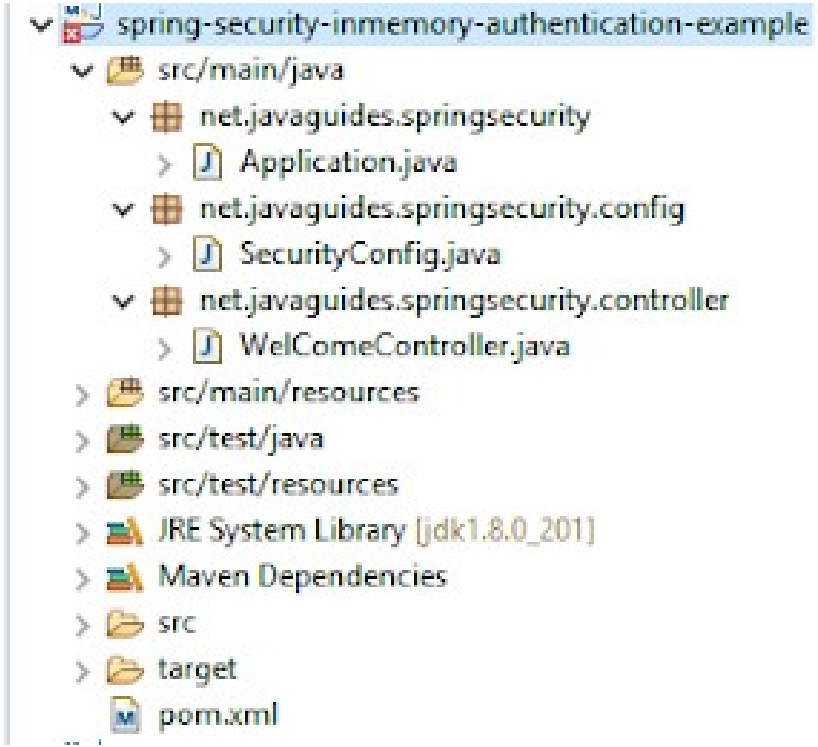
Spring Security In Memory Authentication Example

Spring Security AuthenticationManager to use Spring Security in-memory authentication and add multiple users with different attributes, authorities, and roles.

Creating a Spring Boot Application



Dependencies

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Application.properties

```
spring.security.user.name=abc

spring.security.user.password=aaaAAA123

spring.main.allow-bean-definition-overriding=true

==========================================================================

@Bean

public BCryptPasswordEncoder getPasswordEncode() {

return new BCryptPasswordEncoder();

}
```

Spring Security In-Memory Authentication

```
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http.csrf().disable().authorizeRequests().anyRequest().authenticated().and().httpBasic().and()
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    }
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception

{ auth.inMemoryAuthentication().withUser("tarkesh").password(getPasswordEncode().encode("aaaAAA123"))
            .roles("USER").and().withUser("admin")
           .password(getPasswordEncode().encode("aaaAAA123")).credentialsExpired(true).accountExpired(tr
ue).accountLocked(true)
           .authorities("WRITE_PRIVILEGES", "READ_PRIVILEGES").roles("ADMIN");
    }
}
```

In-Memory Authentication

We have used the AuthenticationManagerBuilder with the **InMemoryUserDetailsManagerConfigurer** to configure the Spring Security In-Memory Authentication.

we are using a builder pattern to create multiple users with different attributes, authorities, and roles. This automatically configures a **UserDetailsService** which we can use.

Note that we have added a password storage format, for plain text, add {aaaAAA123}. Prior to Spring Security 5.0, the default **PasswordEncoder** was **NoOpPasswordEncoder** which required plain text passwords. In Spring Security 5, the default is **DelegatingPasswordEncoder**, which required Password Storage Format like **{noop}**.

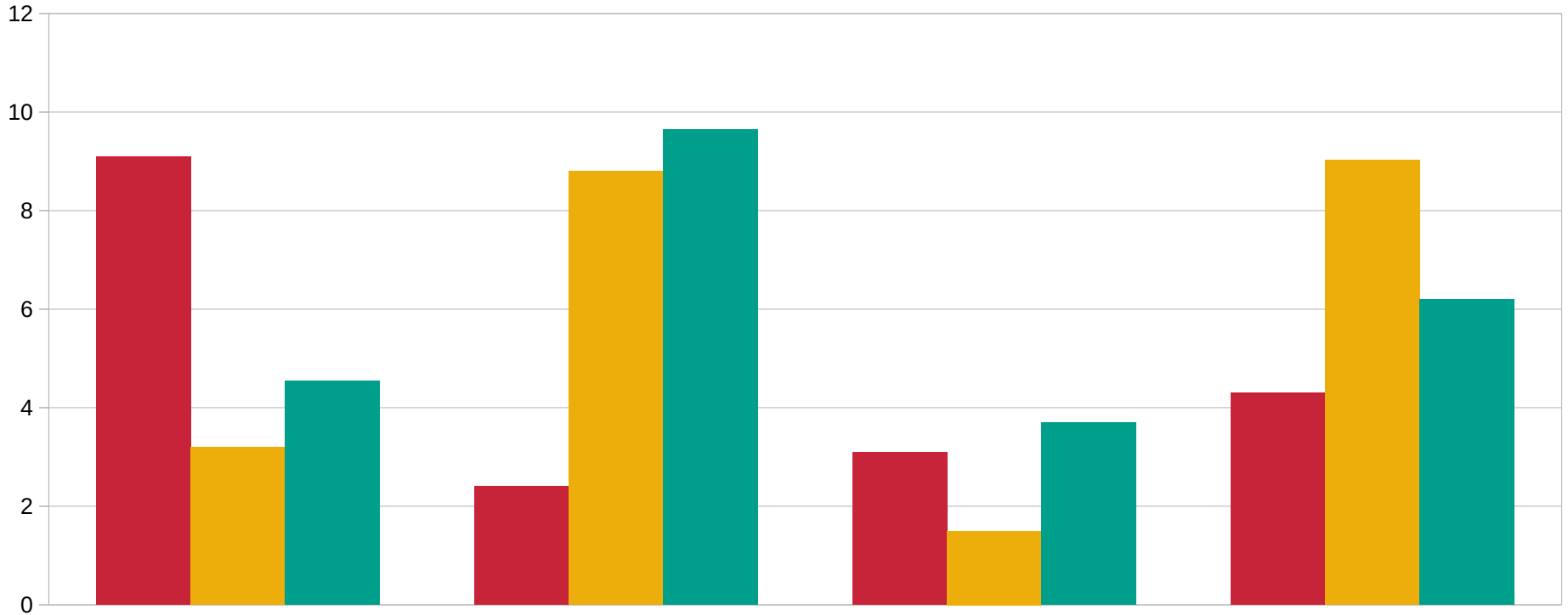
In-Memory Authentication

```
@RestController
public class WelComeController {

    @GetMapping("/")
    public String greeting(Authentication authentication) {

        String userName = authentication.getName();

        return "Spring Security In-memory Authentication Example - Welcome " + userName;
    }
}
```



12
10
8
6
4
2
0