VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI 590018

Project Report on

"Habit Hero- Habit Tracker"

By

HRISHIKESH PATIL(1BM25CS457) JAYANTH M M(1BM25CS459)
JEEVAN R(1BM25CS461) JNANESH J(1BM25CS462)
SHASHANK J(1BM25CS513)

Under the Guidance of

MONISHA H M
Assistant Professor, Department of CSE
BMS College of Engineering

Work carried out at

Department of Computer Science and Engineering
BMS College of Engineering
(Autonomous college under VTU)

P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019
2025-2026

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOPS with JAVA project titled " Habit Hero- Habit Tracker" has been carried out by HRISHIKESH PATIL(1BM25CS457), JAYANTH M M(1BM25CS459), JEEVAN R(1BM25CS461), JNANESH J(1BM25CS462), SHASHANK J(1BM25CS513)during the academic year 2025-2026.

Signature of the guide
MONISHA H M
Assistant Professor,
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, HRISHIKESH PATIL(1BM25CS457),JAYANTH M M(1BM25CS459),JEEVAN R(1BM25CS461), JNANESH J(1BM25CS462), SHASHANK J(1BM25CS513), students of 3rd Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this project work entitled "HABIT HERO-HABIT TRACKER" has been carried out by us under the guidance of MONISHA H M , ASSISTANT Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester Sep-Dec 2025. We also declare that to the best of our knowledge and belief, the project reported here is not from part of any other report by any other students.

Signature of the Candidates

HRISHIKESH PATIL(1BM25CS457)

JAYANTH M M(1BM25CS459)

JEEVAN R(1BM25CS461)

JNANESH J(1BM25CS462)

SHASHANK J(1BM25CS513)

TABLE OF CONTENTS

Problem Statement

Project Title

Habit Hero: Personal Habit Tracking Application

Problem Definition

To develop a Java-based desktop application that helps users build and maintain positive habits through daily tracking, streak monitoring, and progress visualization. The system should:

- Provide secure user registration and authentication
- Allow users to create, update, and delete habits
- Track daily completion status of habits
- Calculate and maintain streak counts for each habit
- Display motivational statistics and quotes
- Store user data persistently between sessions
- Provide an intuitive graphical user interface
- Handle multiple user accounts securely

Objectives

To implement a complete habit tracking system using Java Swing.

To demonstrate object-oriented programming principles effectively.

To create a user-friendly graphical interface for habit management.

To implement secure authentication with password hashing.

To provide persistent data storage using file serialization.

Introduction

What is Habit Tracking?

Habit tracking is a method of monitoring daily behaviors to build consistency and achieve personal goals. Research shows that tracking habits increases the likelihood of success by 80% by providing visual feedback and accountability.

Digital Habit Tracking Systems

Modern habit tracking applications combine behavioral psychology with technology to:

- Provide reminders and notifications
- Track streaks and consistency
- Offer motivational feedback
- Analyze patterns and progress
- Support goal achievement through visualization

