TASK 3
CODEALPHA

SECURE CODING REVIEW

Java Spring Boot Application

Programming language-java

Application- spring boot application

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.*;
import java.util.HashMap;
import java.util.Map;
@SpringBootApplication
@RestController
public class SimpleApplication {
private static Map<String, String> users = new HashMap<>();
public static void main(String[] args) {
        SpringApplication.run(SimpleApplication.class, args);
    }
@PostMapping("/register")
    public String registerUser(@RequestBody UserRegistrationRequest request) {
        String username = request.getUsername();
        String password = request.getPassword();
    // Simulate user registration
        users.put(username, password);
        return "User registered successfully";
    }
    @GetMapping("/user/{username}")
    public User getUserProfile(@PathVariable String username) {
```

TASK 3
CODEALPHA

```
        // Simulate user profile retrieval
        String password = users.get(username);
        if (password != null) {
            return new User(username, "Sample Profile");
        } else {
            throw new UserNotFoundException("User not found");
        }
    }
    static class UserRegistrationRequest {
        private String username;
        private String password;
        // getters and setters
    }
    static class User {
        private String username;
        private String profile;
    // constructors, getters, and setters
    }
    @ResponseStatus
    static class UserNotFoundException extends RuntimeException {
        public UserNotFoundException(String message) {
            super(message); }
}
}
```

TASK 3
CODEALPHA

Security Vulnerabilities:

Injection Attacks:

- **Issue: Lack of input validation and sanitation in the registration endpoint, making it susceptible to injection attacks.**
- **Recommendation: Use validation frameworks like Hibernate Validator for input validation and sanitize user inputs.**

```
import javax.validation.constraints.NotBlank;
public class UserRegistrationRequest {
   @NotBlank(message = "Username cannot be blank")
   private String username;
   @NotBlank(message = "Password cannot be blank")
   private String password;
  // getters and setters
}
```

Sensitive Data Exposure:

- **Issue: Storing passwords in plaintext.**
- **Recommendation: Always hash passwords using a strong hashing algorithm (e.g., bcrypt) before storing them.**

```
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
// Inside the registration endpoint
String hashedPassword = new BCryptPasswordEncoder().encode(password);
users.put(username, hashedPassword);
```

TASK 3
CODEALPHA

Insecure Direct Object References (IDOR):

- **Issue: Lack of proper authorization checks when retrieving user profiles.**
- **Recommendation: Implement proper authorization checks to ensure that users can only access their own profiles.**

```
import org.springframework.security.core.context.SecurityContextHolder;

// Inside the getUserProfile method

String authenticatedUser =
SecurityContextHolder.getContext().getAuthentication().getName();

if (!authenticatedUser.equals(username)) {

   throw new AccessDeniedException("Unauthorized access");

}
```

General Secure Coding Practices:

Security Headers:

- **Recommendation: Implement security headers to enhance security.**

```
import org.springframework.http.HttpHeaders;

import org.springframework.web.bind.annotation.GetMapping;

// Inside a controller

@GetMapping("/secure-resource")

public ResponseEntity<String> getSecureResource() {

   HttpHeaders headers = new HttpHeaders();

   headers.add(HttpHeaders.CONTENT_SECURITY_POLICY, "default-src 'self'");

   return new ResponseEntity<>("Secure Content", headers, HttpStatus.OK);

}
```

TASK 3
CODEALPHA

Error Handling:

- **Recommendation: Implement proper error handling to avoid exposing sensitive information to users.**

```
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
@ControllerAdvice
public class GlobalExceptionHandler {
@ExceptionHandler(UserNotFoundException.class)
@ResponseStatus(HttpStatus.NOT_FOUND)
   public ResponseEntity<String> handleUserNotFoundException(UserNotFoundException ex)
{
      return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT_FOUND);
}
}
```

HTTPS:

- **Recommendation: Always use HTTPS to encrypt data in transit.**

```
// In application.properties or application.yml
server.port=8443
server.ssl.key-store=classpath:keystore.jks
server.ssl.key-store-password=your_keystore_password
server.ssl.key-password=your_key_password
server.ssl.enabled=true
```

TASK 3
CODEALPHA

Dependencies and Security Auditing:

- **Recommendation: Regularly update dependencies and use tools like OWASP Dependency-Check to identify and fix security vulnerabilities**

```
# Run Dependency-Check to check for vulnerabilities
./dependency-check.sh --project MyProject --scan /path/to/project
```