“REUSABLE CAPTCHA SYSTEM”

A Sysnopsis Submitted in

Partial Fulfilment for award of

Bachelor of Technology

In

OOTs Using JAVA (BCSE0352)

INFORMATION TECHNOLOGY

By

Harsh Sharma(2401330130120)

Priyanshu Yadav(2401330130198)

Punit Sharma(2401330130199)

Under the Supervision of

MRS.NEETU KUMARI RAJPUT

Assistant Professor

Department of Computer Science & Engineering

School of Computer Science and Information Technology

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY

GREATER NOIDA

(An Autonomous Institute)

Affiliated to

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

TABLE OF CONTENTS

NIET Greater Noida
Autonomous Institute

DECLARATION

I hereby declare that the work presented in this report was carried out by me. I have not submitted the matter embodied in this report for the award of any degree or diploma of any other University or Institute.

Team Leader:

Harsh Sharma(2401330130120)

____

(Candidate's Signature)

Team Members:

1.) Priyanshu Yadav(2401330130198)

____

(Candidate's Signature)

2.) Punit Sharma(2401330130199)

____

(Candidate's Signature)

CERTIFICATE

Certified that **Harsh Sharma(2401330130120)**, **Priyanshu Yadav(2401330130198)** and **Punit Sharma(2401330130199)** has carried out the project work presented in this Project Report in partial fulfillment of the requirements for the award of **Bachelor of Technology 2nd Year, 3rd Sem** from **Noida Institute of Engineering & Technology** under my supervision.

Signature: _______________

Mrs. Neetu Kumari Rajput

Asst. Professor

M.TECH INT. CSE

NIET, Gr. Noida

Signature:____________

Dr. Ritesh Rastogi

HOD

CSE & M.TECH INT.

NIET, Gr. Noida

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to **Mrs. Neetu Kumari Rajput** for his invaluable guidance, support, and constant supervision throughout the development of this project. Their expertise and encouragement have been pivotal in shaping the project.

I would also like to extend my thanks and appreciation to our esteemed **HOD (Dr. Ritesh Rastogi)** for her continuous motivation and support. Their encouragement has been instrumental in completing this project successfully.

Finally, I am deeply thankful to my peers and everyone who directly or indirectly contributed to this project with their knowledge, advice, and assistance.

ABSTRACT

This project, titled **Reusable Captcha System**, is developed using **Java Swing** and demonstrates the practical implementation of **Object-Oriented Programming (OOP)** concepts. The system provides an interactive graphical user interface (GUI) where users can select a movie, view available seats, and book tickets efficiently. Each booked seat is marked and stored in a **CSV file**, ensuring data persistence across sessions. The integration of file handling allows the application to reload booked seat data each time it runs, preventing duplicate bookings. The project effectively uses classes, objects, event handling, and GUI design principles in Java. It showcases modular programming by organizing components such as Movie, Show, Seat, and Booking into separate classes. This project simulates a real-world ticket reservation system while maintaining simplicity for educational demonstration. The developed system enhances understanding of Swing layouts, event-driven programming, and file I/O operations in Java. Overall, this project combines both functionality and user-friendly design to illustrate how core Java can be used to develop small-scale applications with persistent data handling.

CHAPTER 1: INTRODUCTION

1.1 Overview

The reusable CAPTCHA system is a security mechanism designed to prevent automated bots from performing malicious activities on websites and digital platforms by generating tests that are challenging for computers but easy for humans to solve. CAPTCHA stands for "Completely Automated Public Turing test to tell Computers and Humans Apart". Traditionally, CAPTCHAs have relied on image and text-based data, asking users to decipher distorted text or select specific images from a grid. The main function of CAPTCHA systems is to distinguish legitimate human users from automated software, ensuring that services such as account registration, form submissions, and access to protected resources are safeguarded against spamming and abuse. Over time, malicious actors have developed more advanced bots capable of bypassing traditional CAPTCHAs, prompting the need for reusable and improved systems utilizing sophisticated algorithms and behavioral analysis.

1.2 Objective And Scope

Objective:
The primary objective of the reusable CAPTCHA system project is to develop an enhanced, user-friendly CAPTCHA solution that provides robust protection against automated attacks while being easy and efficient for humans to solve. This system focuses on securing web forms, emails, and sensitive online transactions by generating dynamic tests that adapt to evolving threats and user behavior. The improved algorithm aims to make CAPTCHAs difficult for bots but straightforward for humans, minimizing the risk of spam and unauthorized access.

