22AIE111 – Object Oriented Programming in Java

A THESIS
Submitted by

Group – 9
Group Members

1. D. NIKHIL SAI - CB.SC.U4AIE23019

2. DEV BALA SARAGESH - CB.SC.U4AIE23022

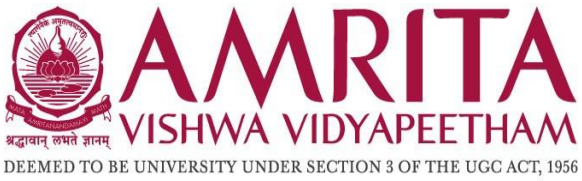
3. HARI HEMAN V K - CB.SC.U4AIE23029

4. RAGHAV N - CB.SC.U4AIE23059

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE ENGINEERING – ARTIFICIAL INTELLIGENCE



Centre for Computational Engineering and Networking

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

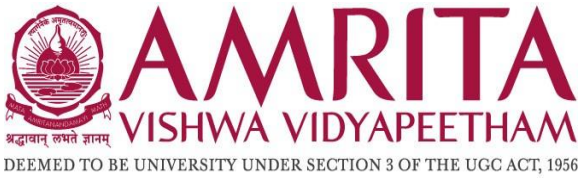
COIMBATORE – 641 112 (INDIA)

June 2024

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE – 641 112 (INDIA)



BONAFIDE CERTIFICATE

This is to certify that the thesis entitled "**Password Generator and Password Manager**" submitted by D. Nikhil Sai - CB.SC.U4AIE23019, Dev Bala Saragesh - CB.SC.U4AIE23022, Hari Heman V K - CB.SC.U4AIE23029, Raghav N - CB.SC.U4AIE23059) for the award of the Degree of Bachelor of Technology in the "CSE(AI)" is a bonafide record of the work carried out by his under our guidance and supervision at Amrita School of Artificial Intelligence, Coimbatore.

Mr. Vipin Das
Project Guide

Dr. K.P.Soman
Professor and Head CEN

Submitted for the university examination held on 05-06-2024

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE – 641 112 (INDIA)

DECLARATION

We Group – 9 (D. Nikhil Sai - CB.SC.U4AIE23019, Dev Bala Saragesh - CB.SC.U4AIE23022, Hari Heman V K - CB.SC.U4AIE23029, Raghav N - CB.SC.U4AIE23059) hereby declare that this thesis entitled "**Password Generator and Password Manager**"**,** is the record of the original work done by us under the guidance of Mr. Vipin Das, Centre for Computational Engineering and Networking, Amrita School of Artificial Intelligence, Coimbatore. To the best of my knowledge this work has not formed the basis for the award of any degree/diploma/ associate ship/fellowship/or a similar award to any candidate in any University.

Place: Coimbatore

Date:05-06-2024

Signature of the Students

TABLE OF CONTENT

List of Figures 2

Acknowledgement 3

Abstract 4

1. Introduction to Password Manager

1.1 Overview of Password Manager 5

2. Implementation of Password Manager

2.1 MODULE-1 (Password Generation) …… 6

2.2 MODULE-2 (Password Manager – Part:1) ……. 6

2.3 MODULE-3 (Password Manager – Part:2) ……... 8

2.4 MODULE-4 (GUI) 9

3. Result 12

4. Conclusion and Future Work

4.1 Conclusion 15

4.2 Future Works 16

Bibliography 17

List of Figures

Fig 2.4.1 UML Diagram	**11**
Fig 3.1 Menu	**12**
Fig 3.2 Generate password	**12**
Fig 3.3 Enter password	**13**
Fig 3.4 Find password	**13**
Fig 3.5 Delete password	**15**
Fig 3.6 If the user already exist	**17**

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our teacher (Mr.VIPIN DAS sir), who gave us the golden opportunity to do this wonderful project on the topic "**Password Generator and Password Manager**" which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given. We would also like to thank our group members, as without their cooperation, we would not have been able to complete the project within the prescribed time.