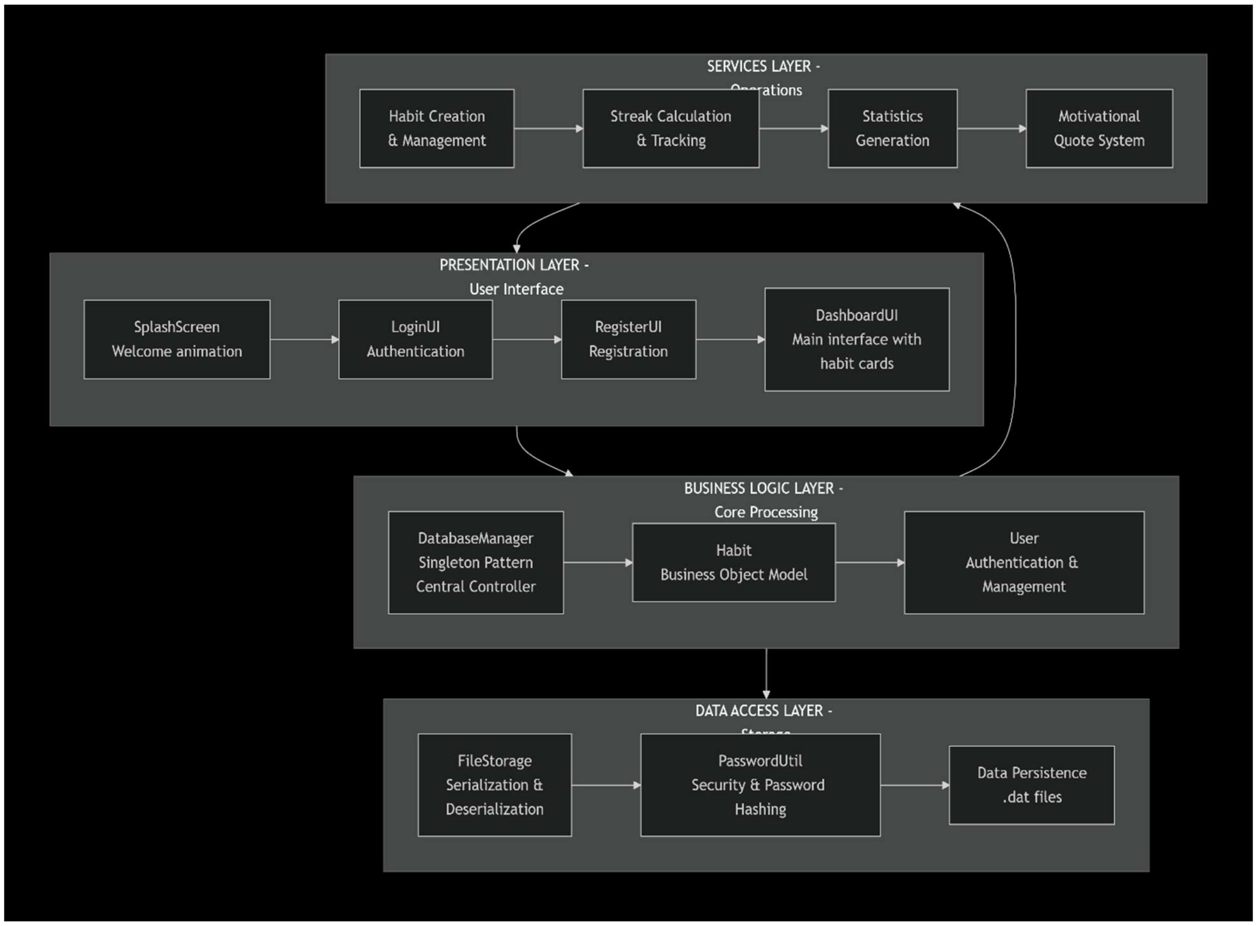
Project Significance

This project serves as:

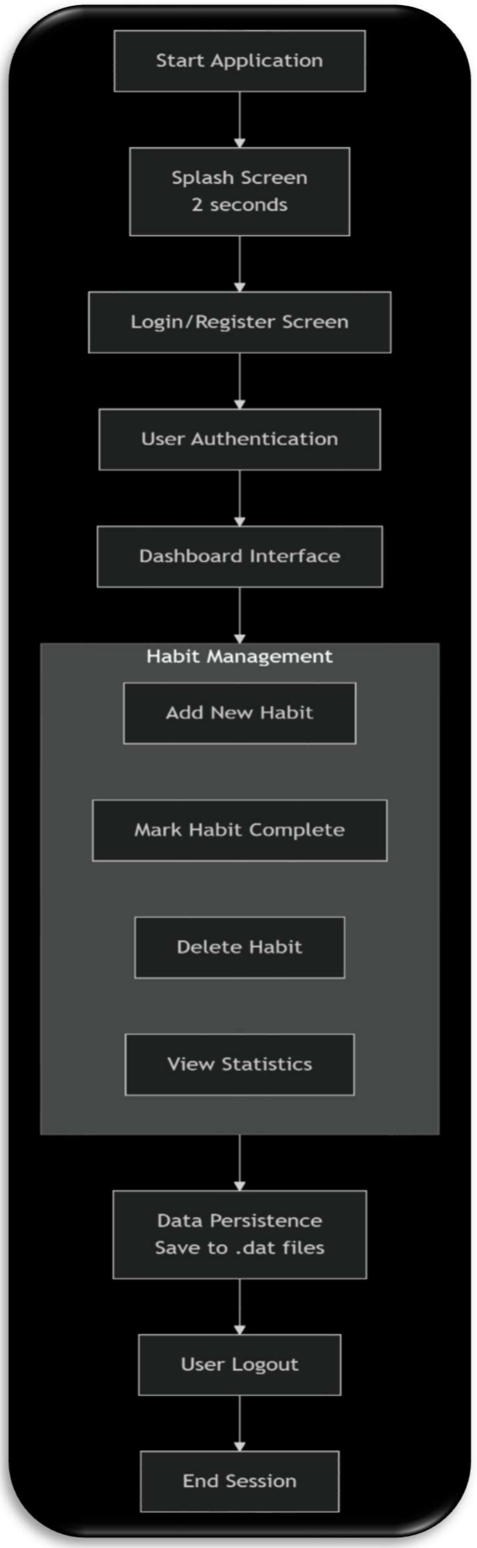
- A practical tool for personal development and habit formation
- A demonstration of Java Swing GUI development capabilities
- An example of file-based data persistence implementation
- A showcase of object-oriented design principles
- An educational resource for learning Java application development