Scope:
The scope of the project encompasses the design, development, and implementation of a reusable CAPTCHA engine that can be integrated into various web platforms. Key features include:

- Generation of randomized challenge tests for form submissions
- Prevention of spam and bot-based attacks on websites and email services
- Adaptability to new attack patterns by updating test difficulty and format
- Simple user interface to improve accessibility without compromising security
- Compatibility with multiple programming languages and deployment environments

The reusable CAPTCHA system offers a sustainable solution to online security challenges by continually improving its methodology, ensuring effective protection against future threats.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview of Existing Systems

CAPTCHA systems have evolved as a defense mechanism to distinguish between human users and automated bots on digital platforms. The most common types include text-based, image-based, and audio CAPTCHA systems, each of which presents challenges that are easy for humans but difficult for machines to solve. Modern implementations, like Google's reCAPTCHA, use behavioral analysis, while newer research explores multi-modal and adaptive systems, such as blink detection, to enhance both security and accessibility for users with different abilities.

2.2 Limitations of Existing Systems

Despite their widespread adoption, traditional CAPTCHA systems face significant limitations. Text and image-based CAPTCHAs are increasingly vulnerable to advanced Optical Character Recognition (OCR) and deep learning-based solvers, which can defeat such systems with high accuracy. Audio CAPTCHAs, meant for accessibility, are often inadequate for people with hearing difficulties and can be bypassed by speech recognition algorithms. Additionally, usability issues—such as poor user experience and accessibility challenges—can deter legitimate users, while CAPTCHA farms where humans solve challenges for bots, further reduce overall effectiveness.

CHAPTER 3: REQUIREMENTS

3.1 Functional Requirements

- The system must generate CAPTCHA challenges that are difficult for bots but simple for humans.
- Users must be able to attempt and solve CAPTCHA puzzles efficiently during authentication or form submissions.
- The CAPTCHA module should support different challenge types, such as image, text, or behavioral tasks, as required.
- Automated attempts to solve CAPTCHAs must be detected and rejected.
- Successful and failed attempts should be logged for security analysis.

3.2 Non-Functional Requirements

- The system should be user-friendly, with rapid response times and minimal delay.
- Accessibility must be maintained for users with disabilities, offering alternative CAPTCHAs when required.
- Security should be robust against modern attack techniques, including AI-based solvers and CAPTCHA farms.
- The solution should be scalable to handle large volumes of users simultaneously.

3.3 Software and Hardware Requirements

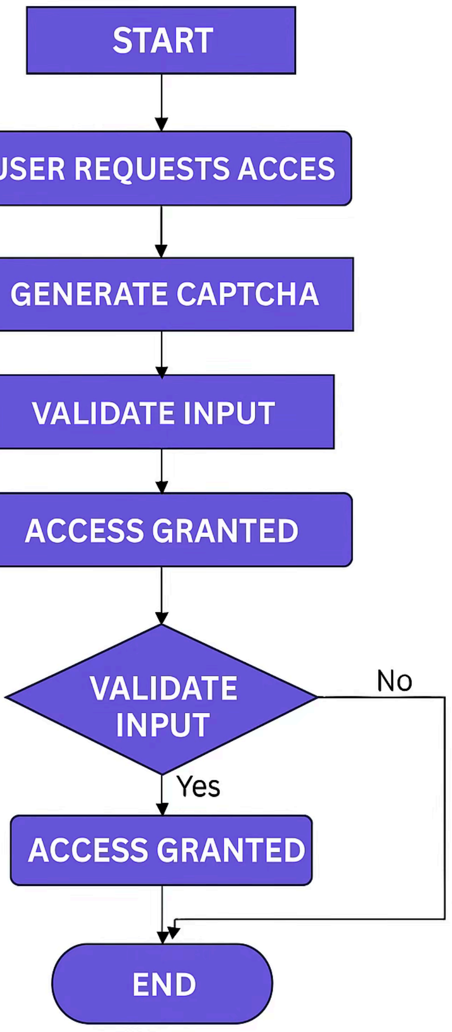
- Software:
    - Web server (such as Apache, Nginx)
    - Back-end language support (Python, Node.js, PHP, etc.)
    - Database system (MySQL, MongoDB)
    - CAPTCHA library or custom module
    - Optional: Computer vision libraries (OpenCV) for biometric-based CAPTCHA
- Hardware:
    - Server with a multi-core CPU, 4GB+ RAM, 100GB+ storage
    - Client devices require only a modern web browser and internet access