ABSTRACT

In the modern world of technology, having strong and secure passwords is crucial. To meet this challenge, our project introduces a complete solution for creating and organizing passwords in Java. Our system utilizes reliable cryptographic methods to produce extremely secure passwords with different conditions with industry standards like different case sensitive, special characters and symbols to make it unbreakable through attacks like brute force, etc...It also offers an easy-to-use interface for effective password management, enabling users to securely store, access, and modify passwords. To protect sensitive data like master passwords and stored credentials from unauthorized individuals, the system employs advanced encryption techniques. Furthermore, it includes functions like evaluating password strength and automatically expiring passwords.

1. INTRODUCTION TO PASSWORD MANAGER

1.1 Overview of Password Manager

Password managers offer a secure solution by storing your login credentials for all your online accounts in one encrypted vault. This eliminates the need to remember countless passwords and frees you from the temptation to reuse weak ones. With strong, unique passwords generated by the manager, your online accounts become more resistant to hacking. Password managers often come with automatic login, saving you time and effort by filling in login information on websites and apps. By promoting strong, unique passwords and discouraging reuse, password managers significantly improve your overall online security.

2. IMPLEMENTING PASSWORD MANAGER

2.1 MODULE -1 : PASSWORD GENERATION

The Java code defines a class called PasswordGen that helps you create secure passwords. It accomplishes this by generating a random sequence of characters containing uppercase and lowercase letters, numbers, and special symbols. You control the strength of the password by specifying its desired length when calling the Password method.

- First, the code separates uppercase, lowercase, numbers, and special characters into individual strings. It then combines these strings into a single pool of possible characters for the password.
- Next, it creates a random number object to generate unpredictable selections.
- The code then defines an empty character array with a size matching your chosen password length.
- It enters a loop that iterates for the desired password length. Inside the loop, a random index is picked within the range of the entire character pool. The character at that index is then extracted and placed in the corresponding position of the password array.
- Finally, after the loop finishes, the code returns the completed password array, containing a random mix of uppercase and lowercase letters, numbers, and special characters.

2.2 MODULE -2 : PASSWORD MANAGER – PART : 1

Three methods within the Manager class that handle checking for existing entries, generating passwords, and displaying data:

1. **if_exist(Username, App_name)**

   This method checks if a specific username and app name combination already exists in the database.

   - It takes two arguments: Username and App_name.
   - It attempts to establish a connection to the database using the same approach as the previous methods.

- It constructs a PreparedStatement for a SELECT operation. The statement retrieves all columns (*) from the password table where the username and app_name match the provided arguments.
- It executes the prepared statement and retrieves the result set (ResultSet).
- It uses the rs.next() method to check if there's at least one row in the result set. If rs.next() returns true, it means a matching entry exists.
- If a matching entry exists, it returns true. Otherwise, it returns false.
- If a SQLException occurs, it prints the stack trace but still returns false.

2. **gen_password(username, appname)**

This method generates a new password and adds a new entry to the database.

- It takes two arguments: username and appname.
- It creates a new PasswordGen object (assuming it provides a method to generate random passwords).
- It generates a new random password of length 16 (modify this as needed) using PasswordGen.Password(16) and stores it in the password variable (consider not storing the password in plain text).
- It establishes a database connection.
- It constructs a PreparedStatement for an INSERT operation. The statement inserts the provided username, generated password, and app name into the password table.
- It executes the prepared statement and checks for success.
- It doesn't return a value, but any SQLException is caught and the stack trace is printed.

3. **show_data(appname)**

This method retrieves and displays username and password details for a specific app name.

- It takes one argument: appname.
- It constructs a string containing the SQL query to select username and password from the password table where the app_name matches the provided argument.
- It executes the query using a regular Statement (not recommended) and retrieves the result set.
- It iterates through the result set using a while loop. Inside the loop:
- It retrieves the username and password values from the current row of the result set.
- It constructs a formatted string containing "User Name:" and "Password:" labels with the retrieved values.

