**5.authenciation providers**

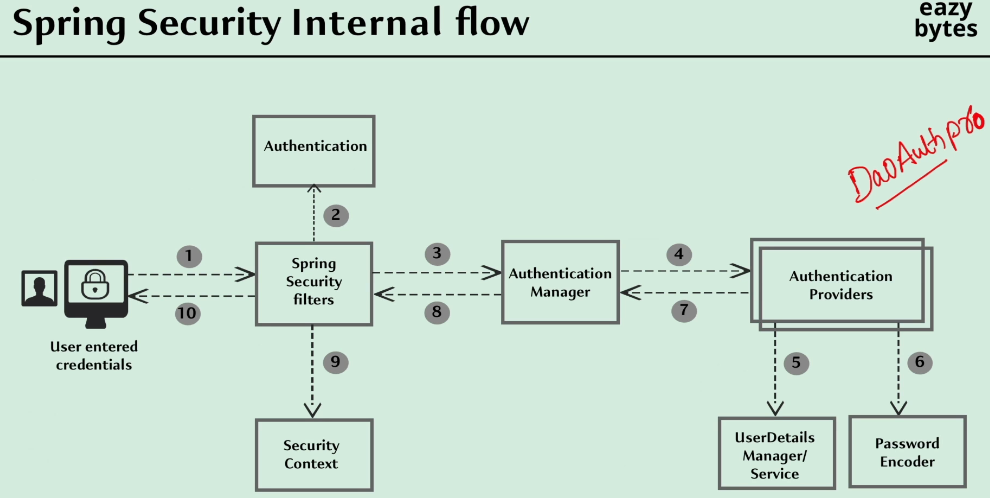
The very first question that you may have is the default authentication provider,

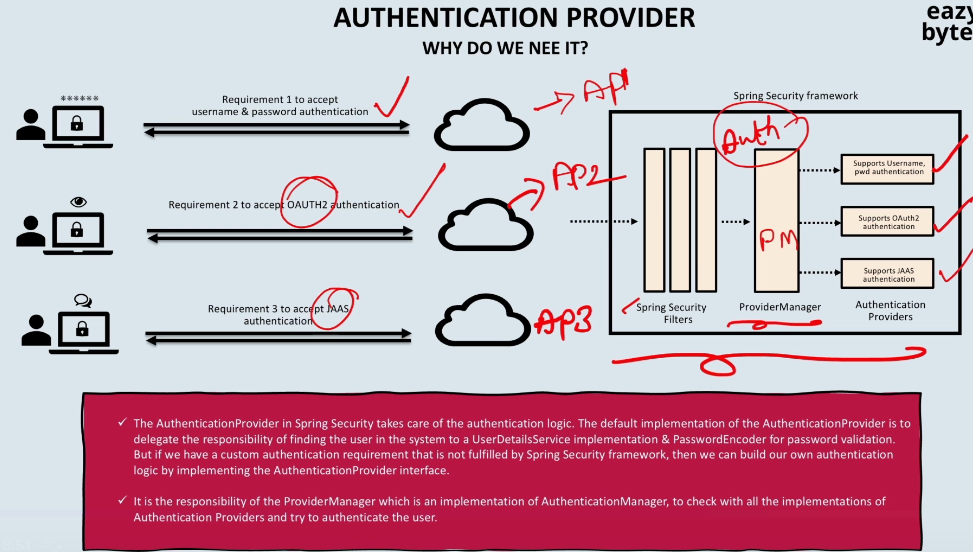
which is DaoAuthenticationProvider is doing a great job and we are able to perform the authentication of the user by just loading the user details from the database

with our own UserDetailsService implementation.

So, everything looks perfect.

Then, why should I learn about AuthenticationProvider and under which scenario we need to provide our own custom Authentication Provider?