Overview

System Architecture


SERVICES LAYER - Operations
Habit Creation & Management
Streak Calculation & Tracking
Statistics Generation
Motivational Quote System
PRESENTATION LAYER - User Interface
SplashScreen Welcome animation
LoginUI Authentication
RegisterUI Registration
DashboardUI Main interface with habit cards
BUSINESS LOGIC LAYER - Core Processing
DatabaseManager Singleton Pattern Central Controller
Habit Business Object Model
User Authentication & Management
DATA ACCESS LAYER - Storage
FileStorage Serialization & Deserialization
PasswordUtil Security & Password Hashing
Data Persistence .dat files

Flow Diagram



Tools Used

Programming Languages and Frameworks

Java SE 8+: Core programming language
Java Swing: For graphical user interface
Java AWT: Event handling and basic components
Java Collections Framework: HashMap, ArrayList for data storage

Development Environment

IDE: Visual Studio Code / IntelliJ IDEA / Eclipse
Build Tool: Standard Java Compiler (javac)
Version Control: Git (optional)

Required Libraries

Java Standard Edition Runtime Environment (JRE)
Java Development Kit (JDK)

Key Java Packages Used

javax.swing.* - GUI components
java.awt.* - Event handling and graphics
java.io.* - File I/O operations
java.security.* - Password hashing
java.time.* - Date handling for streak calculation

OOPs concept used & it's Explanation

Abstraction

```
public class DatabaseManager {
    private Map<String, User> users;
    private Map<Integer, Habit> habits;
    // ... private methods for data handling
}
```

Concept: Hiding complex implementation details and showing only essential features to the user.

Implementation: DatabaseManager abstracts all data operations, providing simple public methods like registerUser(), loginUser(), addHabit().

Benefit: Users interact with simple interfaces without worrying about file storage, serialization, or data structure complexities.

Encapsulation

```
public class Habit {
    private int habitId;
    private int userId;
    private String name;
    private String description;
    private LocalDate createdDate;
    private int streak;
    // ... getter and setter methods }
```

Concept: Bundling data and methods that operate on that data within a single unit (class).

Implementation:

- All Habit properties are private
- Access controlled through public getter and setter methods
- Internal streak calculation logic is encapsulated within the class

Benefit: Data integrity, controlled access, and implementation hiding.

Inheritance

```
public class SplashScreen extends JWindow {
   // Inherits JWindow properties and methods}
public class LoginUI extends JFrame {
   // Inherits JFrame properties and methods}
```

Concept: Creating new classes based on existing classes to reuse code.

Implementation: All UI classes extend Swing components (JWindow, JFrame, JPanel) to inherit their properties and methods.

Benefit: Code reusability, consistent behavior, and leveraging existing Swing functionality.