- It adds the formatted string to the data list.
- If a SQLException occurs, it prints the stack trace.
- Finally, it returns the data list containing formatted strings with username and password information for the specified app name.

2.3 MODULE -3 : PASSWORD MANAGER – PART : 2

Three methods within the Manager class that handle adding, modifying, and deleting password entries in a database:

1. **enter_password(username, password, appname)**

   This method adds a new password entry to the database.
   - It takes three arguments: username (the user's username for the application), password (the password to store), and appname (the name of the application).
   - It attempts to establish a connection to the database using the provided URL, username (root), and password (HEMAN) (consider securing these credentials).
   - It constructs a SQL statement using a PreparedStatement to prevent SQL injection vulnerabilities. The statement inserts the provided username, password, and app name into the password table.
   - It executes the prepared statement and checks for success.
   - If successful, it returns true to indicate a successful addition.
   - If a SQLException occurs (e.g., connection failure or database error), it prints the stack trace and returns false to signal an error.

2. **change_data(username, password, appname)**

   This method updates an existing password entry in the database.
   - It takes three arguments: username, password (the new password), and appname (the application name for the entry to update).
   - It first checks if the provided password is empty.
   - If empty, it creates a new PasswordGen object and generates a random password of length 16 (modify this length as needed).
   - It establishes a database connection similar to enter_password.
   - It constructs a PreparedStatement for an UPDATE operation. The statement sets the password for the specified username and app name combination.
   - It executes the prepared statement and checks for success.
   - It returns true if successful or false if a SQLException occurs (printing the error details).

3. **delete_data(appname, username)**

   This method removes a password entry from the database.

   - It takes two arguments: appname (the application name) and username (the user for the entry to delete).
   - It establishes a database connection using the same approach as the previous methods.
   - It constructs a PreparedStatement for a DELETE operation. The statement removes the entry identified by the username and app name combination.
   - It executes the prepared statement and checks for success.
   - It returns true if successful or false if a SQLException occurs (printing the error details).

2.4 MODULE -4 : GUI

The App.java file contains the code for the graphical user interface (GUI) of our Password Manager application. This Java program utilizes the Swing library to create an intuitive and user-friendly interface, allowing users to securely manage their passwords.

1. **Class Structure:** The App class extends JFrame, making it a standalone window in Java's Swing framework. It also implements ActionListener, enabling it to respond to user actions such as button clicks. This design follows object-oriented principles, encapsulating both the GUI elements and their behavior within a single class.

2. **GUI Components:** The application window is composed of several key components:
   - **Text Fields:** usernameField, appNameField, and passwordField allow users to input their credentials.
   - **Buttons:** generateButton, enterButton, findButton, and deleteButton offer various password management functions.
   - **Text Area:** outputArea displays operation results, providing instant feedback.
   - **Panels:** headerPanel and others organize components for a clean layout.

   This modular design makes the interface both functional and easy to navigate.

3. **Constructor:** The App() constructor sets up the entire GUI. It uses BorderLayout to arrange components logically:
   - North: Header with logo and title
   - Centre: Input fields

- South: Action buttons
- East: Output area

The constructor also sets window properties like size and location, ensuring a consistent look across different systems.

4. **Aesthetic Design:** We've paid close attention to aesthetics:
   - **Color Scheme:** Dark gray (RGB: 51, 51, 51) for headers, light gray (RGB: 240, 240, 240) for input areas, and a striking blue (RGB: 0, 0, 255) for buttons against a red (RGB: 255, 0, 0) panel. This creates a professional, high-contrast look.
   - **Typography:** Bold, 24-point Arial font in white for the heading, enhancing readability.
   - **Branding:** Our University logo in the header personalizes the application.

   These design choices make our application visually appealing and align with our institution's branding.

5. **Event Handling:** The actionPerformed method is the heart of user interaction. It handles button clicks by:
   1. Identifying the clicked button
   2. Retrieving user input
   3. Calling appropriate Manager class methods
   4. Displaying results in the outputArea

   Key functionalities include:
   - **Generate Password:** Creates a new, secure password.
   - **Enter Password:** Stores a user-provided password.
   - **Find Password:** Retrieves passwords for a given app.
   - **Delete Password:** Removes a password entry.