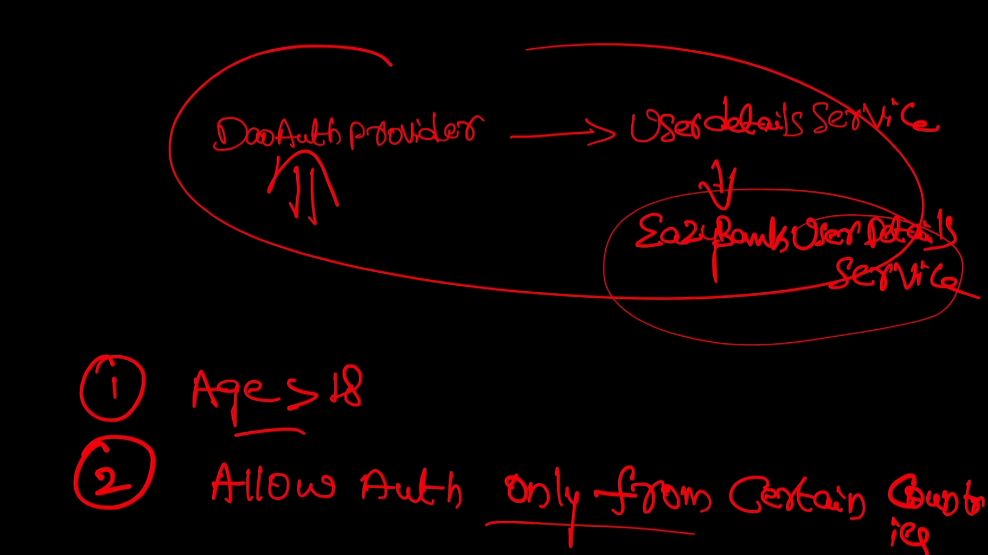


"How do my ProviderManager will know which authentication provider to invoke?"

That's where the beauty of Authentication object will come into picture.

So, based upon the type of your Authentication object, whether it is an UsernamePasswordAuthenticationToken, or whether it is a JAAS authentication token type.

So, based upon the Authentication object type, the ProviderManager is going to identify the corresponding Authentication Provider.



DaoAuthenticationProvider is not going to help you because whatever logic that you're going to write inside the EazyBankUserDetailsService, it is only going to help you to load the user details from the storage system.

But it is not going to help you to validate the age details or country details, or any other details that are required for your custom authentication.

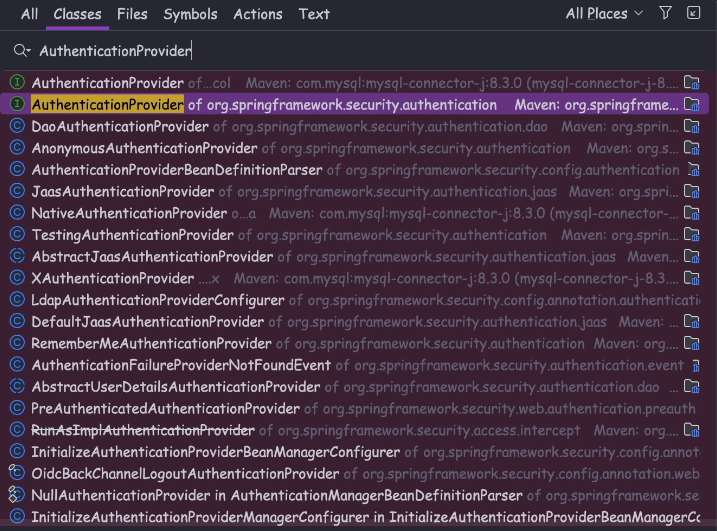
To accommodate these kind of requirements, you need to define

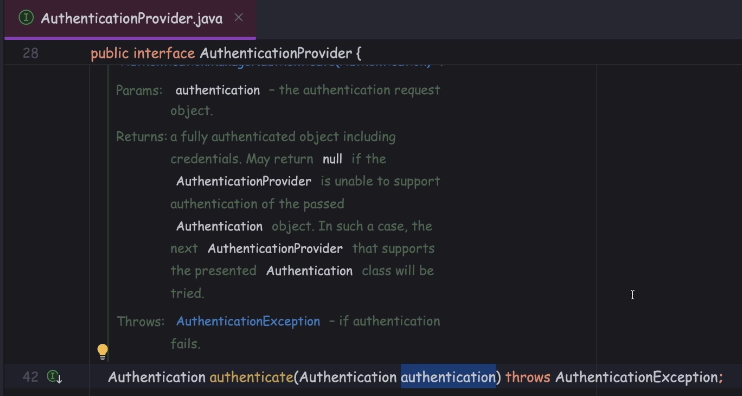
your own AuthenticationProvider implementation. And inside your own AuthenticationProvider implementation, once the user details are loaded