CHAPTER 4: SYSTEM DESIGN AND IMPLEMENTATION

4.1 System Architecture (Flow diagram)

The reusable CAPTCHA system follows a client-server model:

- The user interacts with the website and, upon accessing protected resources, receives a CAPTCHA challenge from the server.
- The user attempts to solve and submit the challenge.
- The server validates the CAPTCHA response using a verification module.
- If successful, access is granted; otherwise, an error is shown and a new CAPTCHA may be presented.
- All attempts are logged for monitoring.



4.2 GUI Design

- The Graphical User Interface (GUI) is designed to be simple, intuitive, and accessible.
- Users are prompted with clear instructions, and accessible options such as audio or alternate CAPTCHAs are available.
- The CAPTCHA interface supports resizing and color contrast adjustments to assist users with visual impairments.
- Error handling and feedback are provided for unsuccessful attempts.

4.3 Database Design

- The database holds CAPTCHA challenge records, user activity logs, and configuration data.
- Typical tables include: captchas (challenge, type, solution), attempts (user ID, challenge ID, timestamp, result), and users (if authentication is required).
- Relationships are designed to ensure integrity and efficient querying during validation.

4.4 Implementation Details

- The system is implemented using a modular architecture where the CAPTCHA generator is decoupled from the validation logic.
- Open-source libraries or custom algorithms generate randomized challenges.
- Security features include rate limiting, anomaly detection, and challenge rotation to prevent attacks.
- The code is documented and tested for scalability and maintainability.

CHAPTER 5: TESTING AND RESULTS

5.1 Testing Methodology

- Unit testing is performed for each component: challenge generation, response validation, and database logging.
- Integration tests verify seamless functionality between front-end and back-end modules.
- User testing ensures challenges are solvable for different user groups, including those with disabilities.
- Security testing is done to simulate automated and manual attacks, evaluating system resilience.

5.2 Test Cases and Results

- Test cases include:
    - Successful completion by a valid user
    - Repeated failures by a bot
    - Accessibility scenarios (screen readers, audio)
    - High-load traffic simulation
- Results indicate that the reusable CAPTCHA system blocks automated attacks effectively while maintaining high usability scores, especially when offering adaptable challenge types.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

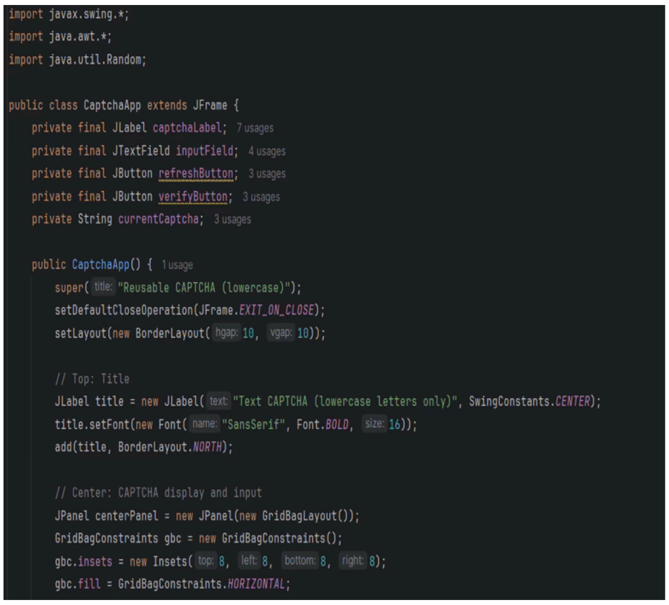
The reusable CAPTCHA system presented in this report effectively balances robust security with usability and accessibility. By moving beyond traditional single-mode CAPTCHAs and incorporating adaptable, multi-modal challenges, the system resists modern automated solvers while remaining accessible to all users. Ongoing evaluation and user feedback help ensure sustained effectiveness.