   The method also handles edge cases, like asking for confirmation before overwriting an existing password. This thoughtful design prevents accidental data loss.

6. **Data Presentation:** When displaying data, like in the findButton action, we use Java's StringBuilder to efficiently concatenate strings. This is not just a performance optimization; it also allows us to format output neatly, with each entry on a new line. Clear presentation is crucial in a password manager, where misreading data could lead to login failures.

7. **Error Handling:** The application provides clear feedback for all operations. Phrases like "Password changed successfully" or "Failed to delete password" keep users informed. In a security-critical app, this transparency is vital—users need to know exactly what happened to their sensitive data.

8. **Main Method:** The main method uses SwingUtilities.invokeLater to create the GUI on Swing's Event Dispatch Thread. This is a best practice that prevents GUI freezes and ensures a responsive user experience.

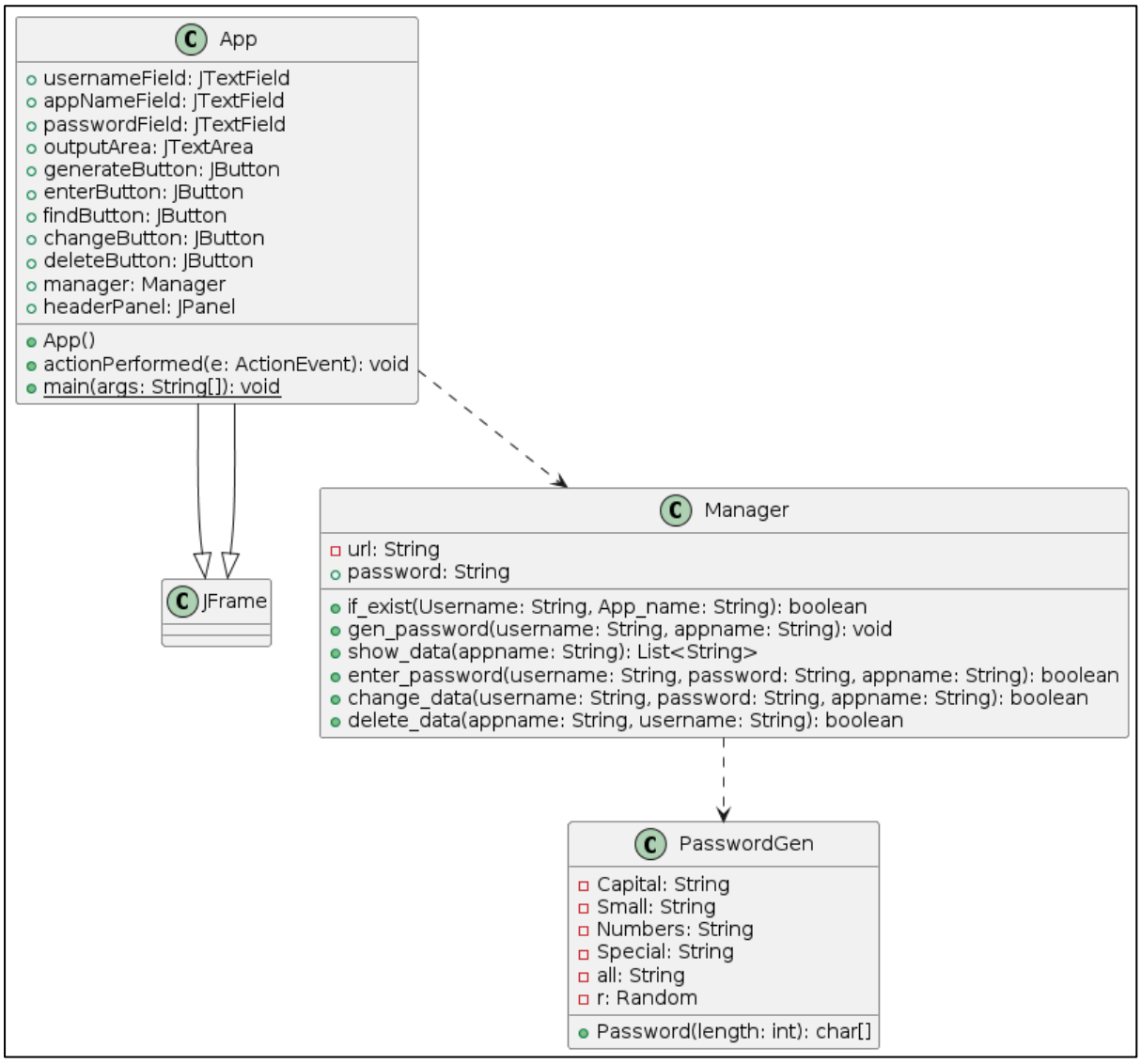


Fig 2.4.1 UML Diagram

3. RESULT

Fig 3.1 Menu

Fig 3.2 Generate Password

Fig 3.3 Enter Password

Fig 3.4 Find Password

Fig 3.5 Delete Password

Fig 3.6 If user already exist

4. CONCLUSION AND FUTURE WORK

4.1 Conclusion

In conclusion, our Password Manager application successfully addresses the critical need for secure and user-friendly password management in today's digital landscape. By integrating an intuitive Java Swing GUI with a robust MySQL backend, we've created a tool that not only safeguards sensitive information but also guides users towards better security practices. The project's modular design, showcased in the **App.java** and **Manager.java** files, ensures maintainability and sets a strong foundation for future enhancements. Key features like SQL injection prevention and password generation underscore our commitment to security. This application isn't just a technical solution; it's a step towards a future where robust protection and ease of use coexist, empowering users to take control of their digital identities in an increasingly complex online world.

