Spring framework was selected over JAAS due to the large number of deprecated dependencies subsequently Spring framework assists in encapsulating modules that perform authentication for users. This technology tool allows the graceful implementation of both authenticating and authorizing actors, specifically in the library Spring Security. In addition, this provides actual login and access control logic. The system is constrained to Java implementation and runs on a Windows environment via command line interface, on Visual Studio Code or IntelliJ IDE.

Firstly, the SpringBoot module "@SpringBootApplication" is a wrapper class that serves as an entry point for the application. It is composed of three other modules: "@SecurityConfig", "@EnableAutoConfig", and "@ComponentScan". This system is responsible for authorizing actors. The system requires authentication of actors via a password. Particularly the "UserDetailsService" is a central authentication module, storing their information in memory. Upon launching the entry point, a login is prompted and if incorrect, the actor is not authenticated as they are not confirmed to be who they are without the correct password. The passwords are encrypted at rest by a hash using the Spring Framework.

User accounts:

- bob/password
- alice/password

The actors types are admin, user, and public. What this translates to is that certain types or in this case roles have more access than others on the endpoints, designated by an URL. Whereas admin or user must be logged in to access their roles and endpoint URLs. Given incorrect login, the system reprompts login information. The system supports validating input and is sanitized by Spring framework.

The system supports the authorization of actors by ensuring that an authenticated actor has the rights to access a service for that designated role. For example, in the public URL, anyone can see that endpoint. Alice, who is a user, can only login and see that she is authenticated, but bob who is an admin, has greater rights in the more restricted URL. This is verified by the ".hasRole()" command. Hence, proper identification is verified; the system also contains credential-based authentication in the form of username/password). One late bound component that exists here is the "security.core.Authentication" module as it is bound at runtime, and only created after a user has provided credentials.

Endpoints:

- /
- /public
- /user

- /admin

Each endpoint has their own “@RestController” which is composed of “@Controller” and “@ResponseBody”, that is a RESTful web service. For instance, in the AdminController class, return “admin endpoint” is returned as a raw HTTP response body. The “/” endpoint occurs by default after a successful login, in which the user can know who and what role they are. These above aforementioned tactics satisfy the security quality attribute.