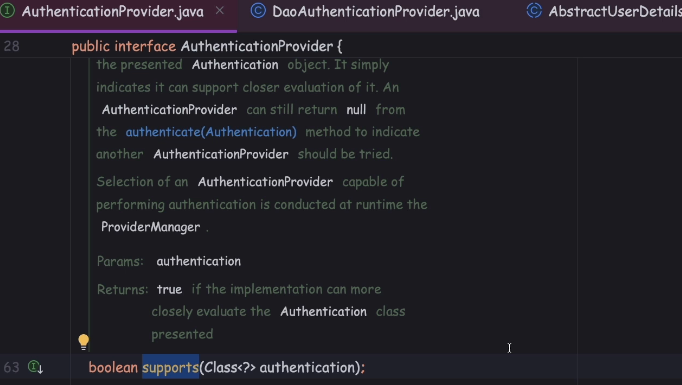
from your UserDetailsService, you can perform the additional checks

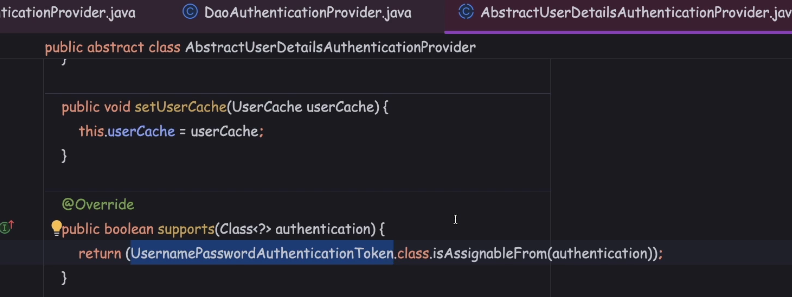
like age check or country check, or any other custom checks.

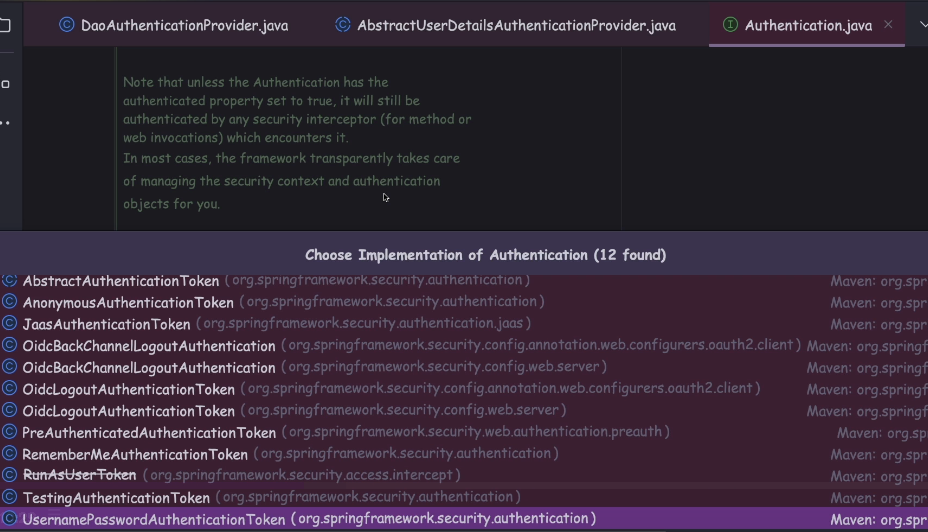
Understanding AuthenticationProvider methods









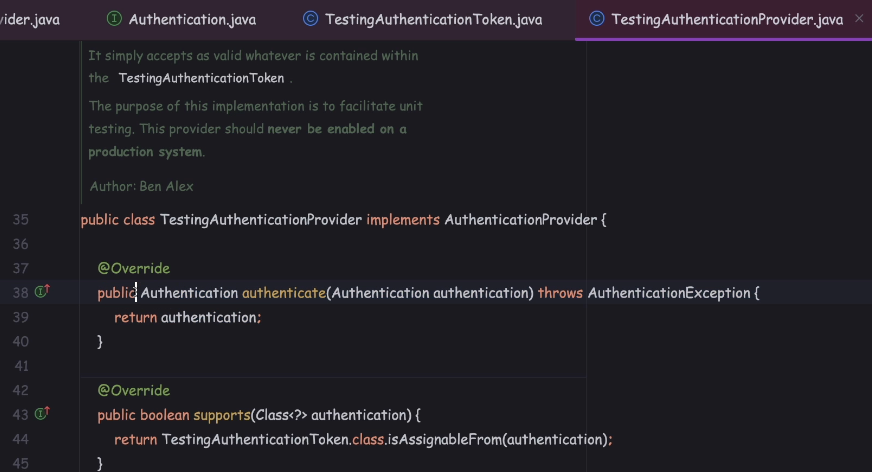


OidcLogoutAuthenticationToken, Oidc token is the token which is going to be used

in the scenario of Oauth2 authentication.