Polymorphism

```
// Method overriding in event handling
loginBtn.addActionListener(e -> login());
registerLink.addMouseListener(new MouseAdapter() {
   public void mouseClicked(MouseEvent e) {
      dispose();
      new RegisterUI().setVisible(true);
   } });
```

Concept: One interface, multiple implementations.

Implementation:

- Method overriding in anonymous inner classes
- Interface-based event handling (ActionListener, MouseAdapter)
- Runtime method resolution for event callbacks

Benefit: Flexible and extensible event handling system.

Exception Handling

```
try (ObjectOutputStream oos = new ObjectOutputStream(
        new FileOutputStream(USERS_FILE))) {
    oos.writeObject(users);
} catch (IOException e) {
    e.printStackTrace();
}
```

Concept: Handling runtime errors gracefully to ensure application stability.

Implementation:

- Try-with-resources for automatic resource management
- File I/O exception handling
- User input validation
- Graceful error messages in GUI

Benefit: Robust application that doesn't crash on unexpected inputs or file errors.

Implementation

DatabaseManager Class (Singleton)

```
public class DatabaseManager {
    private static DatabaseManager instance;
    private Map<String, User> users;
    private Map<Integer, Habit> habits;

    private DatabaseManager() {
        loadData(); // Private constructor
    }

    public static synchronized DatabaseManager getInstance() {
        if (instance == null) {
            instance = new DatabaseManager();
        }
        return instance;
    }

    public boolean registerUser(String username, String password) {
        if (users.containsKey(username)) {
            return false;
        }
        User user = new User(nextUserId++, username,
                    PasswordUtil.hash(password));
        users.put(username, user);
        saveData();
        return true;
    }
}
```

Key Features:

- Singleton pattern ensures single instance
- Thread-safe with synchronized method
- Centralized data management
- Automatic data persistence

FileStorage Class

```
public class FileStorage {
    private static final String USERS_FILE = "users.dat";
    private static final String HABITS_FILE = "habits.dat";

    public static void saveUsers(Map<String, User> users) {
        try (ObjectOutputStream oos = new ObjectOutputStream(
                new FileOutputStream(USERS_FILE))) {
            oos.writeObject(users);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @SuppressWarnings("unchecked")
    public static Map<String, User> loadUsers() {
        File file = new File(USERS_FILE);
        if (!file.exists()) return new HashMap<String, User>();

        try (ObjectInputStream ois = new ObjectInputStream(
                new FileInputStream(USERS_FILE))) {
            return (Map<String, User>) ois.readObject();
        } catch (Exception e) {
            e.printStackTrace();
            return new HashMap<String, User>();
        }
    }
}
```

Data Persistence Strategy:

- Java Object Serialization to binary files
- Separate files for users and habits
- Automatic loading on application start
- Error handling for missing files

Habit Class (Core Business Object)

```
public class Habit {
    private int habitId;
    private int userId;
    private String name;
    private String description;
    private LocalDate createdDate;
    private int streak;
    private LocalDate lastCompleted;

    public boolean completeHabit() {
        LocalDate today = LocalDate.now();
        if (lastCompleted != null && lastCompleted.equals(today)) {
            return false; // Already completed today
        }

        // Streak calculation logic
        if (lastCompleted != null && lastCompleted.plusDays(1).equals(today)) {
            streak++;
        } else {
            streak = 1;
        }

        lastCompleted = today;
        return true;
    }
}
```

Streak Calculation Algorithm:

1. Check if habit was completed today (prevent duplicate completion)
2. If last completed was yesterday, increment streak
3. If last completed was earlier or never completed, set streak to 1
4. Update last completed date to today

DashboardUI Class

```
public class DashboardUI extends JFrame {
    private User currentUser;
    private DatabaseManager db;
    private JPanel habitsPanel;
    private JLabel statsLabel;

    private void loadHabits() {
        habitsPanel.removeAll();
        List<Habit> habits = db.getUserHabits(currentUser.getId());

        if (habits.isEmpty()) {
            // Display empty state message
        } else {
            for (Habit habit : habits) {
                habitsPanel.add(createHabitCard(habit));
            }
        }
        habitsPanel.revalidate();
        habitsPanel.repaint();
    }

    private JPanel createHabitCard(Habit habit) {
        // Creates interactive habit card with:
        // - Habit name and description
        // - Streak counter
        // - Complete button
        // - Delete button
        // - Creation date
    }
}
```

