CAPTCHA GENERATOR

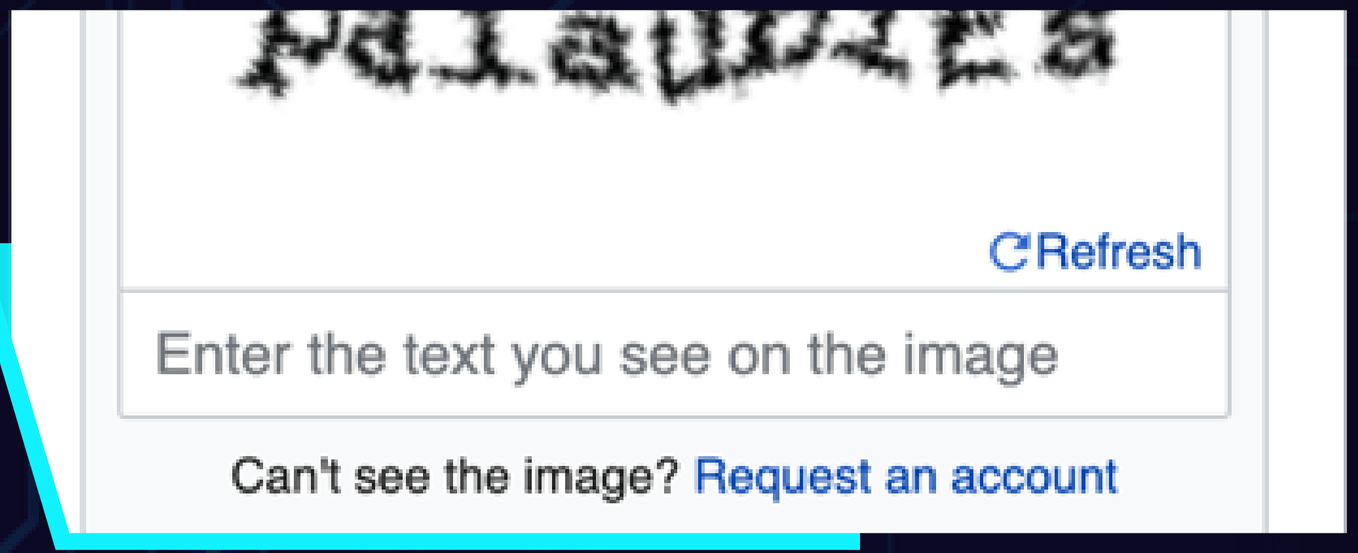
AGENDA

- Introduction
- Importance
- Types
- Mechanism
- Technology stack
- Project setup
- Genereating
- Sample code
- Integration with web application
- Demo
- Conlclusion

INTRODUCTION

What is captcha?

Definition: Completely Automated Public Turing test to tell Computers and Humans Apart.



Purpose Prevent automated bots from accessing websites.

Types

Text-based CAPTCHA: Users are asked to read and type a sequence of distorted characters.

Image-based CAPTCHA: Users might be asked to select certain images that match a description (e.g., "select all images with traffic lights").

Audio CAPTCHA: Provides an audio clip for users to interpret and input, often used to assist visually impaired users.

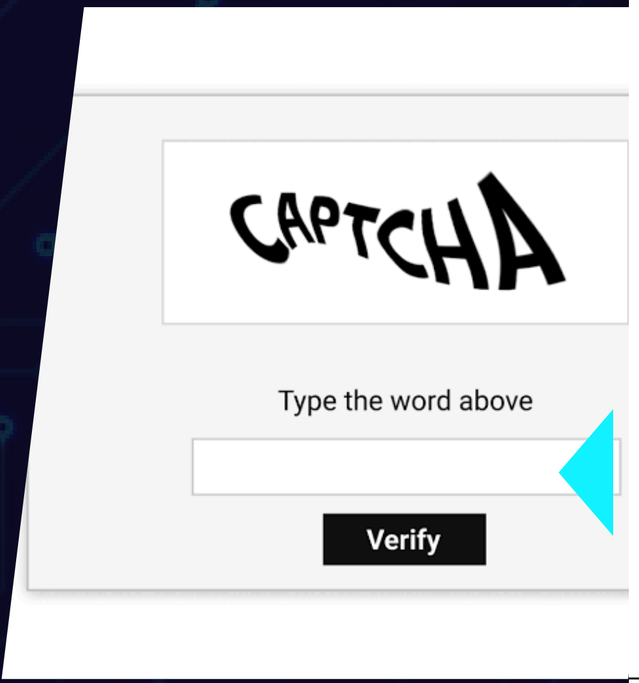
IMPORTANCE

Why Use CAPTCHA?