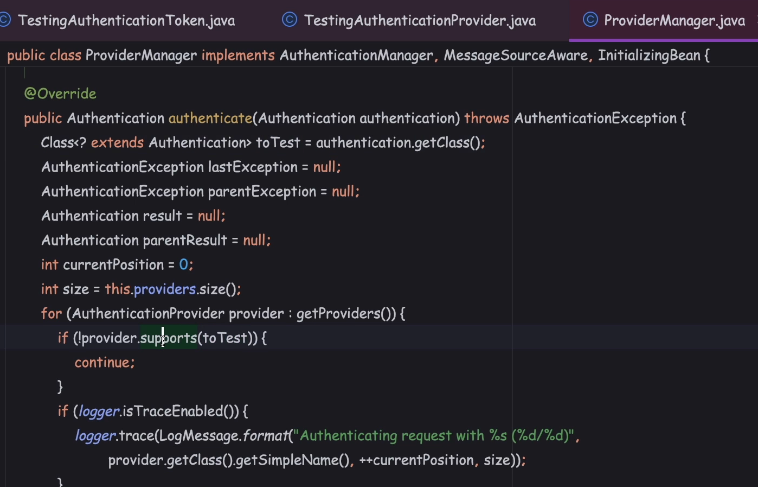
who is anonymous, which means this user is trying to access the application without any security. For example, a public page or a public API the end user is trying to access, in such scenarios the AnonymousAuthenticationToken

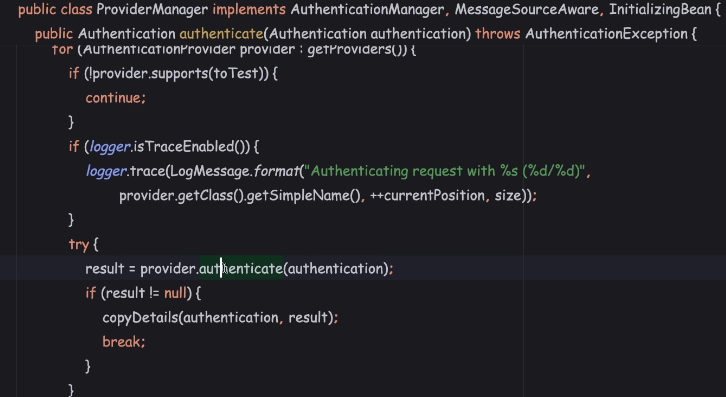
is going to be created by the spring security framework.

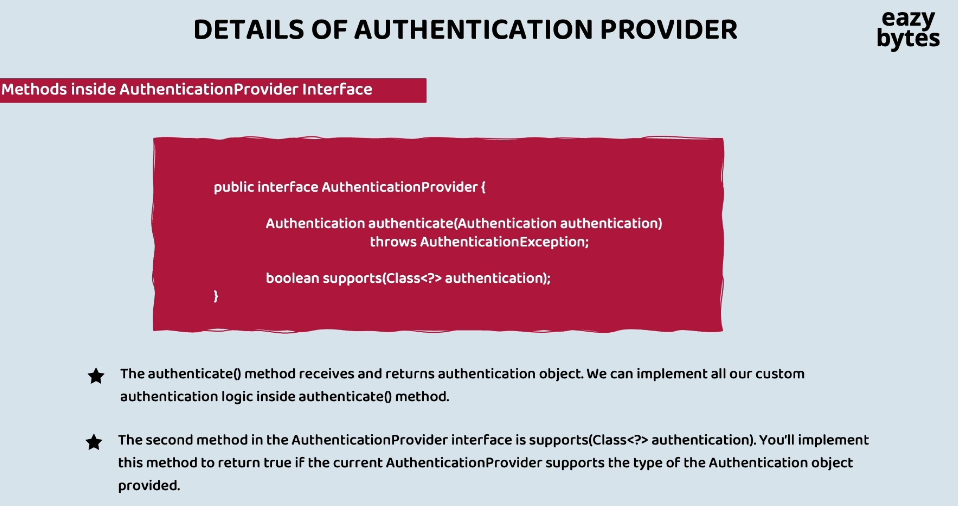


whenever you try to use the TestingAuthenticationToken, the actual authentication is not going to happen. And this type of token is going to be super helpful