GUI Features:

- Dynamic habit card generation
- Real-time statistics updates
- Responsive layout with scrolling
- Interactive buttons with event handling
- Visual feedback for user actions

Password Security Implementation

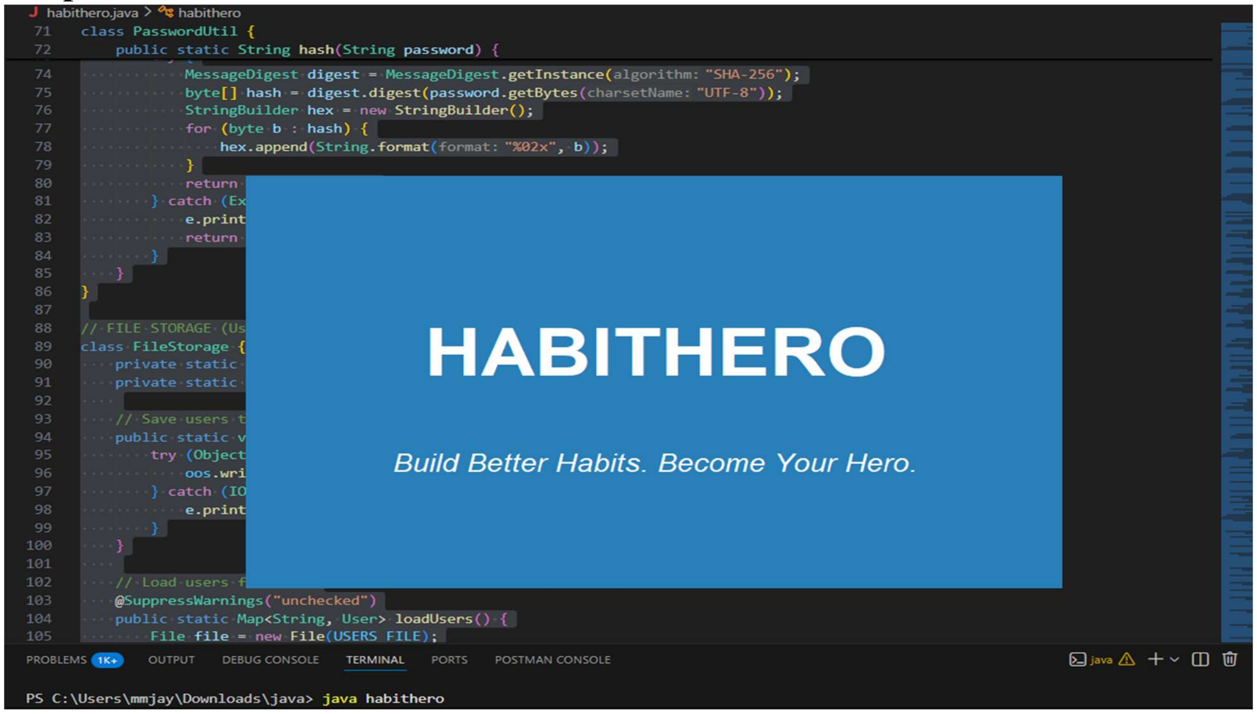
```
public class PasswordUtil {
    public static String hash(String password) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hash = digest.digest(password.getBytes("UTF-8"));
            StringBuilder hex = new StringBuilder();
            for (byte b : hash) {
                hex.append(String.format("%02x", b));
            }
            return hex.toString();
        } catch (Exception e) {
            e.printStackTrace();
            return password;
        }
    }
}
```

Security Features:

- SHA-256 cryptographic hashing
- Salt-free implementation (for simplicity)
- Password never stored in plain text
- Consistent hashing for authentication

