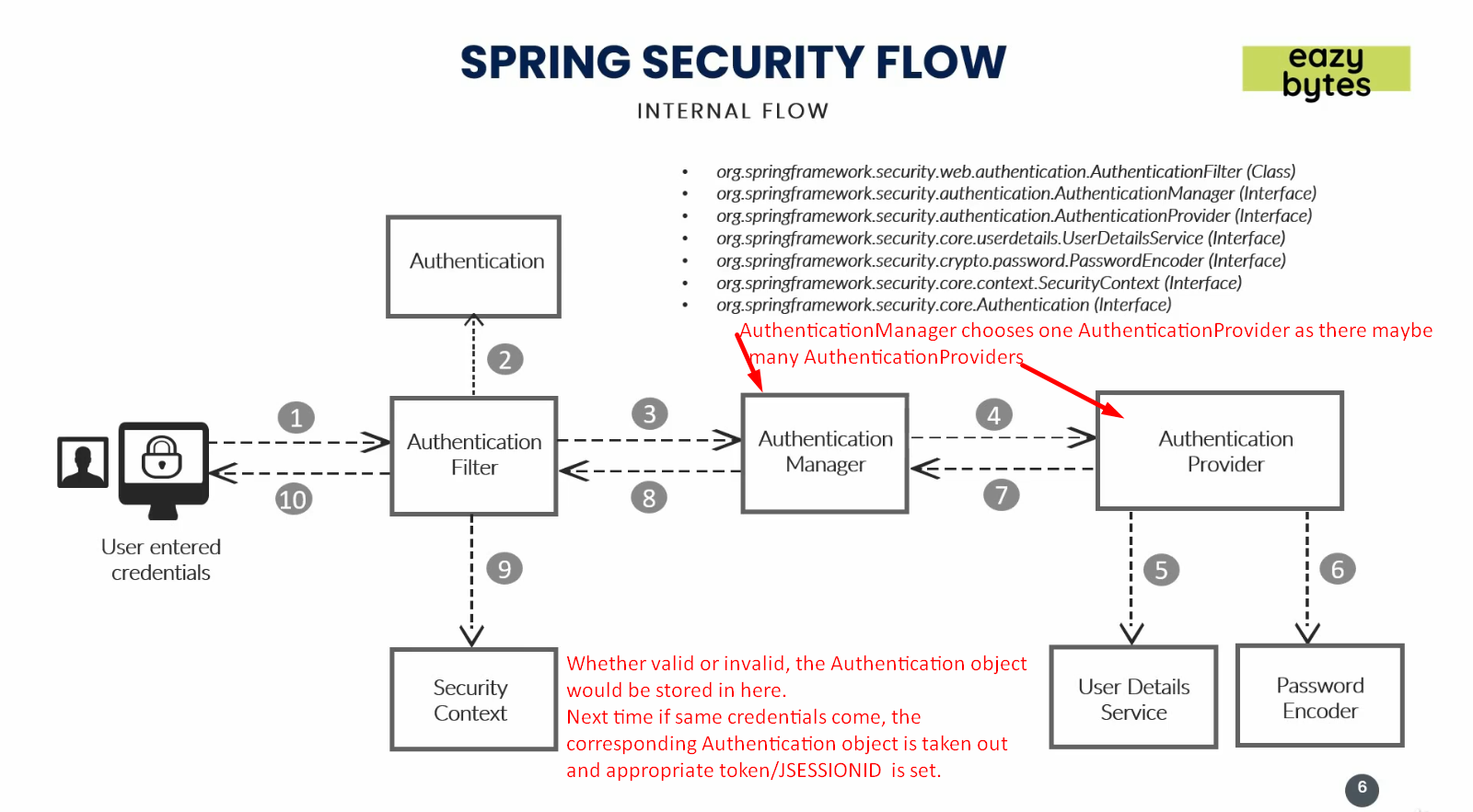
1. **So Far:**
   1. Created Spring Boot App.
   2. Enabled Spring Security.
   3. Exposed RESTful API.
   4. Tested through browser and POSTMAN.
2. **Agenda**:
   1. How Spring Security works internally when a request comes from Front-end App or 3rd Party RESTful API Consumers?
3. This following diagram is the flow.  
   
4. **Components in Spring Security**:
   1. **Authentication Filter**:
      1. Present in Spring Security Framework.
      2. Will intercepts all the incoming requests.
      3. It will try to convert the received authentication details such as username, password into **Authentication Object**.  
         This object is a base where all the validation of user credentials will be validated in further steps.
      4. Then this Authentication object is passed to **AuthenticationManager.**
   2. **AuthenticationManager:**
      1. There may be many Authentication Providers such as
         1. DB Provider.
         2. LDAP Provider.
         3. OAuth Provider.
      2. Authentication Provider provides the stored credentials against which the received credentials are matched.
      3. So, Basically, AuthenticationManager identifies the appropriate Authentication Provider and sends the Authentication Object to that provider.
   3. **Authentication Provider**:
      1. This is the place where all our business logic related to security/authentication is implemented.
      2. Authentication Providers use internally two other interfaces such as
         1. UserDetailsService
         2. PasswordEncoder
   4. **User Detail Service**:
      1. It is an interface which holds user schema like how my user should look like.
         1. Schema about the user details.
      2. Actually, the properties such as username, password, OTP etc.
   5. **Password Encoder**:
      1. Interface about how to encrypt and decrypt passwords.
      2. Many such as
         1. NoPasswordEncoder.
         2. DecryptPasswordEncoder.
   6. **Authentication Object**:
      1. Would hold the followings after validation process.
         1. Is Valid.
         2. What Roles.
         3. What Authorities etc.
   7. **Security Context**:
      1. This is a bean (interface) which will be managed by Spring container.
      2. The Authentication object which is retuned by AuthenticationManager is stored in Security Context.
      3. This Authentication object contains info about whether authentication is valid or invalid.
      4. Remember, JSESSION token is created on successful authentication.  
         So, in this scenario, Authentication will hold that token and this Authentication object will be stored in SpringSecurityContext and SpringSecurityContext in Spring container.
      5. If the same credentials info is passed, these steps are not followed rather Authentication Object from Security context is fetched and user is validated.