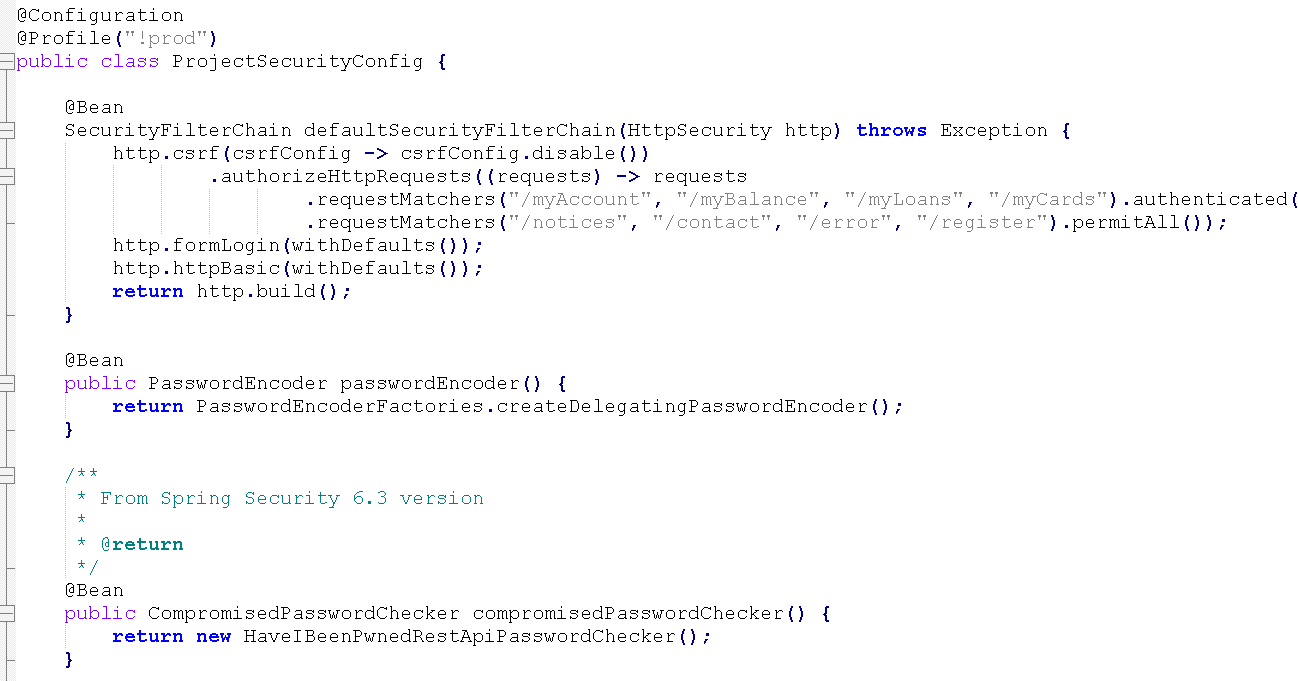
in the scenarios of unit testing.