Result

1. Splash Screen



habithero.java > habithero

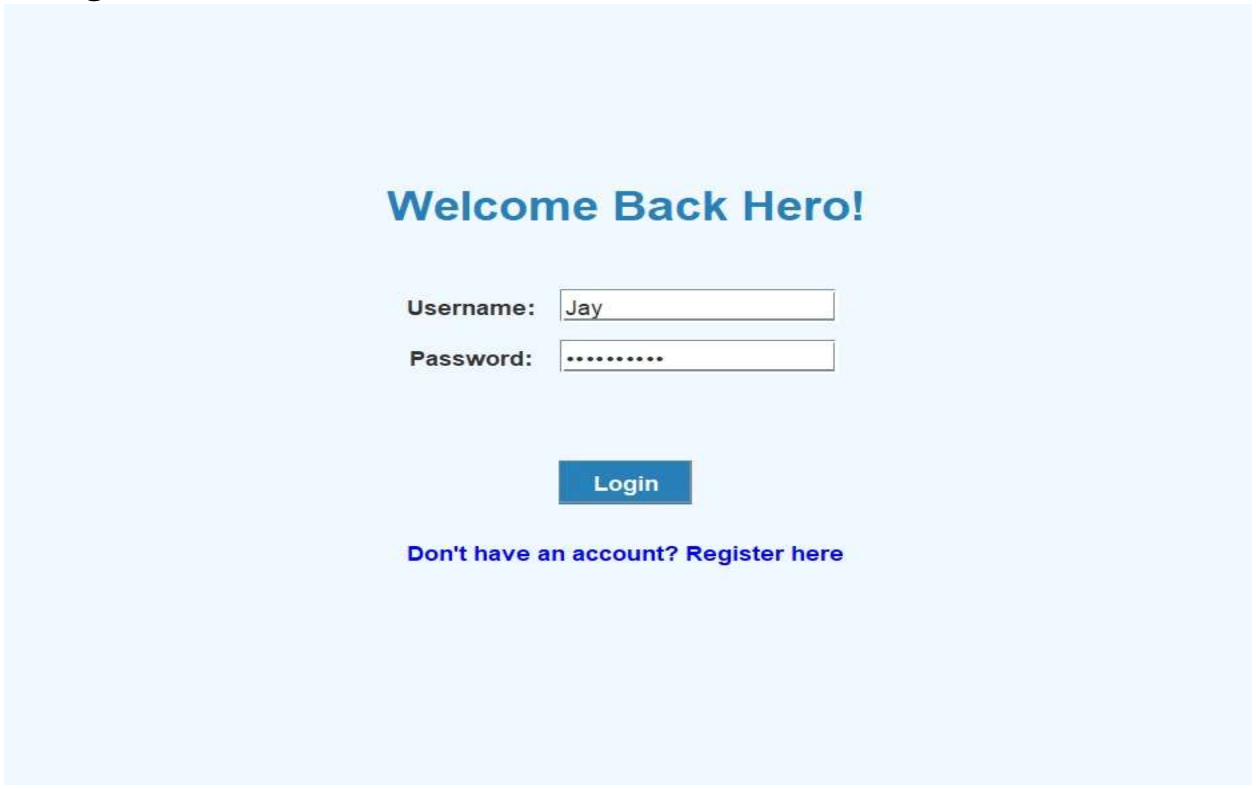
```
class PasswordUtil {
    public static String hash(String password) {
            MessageDigest digest = MessageDigest.getInstance(algorithm: "SHA-256");
            byte[] hash = digest.digest(password.getBytes(charsetName: "UTF-8"));
            StringBuilder hex = new StringBuilder();
            for (byte b : hash) {
                hex.append(String.format(format: "%02x", b));
            }
            return
        } catch (Ex
            e.print
            return
        }
    }
}

// FILE STORAGE (Us
class FileStorage {
    private static
    private static

    // Save users t
    public static v
        try (Object
            oos.wri
        } catch (IO
            e.print
        }
    }

    // Load users f
    @SuppressWarnings("unchecked")
    public static Map<String, User> loadUsers() {
        File file = new File(USERS_FILE);
```

```
PS C:\Users\mmjay\Downloads\java> java habithero
```

HABITHERO

Build Better Habits. Become Your Hero.