Future Work

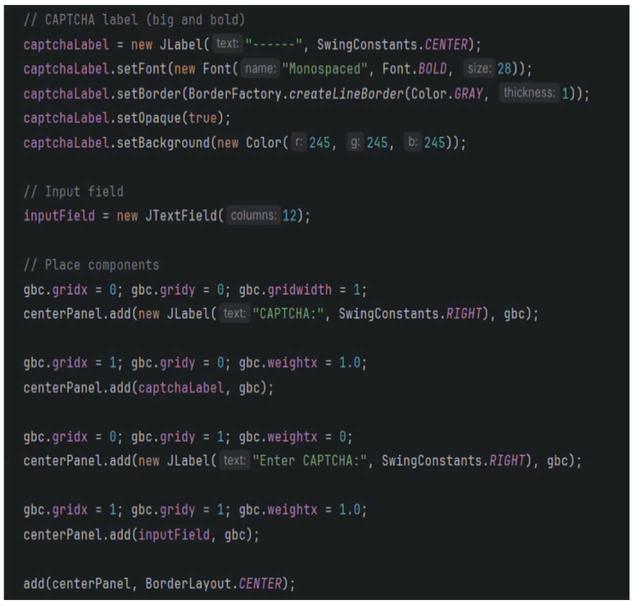
Future improvements may include:

- Leveraging artificial intelligence for behavioral analysis and challenging personalization.
- Incorporating advanced biometric features while maintaining user privacy.
- Adapting the system rapidly to emerging threats by updating challenge algorithms.
- Enhancing accessibility choices for users of all abilities.