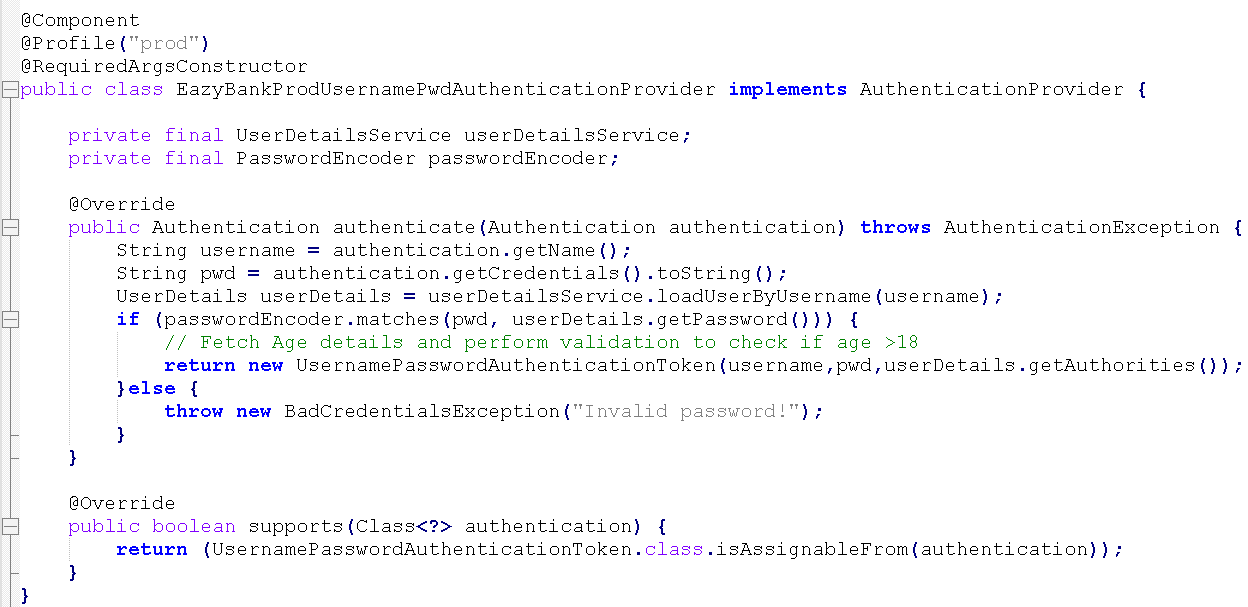




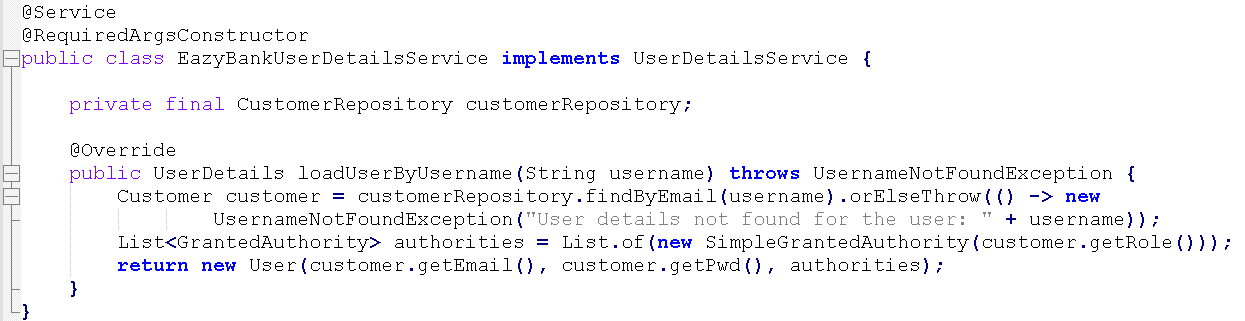
Implementing and Customizing the AuthenticationProvider inside our application ; section6