2. Login Screen

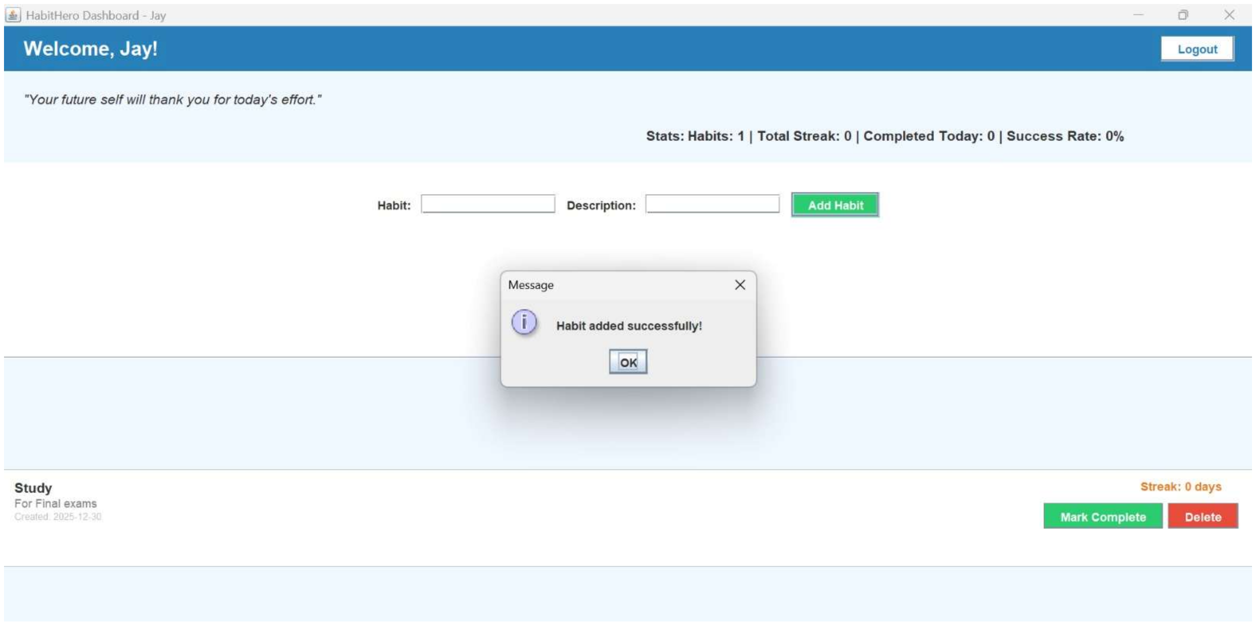


Welcome Back Hero!

Username: Jay

Password: ••••••••••

3. Dashboard Interface


HabitHero Dashboard - Jay

Welcome, Jay!

"Your future self will thank you for today's effort."

Stats: Habits: 1 | Total Streak: 0 | Completed Today: 0 | Success Rate: 0%

Message

Habit added successfully!

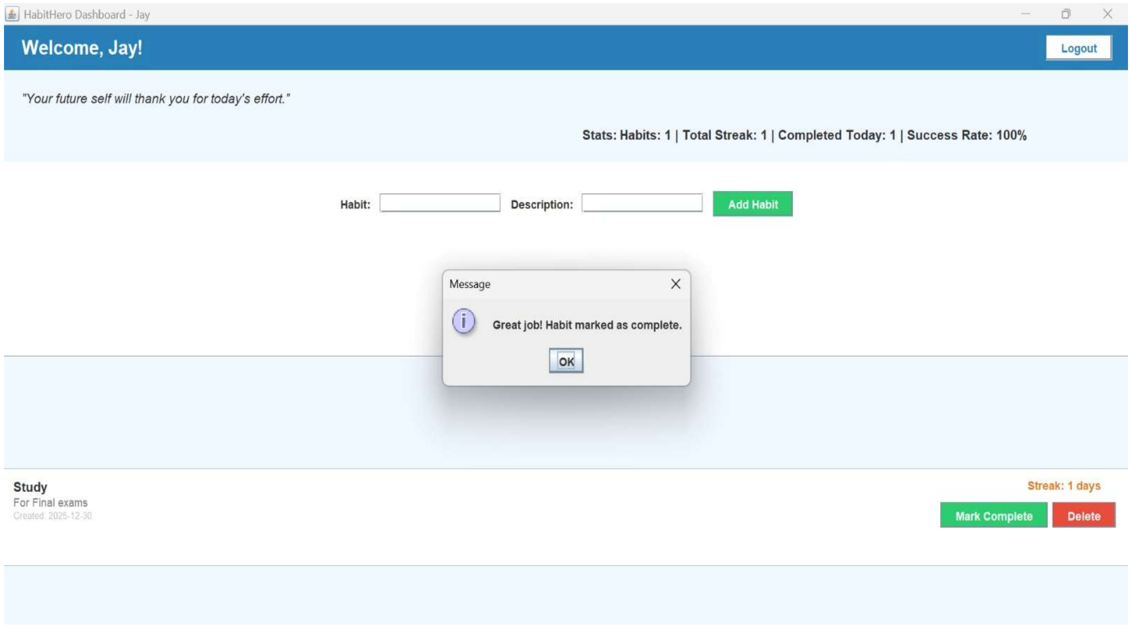
Study
For Final exams
Created: 2025-12-30