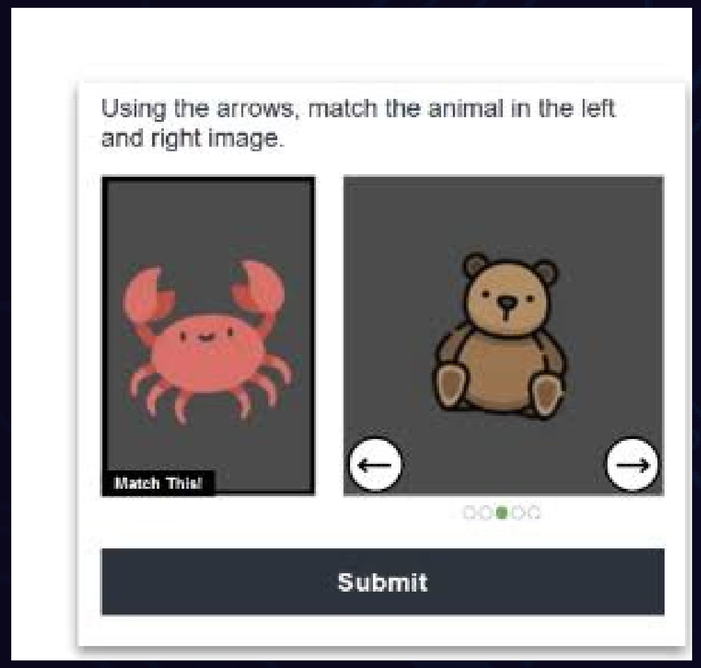
•)Prevents spam and abuse on websites.Use an icon or image of a spam message being blocked to visually represent this point.

•)Enhances security by ensuring interactions are performed by humans.Use a security shield or lock icon to represent enhanced security.

•)Commonly used in forms, sign-ups, login pages, etc.Show icons or small images of a login form, a sign-up form, and a comment section to illustrate these use cases.



MECHANISM



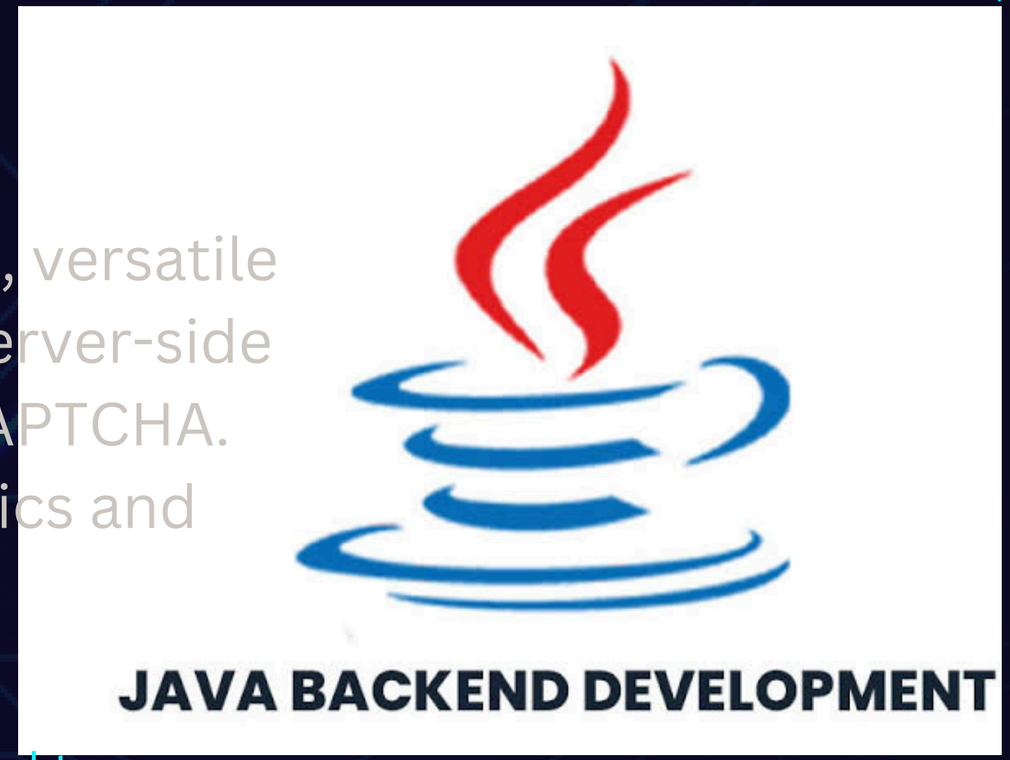
Server Generates Captcha: The server generates a CAPTCHA image or audio file when a user accesses a form or webpage that requires verification. This CAPTCHA contains random elements to ensure uniqueness each time it is generated.