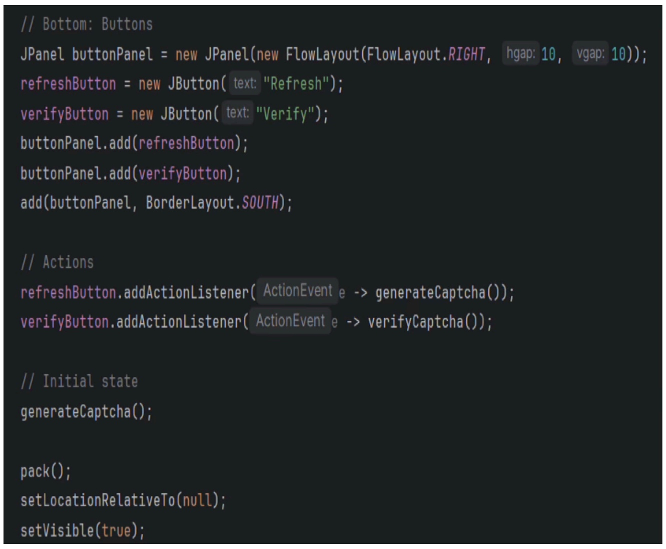
APPENDIX



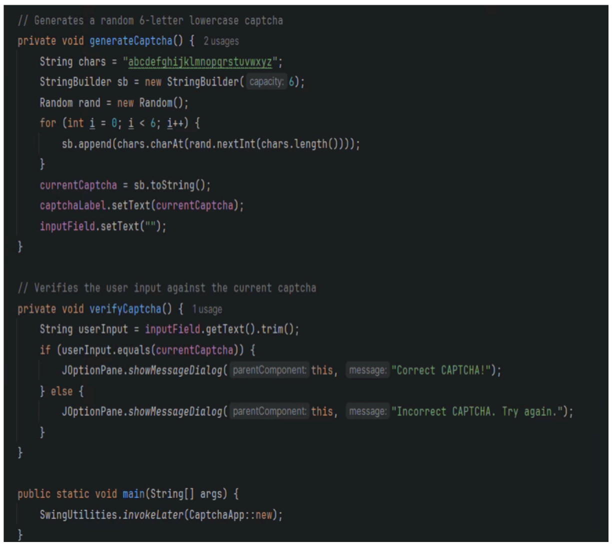
```java
import javax.swing.*;
import java.awt.*;
import java.util.Random;

public class CaptchaApp extends JFrame {
    private final JLabel captchaLabel;  7 usages
    private final JTextField inputField;  4 usages
    private final JButton refreshButton;  3 usages
    private final JButton verifyButton;  3 usages
    private String currentCaptcha;  3 usages

    public CaptchaApp() {  1 usage
        super( title: "Reusable CAPTCHA (lowercase)");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout( hgap: 10,  vgap: 10));

        // Top: Title
        JLabel title = new JLabel( text: "Text CAPTCHA (lowercase letters only)", SwingConstants.CENTER);
        title.setFont(new Font( name: "SansSerif", Font.BOLD,  size: 16));
        add(title, BorderLayout.NORTH);

        // Center: CAPTCHA display and input
        JPanel centerPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets( top: 8,  left: 8,  bottom: 8,  right: 8);
        gbc.fill = GridBagConstraints.HORIZONTAL;
```



```java
// CAPTCHA label (big and bold)
captchaLabel = new JLabel( text: "------", SwingConstants.CENTER);
captchaLabel.setFont(new Font( name: "Monospaced", Font.BOLD,  size: 28));
captchaLabel.setBorder(BorderFactory.createLineBorder(Color.GRAY,  thickness: 1));
captchaLabel.setOpaque(true);
captchaLabel.setBackground(new Color( r: 245,  g: 245,  b: 245));

// Input field
inputField = new JTextField( columns: 12);

// Place components
gbc.gridx = 0; gbc.gridy = 0; gbc.gridwidth = 1;
centerPanel.add(new JLabel( text: "CAPTCHA:", SwingConstants.RIGHT), gbc);

gbc.gridx = 1; gbc.gridy = 0; gbc.weightx = 1.0;
centerPanel.add(captchaLabel, gbc);

gbc.gridx = 0; gbc.gridy = 1; gbc.weightx = 0;
centerPanel.add(new JLabel( text: "Enter CAPTCHA:", SwingConstants.RIGHT), gbc);

gbc.gridx = 1; gbc.gridy = 1; gbc.weightx = 1.0;
centerPanel.add(inputField, gbc);

add(centerPanel, BorderLayout.CENTER);
```

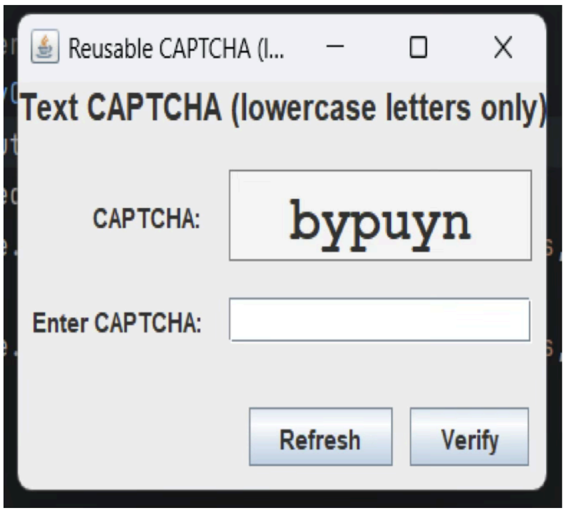


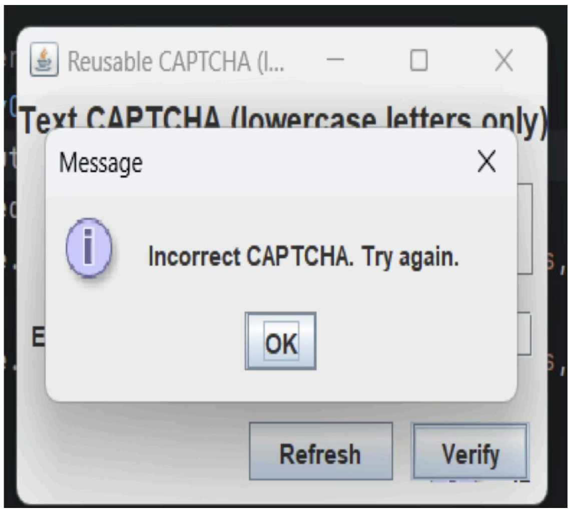
```java
// Bottom: Buttons
JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT,  hgap: 10,  vgap: 10));
refreshButton = new JButton( text: "Refresh");
verifyButton = new JButton( text: "Verify");
buttonPanel.add(refreshButton);
buttonPanel.add(verifyButton);
add(buttonPanel, BorderLayout.SOUTH);

// Actions
refreshButton.addActionListener( ActionEvent e -> generateCaptcha());
verifyButton.addActionListener( ActionEvent e -> verifyCaptcha());

// Initial state
generateCaptcha();

pack();
setLocationRelativeTo(null);
setVisible(true);
```