Streak: 0 days

4. Habit Completion Feedback

When user clicks "Complete" button:

- Streak counter increments
- Statistics update in real-time
- Success rate percentage recalculates
- Visual confirmation appears


HabitHero Dashboard - Jay

Welcome, Jay!

"Your future self will thank you for today's effort."

Stats: Habits: 1 | Total Streak: 1 | Completed Today: 1 | Success Rate: 100%

Message

Great job! Habit marked as complete.

Study
For Final exams
Created: 2025-12-30

Streak: 1 days

Conclusion

Project Achievements

Successfully implemented a complete habit tracking application with user authentication.
Applied core OOP principles effectively throughout the project.
Created an intuitive graphical user interface using Java Swing.
Implemented secure password hashing using SHA-256 algorithm.
Achieved data persistence using Java serialization.
Developed accurate streak calculation and tracking system.
Provided motivational features through statistics and quotes.

Technical Strengths

Modular Design: Clean separation of concerns between UI, business logic, and data layers.

Scalability: Easy to add new features like reminders, categories, or social features.

User-Friendly: Intuitive interface with clear feedback and error handling.

Secure: Proper authentication with password hashing.

Persistent: Data survives application restarts through file storage.

Current Limitations:

Single-user focused design (no social features).

No reminder or notification system.

Limited reporting and analytics.

No data export functionality.

Desktop-only application (no mobile access).

Future Enhancements:

Push notification reminders for habit completion.

Data export to CSV/Excel for personal analysis.

Habit categories and tagging system.

Progress charts and visualization graphs.

Mobile application version (Android/iOS).

Social features for accountability partners.

Cloud synchronization across devices.

Advanced analytics and habit insights.

References

Official Documentation

Oracle Java Documentation: https://docs.oracle.com/javase/

Java Swing Tutorial: https://docs.oracle.com/javase/tutorial/uiswing/

Java Security API Documentation: https://docs.oracle.com/javase/security/

Java Time API: https://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html

Technical Resources

Bloch, J. (2018). Effective Java Programming Language Guide

Horstmann, C. S. (2019). Core Java Volume I: Fundamentals

Deitel, P. J., & Deitel, H. M. (2017). Java: How to Program

Java Design Patterns: https://java-design-patterns.com/

Object Serialization in Java: https://docs.oracle.com/javase/8/docs/technotes/guides/serialization/

Academic References

Duhigg, C. (2012). The Power of Habit: Why We Do What We Do in Life and Business

Clear, J. (2018). Atomic Habits: An Easy & Proven Way to Build Good Habits & Break Bad Ones

Fogg, B. J. (2019). Tiny Habits: The Small Changes That Change Everything

Online Resources

W3Schools Java Tutorial: https://www.w3schools.com/java/

GeeksforGeeks Java Programming: https://www.geeksforgeeks.org/java/

Stack Overflow Java Community: https://stackoverflow.com/questions/tagged/java

GitHub Java Projects: https://github.com/topics/java