User Solves CAPTCHA : The user is presented with the CAPTCHA and must solve it by entering the text they see, selecting the correct images, or typing out the audio they hear. This step verifies that the user is human.

Server Validates User Input : Once the user submits the form, the server checks the input against the originally generated CAPTCHA. If the input matches, the user is allowed to proceed. If not, the user is typically prompted to try again. .

TECHNOLOGY STACK

•)Java for Back-end Processing : Java is a robust, versatile programming language that is widely used for server-side development. It is well-suited for creating a CAPTCHA. generator due to its strong support for graphics and image process.



•)Servlet for Handling Requests: Java Servlets are used to handle HTTP requests and responses. They are ideal for web applications that require dynamic content generation, such as serving CAPTCHA images to users and validating their inputs.

•)Libraries: Java AWT, Java Graphics2D

SAMPLE CODE

Importing Necessary Libraries: Import Java libraries for image creation and HTTP setup.

Code snippet:

```
import
javax.servlet.*;
import
javax.servlet.http.*;
import java.awt.*;
import
java.awt.image.*;
import java.io.*;
```

Creating the CAPTCHA Image : The main logic for generating a CAPTCHA image. This involves setting the image size, creating a BufferedImage, and drawing distorted text and graphics on it using Graphics2D.

Code snippet:

Java Copy code

```
BufferedImage
bufferedImage = new
BufferedImage(160, 40,
BufferedImage.TYPE_INT_RGB
);
Graphics2D g2d =
bufferedImage.createGraphi
cs();
g2d.setFont(new
Font("Arial", Font.BOLD,
24));
g2d.setColor(Color.WHITE);
g2d.fillRect(0, 0, 160,
40);
g2d.setColor(Color.BLACK);
g2d.drawString("AB12CD",
10, 30);
```

Code snippet:

```
public void
doGet(HttpServletReque
st request,
HttpServletResponse
response)
        throws
ServletException,
IOException {

response.setContentTyp
e("image/jpeg");

ServletOutputStream
out =
response.getOutputStre
am();

ImageIO.write(buffered
Image, "jpg", out);
    out.close();
}
```

Handling HTTP Requests with Servlets : The code to handle HTTP GET requests, generate the CAPTCHA image, and send it as a response to the user's browser.

Project setup:

IDE: Eclipse/IntelliJ IDEA.
Required Libraries: Java Development Kit (JDK), Servlet API.

DEMO CODE


```
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.ImageIO;
import java.util.Random;

public class CaptchaGenerator {
    private static String
generateRandomText(int length) {
        String chars =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456
789";
        Random random = new
Random();
        StringBuilder text = new
StringBuilder(length);
        for (int i = 0; i <
length; i++) {

text.append(chars.charAt(random.ne
xtInt(chars.length())));
        }
        return text.toString();
    }

    public static void
main(String[] args) {
        int width = 160, height =
40;
        String captchaText =
generateRandomText(6);
```

```
        BufferedImage
bufferedImage = new
BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
        Graphics2D g2d =
bufferedImage.createGraphics();
        g2d.setColor(Color.WHITE);
        g2d.fillRect(0, 0, width,
height);
        g2d.setFont(new
Font("Arial", Font.BOLD, 24));
        g2d.setColor(Color.BLACK);

g2d.drawString(captchaText, 10,
30);
        g2d.setColor(Color.GRAY);
        for (int i = 0; i < 5; i+
+) {
            int x1 = new
Random().nextInt(width);
            int y1 = new
Random().nextInt(height);
            int x2 = new
Random().nextInt(width);
            int y2 = new
Random().nextInt(height);
            g2d.drawLine(x1, y1,
x2, y2);
        }
        g2d.dispose();
        try {
```

```
ImageIO.write(bufferedImage,
"jpg", new File("captcha.jpg"));

System.out.println("CAPTCHA image
generated and saved as
captcha.jpg");
        } catch (IOException e) {

System.err.println("Error: " +
e.getMessage());
        }
    }
}
```

CONCLUSION

1.Importance of CAPTCHA:

- Human verification.
- Security Enhancement

2.Technology Utilized:

- Java
- Graphics Libraries

3.Implementation Overview:

- Random text generator
- Image Creation.

4.Challenges and Considerations:

- Balancing usability and security
- Assessibilty.

5. Future improvements

- Advanced Techniques
- user experience.