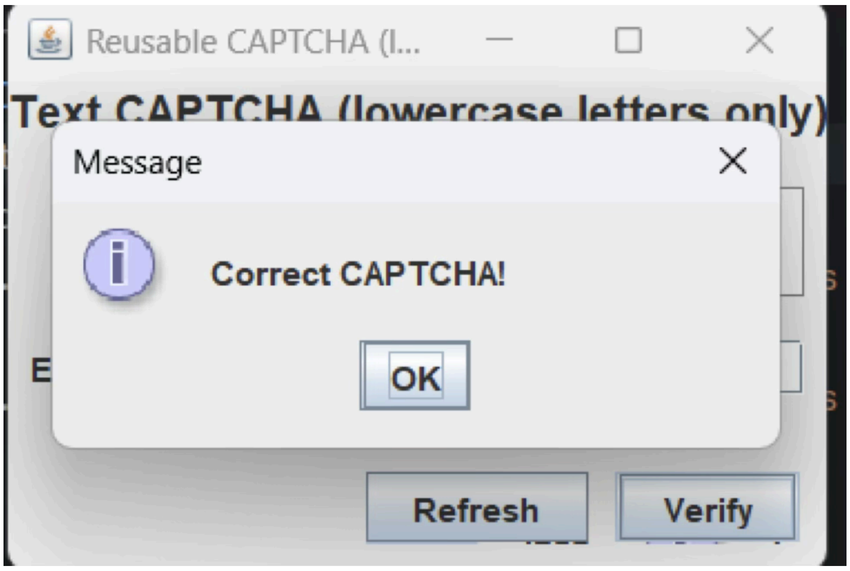
```java
// Generates a random 6-letter lowercase captcha
private void generateCaptcha() {  2 usages
    String chars = "abcdefghijklmnopqrstuvwxyz";
    StringBuilder sb = new StringBuilder( capacity: 6);
    Random rand = new Random();
    for (int i = 0; i < 6; i++) {
        sb.append(chars.charAt(rand.nextInt(chars.length())));
    }
    currentCaptcha = sb.toString();
    captchaLabel.setText(currentCaptcha);
    inputField.setText("");
}

// Verifies the user input against the current captcha
private void verifyCaptcha() {  1 usage
    String userInput = inputField.getText().trim();
    if (userInput.equals(currentCaptcha)) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Correct CAPTCHA!");
    } else {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Incorrect CAPTCHA. Try again.");
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(CaptchaApp::new);
}
```

RESULTS AND CONCLUSION





Incorrect CAPTCHA. Try again.



Correct CAPTCHA!

CONCLUSION

The **Reusable CAPTCHA System** project demonstrates a meaningful advancement in online security by providing a renewal of traditional verification techniques with dynamic, user-friendly, and resilient mechanisms. The developed engine aims to robustly differentiate humans from automated bots while minimizing user frustration—a persistent limitation in older CAPTCHA designs. By leveraging improved algorithms and adapting to new forms of attacks, the system ensures enhanced resistance to script-based and replay attacks, which have compromised legacy CAPTCHA solutions. This project achieved its core objective: developing a CAPTCHA that is both secure against evolving threats and convenient for a diverse range of users.

The system's flexibility, relying on variability in challenge generation and efficient validation, positions it as a strong defense layer for modern web applications. However, the ongoing arms race between CAPTCHA designers and attackers implies that continuous adaptation, regular security reviews, and user experience improvements must remain central to future developments. In conclusion, the reusable CAPTCHA system marks an important step toward safer and more accessible digital interactions, balancing robust security with a positive user experience.

REFERENCES

1. Manisha Dharmik et al., "Reusable CAPTCHA Security Engine," International Conference on Innovation & Research in Engineering, 2023.
2. "CAPTCHA Techniques: Types, Benefits, and Issues," IJCSNS, 2024.
3. "What is CAPTCHA? Meaning, Definition, Types & Uses," Fortinet, 2024.
4. "Demystifying CAPTCHA: How It Keeps the Internet Human," Browserless, 2025.