit,” *W3schools.com*, 2021. https://www.w3schools.com/git/git_commit.asp
- [25] “Convert List to Array in Java,” *GeeksforGeeks*, Mar. 27, 2018. <https://www.geeksforgeeks.org/convert-list-to-array-in-java/> (accessed May 12, 2025).
- [26] “ArrayList in a JList,” *Oracle Forums*, Aug. 04, 2008. <https://forums.oracle.com/ords/apexds/post/arraylist-in-a-jlist-8029> (accessed May 12, 2025).
- [27] “Lesson: Writing Event Listeners (The Java™ Tutorials > Creating a GUI With Swing),” *docs.oracle.com*. <https://docs.oracle.com/javase/tutorial/uiswing/events/index.html>

[28] “Scan other application for input in X window,” *Stack Overflow*, Dec. 22, 2014.

<https://stackoverflow.com/questions/27595139/swing-how-do-i-detect-which-component-i-clicked> (accessed May 12, 2025).

[29] “A Visual Guide to Layout Managers (The Java™ Tutorials > Creating a GUI With JFC/Swing > Laying Out Components Within a Container),” *docs.oracle.com*.

<https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>

[30] “Introduction to Java Serialization,” *Baeldung*, May 15, 2017.

<https://www.baeldung.com/java-serialization>

[31] “Convert List to Array in Java,” *GeeksforGeeks*, Mar. 27, 2018.

<https://www.geeksforgeeks.org/convert-list-to-array-in-java/>

[32] “Stack Overflow,” *Stack Overflow*, Oct. 26, 2012.

<https://stackoverflow.com/questions/13093410/how-to-detect-mouse-clicks-on-a-specific-object-in-java> (accessed May 12, 2025).

[33] “Stroking and Filling Graphics Primitives (The Java™ Tutorials > 2D Graphics > Working with Geometry),” *docs.oracle.com*.

<https://docs.oracle.com/javase/tutorial/2d/geometry/strokeandfill.html>

[34] Ouko Omondi, “How to resolve “The requested URL returned error:403 in GitHub,”

Medium, Jan. 15, 2023. <https://medium.com/@akwando15/how-to-resolve-the-requested-url-returned-error-403-in-github-cd66bac6d9fa>

- [35] “Gitignore Explained: What is Gitignore and How to Add it to Your Repo,” *freeCodeCamp.org*, Jan. 03, 2020. <https://www.freecodecamp.org/news/gitignore-what-is-it-and-how-to-add-to-repo/>
- [36] Coding Examples, “Multi Select JList | Java Swing Tutorial #23,” *YouTube*, Jan. 28, 2021. <https://www.youtube.com/watch?v=WiX3n2BMGlc> (accessed May 12, 2025).
- [37] “Font (Java Platform SE 8),” *docs.oracle.com*.
<https://docs.oracle.com/javase/8/docs/api/java/awt/Font.html>
- [38] “BorderLayout (Java Platform SE 7),” *docs.oracle.com*.
<https://docs.oracle.com/javase/7/docs/api/java/awt/BorderLayout.html>
- [39] “JComboBox (Java Platform SE 8),” *docs.oracle.com*.
<https://docs.oracle.com/javase/8/docs/api/javafx/swing/JComboBox.html>
- [40] Lazic B., “Java swing GUI tutorial #20: GridBagLayout,” *YouTube*, Apr. 03, 2013.
<https://www.youtube.com/watch?v=sq46PYdW4c8> (accessed May 12, 2025).
- [41] Mam Panhavuth Programming, “How to customize JPanel with GridBagLayout and load Images from Files in Java Using Netbeans IDE,” *YouTube*, May 02, 2020.
<https://www.youtube.com/watch?v=mftMp33J4lg> (accessed May 12, 2025).
- [42] “JFrame | Java Swing Tutorial for Beginners,” *Youtube*, Oct. 19th, 2020.
<https://www.youtube.com/watch?v=4BRUmU-ETRk>
- [43] “GridLayout | Java Swing Tutorial for Beginners,” *Youtube*, Dec. 29th, 2020.
<https://www.youtube.com/watch?v=impJtkTcQ94> (accessed May 03, 2024).

[44] Pritam Thing, “How To Transform One Frame To Another Frame in Java Netbeans,” *YouTube*, Aug. 19, 2018. <https://www.youtube.com/watch?v=PuCvkTcutJI> (accessed May 12, 2025).