Quality Attribute 1: Security

Scenario

When the user is not authenticated and they attempt to visit the manage profile page they receive a 401 not authorized page.

Attributes

Source: unauthenticated user
Stimulus: tries to access manage profile page
Artifact: Authorization Component
Environment: train-ticket-system is running normally
Response: 401 HTTP_UNAUTHORIZED error, blocks access
Measure: unauthorized error response is shown within 1 standard deviation of system average response time for 90% of users

Source Code

src/main/java/edu/drexel/TrainDemo/SecurityConfiguration.java:

```
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests(a -> a
                    .antMatchers( ...antPatterns: "/user/**").authenticated()
                    .anyRequest().permitAll()
            )
            .exceptionHandling(e -> e
                    .authenticationEntryPoint(new HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED))
            )
```

src/test/java/edu/drexel/TrainDemo/user/UserControllerTest.java:

```
@Test
public void test_user_controller_endpoints_start_with_user() {
    Method[] methods = UserController.class.getDeclaredMethods();

    for (Method method : methods) {
        Annotation[] methodAnnotations = method.getDeclaredAnnotations();
        for (Annotation methodAnnotation : methodAnnotations) {
            Class annotationType = methodAnnotation.annotationType();
            if (annotationType == GetMapping.class || annotationType == PostMapping.class) {
                String value = extractValueFromAnnotation(methodAnnotation);
                assert value.contains("/user");
            }
        }
    }
}
```

Quality Attribute 2: Modifiability

Scenario

A developer can change the database used by modifying the system application.properties file and setting the `spring.datasource.url` and `spring.jpa.properties.hibernate.dialect` properties. Spring Boot will alter its runtime behavior to support the new database.

Attributes

Source: Developer
Stimulus: modifying the application properties
Artifact: Spring Boot Framework
Environment: runtime
Response: Application runs successfully with the new database connection
Measure: Runtime database dependency is managed automatically

Source Code

src/main/resources/application.properties:

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
# Database connection settings
spring.datasource.url=jdbc:postgresql://192.168.99.100:5432/TrainDemo
```

Quality Attribute 3: Interoperability

Scenario

Our User Component interacts with the Authorization Component by selecting the “Sign in with GitHub” button which redirects the user to the /oauth2/authorization/github endpoint which redirects the user to GitHub’s login portal. Upon successfully entering their credentials, GitHub then redirects the user back to the homepage. The homepage then invokes the user component to display the user’s account information in the top-right section of the navigation bar.

Attributes

Source: User Component
Stimulus: User selects “Sign in with GitHub” button
Artifact: spring-boot-starter-oauth2-client
Environment: train-ticket-system, TrainDemo database, and GitHub are online and running normally
Response: The home page is rendered with the user account’s information in the top-right section of the navigation bar.
Response measure: Our information included correctly 99.9% of time

Source Code

pom.xml:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-client</artifactId>
</dependency>
```

src/main/resources/templates/base.html:

```
<a class="btn btn-block btn-social btn-github" href="/oauth2/authorization/github" role="button">
    <span class="fa fa-github"></span> Sign in with Github
</a>
```

src/main/java/edu/drexel/TrainDemo/user/controllers/UserController.java:

```
@GetMapping("/user")
@ResponseBody
public User getUserInfo(@AuthenticationPrincipal OAuth2User principal) {
    Integer id = principal.getAttribute( name: "id");
    String defaultName = principal.getAttribute( name: "name");
    return userService.getOrCreateUser(id, defaultName);
}
```