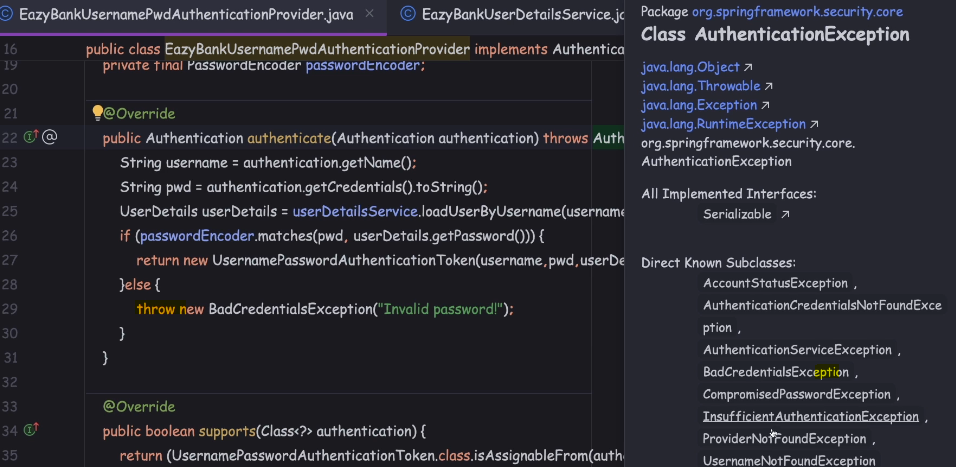


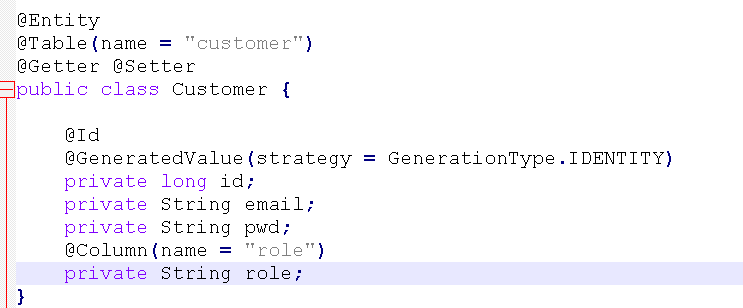


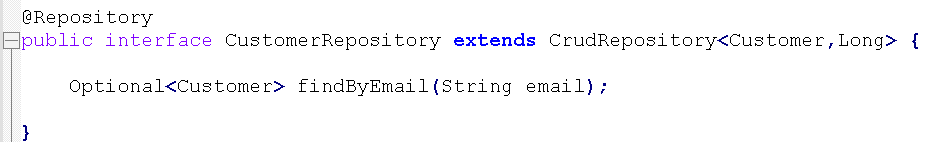




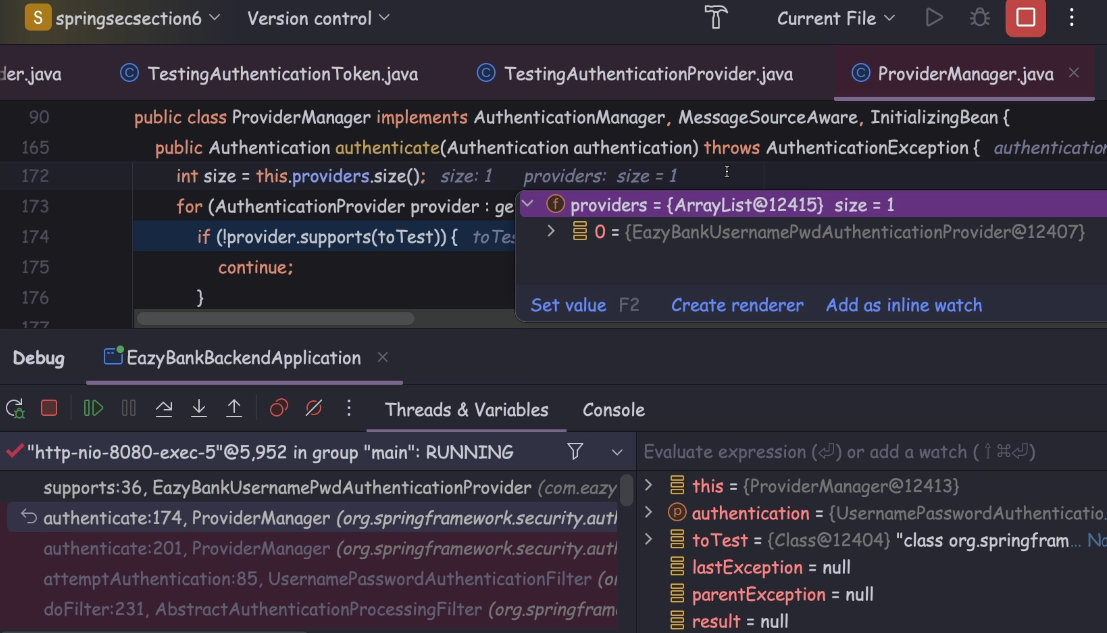


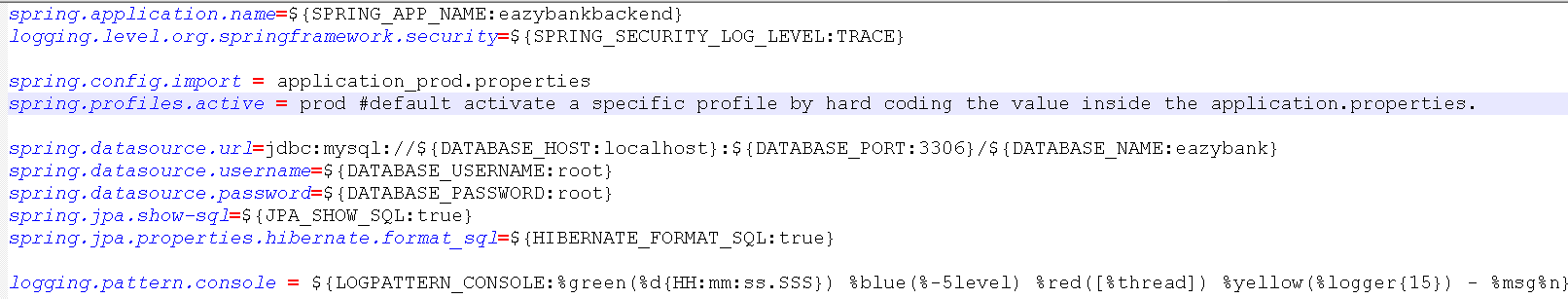


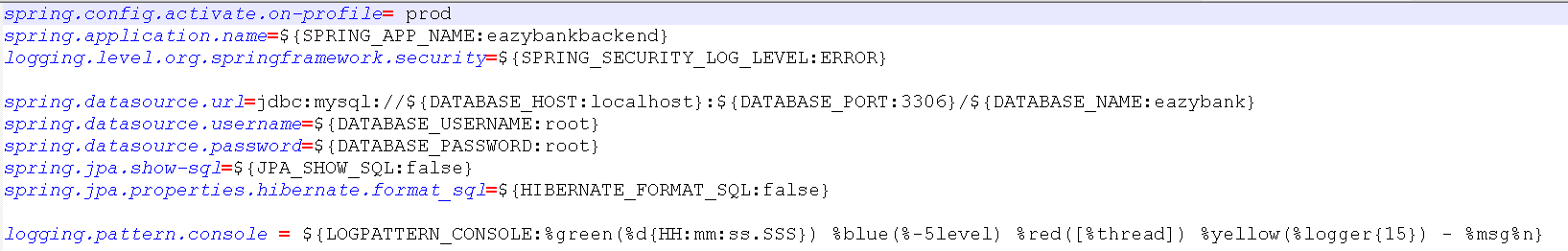




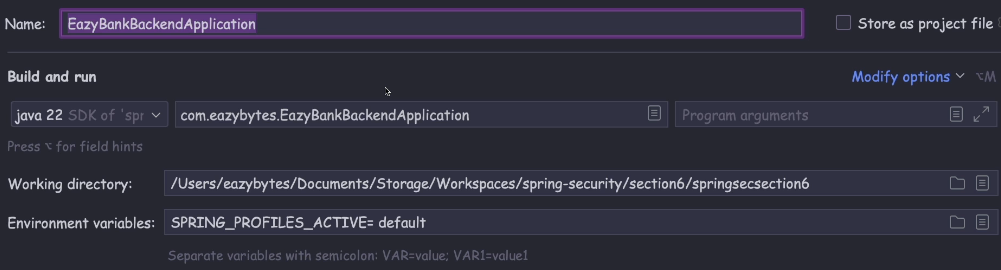








We know by default Spring Boot is going to activate the default profile and in the scenario of default profile, the Spring Boot is going to look for the application.properties.



Top of Form

If we have a custom authentication requirement that is not fulfilled by Spring Security framework then we can build our own authentication logic by implementing the below interface ?

**org.springframework.security.authentication.AuthenticationProvider**

Top of Form

Which of the following is the correct definition of AuthenticationProvider interface

public interface AuthenticationProvider {

* 2. Authentication authenticate(Authentication authentication)
  3. throws AuthenticationException;
  5. boolean supports(Class<?> authentication);
  7. }

Top of Form

Which of the following interface represents the authenticated user once the request has been processed by the AuthenticationProvider.authenticate() method inside Spring Security framework?

**org.springframework.security.core.Authentication**

Top of Form

Which of the following interface is true regarding Authentication interface inside Spring Security?

**Authentication interface extends Principal interface from Java.**