4.2 Future Works

1. Multi-Factor Authentication (MFA):

- Implement MFA for accessing the password manager itself, using methods like:
    - SMS or email-based one-time passwords (OTP)
    - Time-based OTP via apps like Google Authenticator
    - Biometric authentication (fingerprint, facial recognition) on supported devices
- This adds a critical layer of security, ensuring that even if the master password is compromised, unauthorized access is prevented.

2. Password Health Check:

- Integrate with APIs like HaveIBeenPwned to check if stored passwords have been involved in known data breaches.
- Analyze password strength and suggest improvements (e.g., warning about reused or weak passwords).
- Generate a "password health score" to encourage users to enhance their overall security posture.

3. Cross-Platform Synchronization:

- Develop companion apps for iOS, Android, and web browsers.
- Use secure cloud storage (e.g., AWS S3 with encryption) to sync passwords across devices.
- This ensures that users have access to their passwords wherever they are, enhancing convenience without compromising security.

4. Browser Extension Integration:

- Create extensions for popular browsers (Chrome, Firefox, Safari).
- Enable auto-fill functionality for stored passwords.
- Offer to save new passwords as users create accounts or change credentials.
- This seamless integration could significantly boost user adoption.

5. Password Sharing Features:

- Allow secure password sharing among trusted contacts.
- Implement temporary access grants for shared accounts.
- This is particularly useful for families or teams managing joint accounts.

6. BIBLIOGRAPHY

[1] Password Manager - https://dl.acm.org/doi/abs/10.1145/2420950.2420964

[2] Password Generator-
chromeextension://efaidnbmnnnibpcajpcglclefindmkaj/https://eprints.uad.ac.id/43864/1/1-The%20Usage%20of%20Password%20Generators%20to%20Enhance%20Data%20Security%20in%20Most%20Used%20Applications.pdf

[3] How to connect JDBC - https://www.youtube.com/watch?v=92q_iJHjZ1g

[4] How to use GUI in Java - https://youtu.be/5G2XM1nlX5Q?si=_ptVluw2pkyzd11f