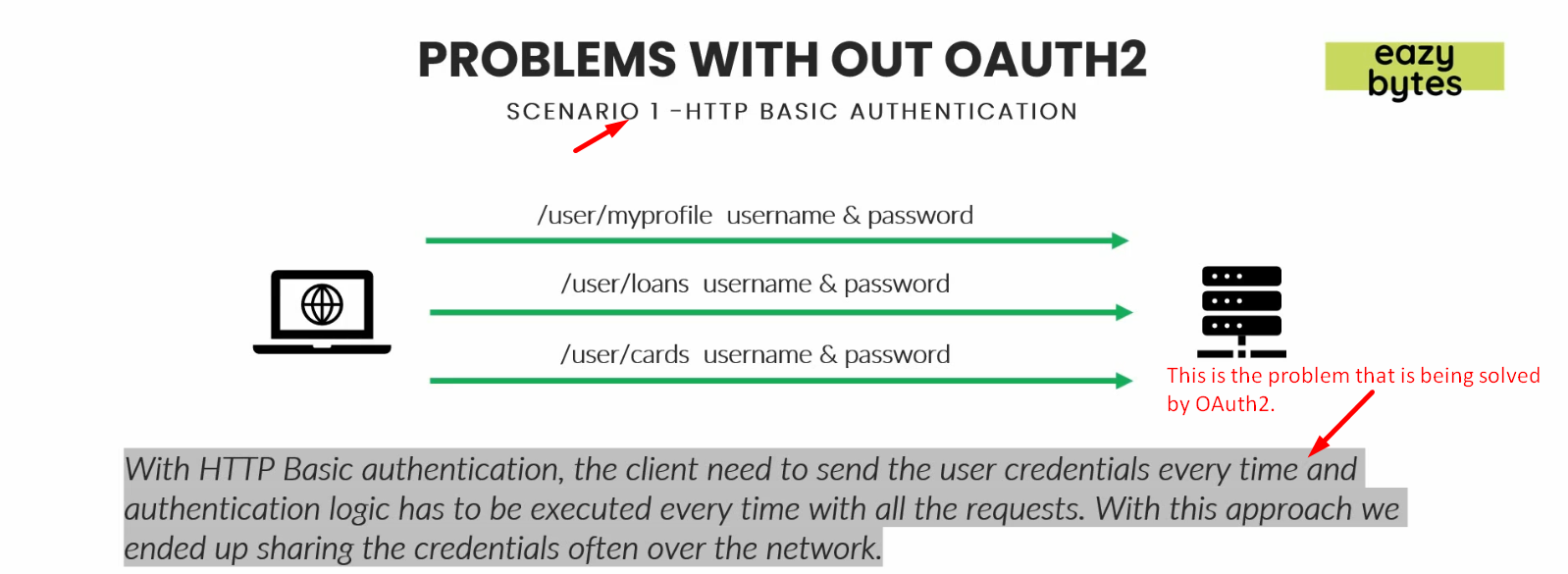
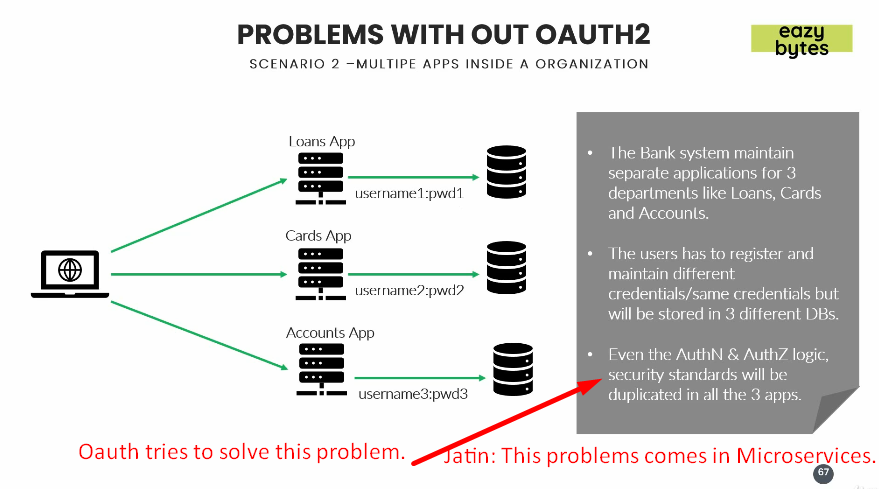
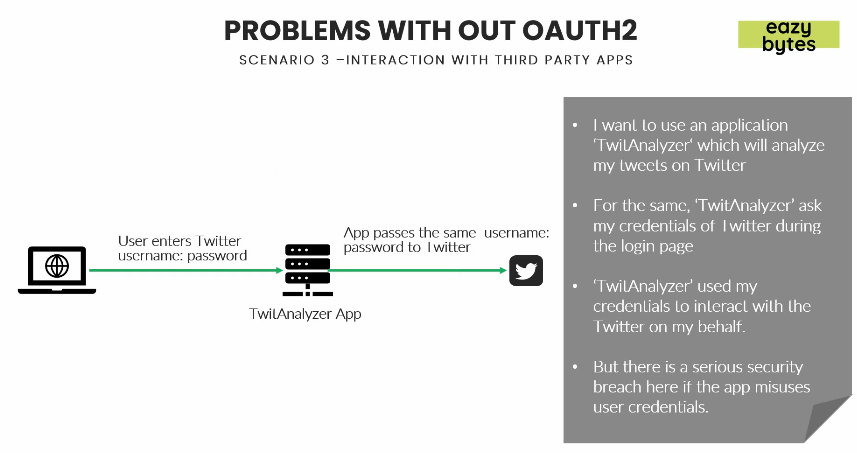
1. **Agenda**:
   1. We will discuss a lot about what the Framework which is the most standard and common authentication and authorization framework followed by many organizations.
2. So before we understand what the Oauth framework works and how Spring Security Framework will help us in adopting Oauth framework, let’s try to spend a few minutes on understanding what are the most common problems that Oauth is trying to solve in the authentication and authorization flows inside any application?
3. **Problems Solved by Oauth** :
   1. The very first and most common scenario that Oauth framework is trying to solve is the sharing of the credentials of the user on the network unnecessarily can be stopped by using Oauth framework efficiently.  
      NOTE: In the below description, the mentor is talking about RESTful Webservice where each time you’re passing credentials.  
      If you use HTTP Basic Authentication b/w client and backend interactions, we may end up sharing the credentials again and again over the network for each and every request that you are going to make to the backend.  
      At the same time, this will also force the backend app to execute the authentication logic again and again for each request.

Such problems can be solved by using tokens such as JWT, JSESSIONID.  


* 1. **Scenario 2**:
     1. Now suppose, my EasyBank app is maintaining 3 different independent applications 🡺 Loan App, Cards App, Accounts App.
     2. Even though your end user is common to all these 3 applications, you end up following one of two approaches.
        1. You will ask your customers to register separately in all these 3 applications either with   
           different credentials or same credentials.   
           So, in such scenarios, you ended up maintaining the authentication and authorization logic in 3 different applications and storing the client related credentials information in different databases.  
           At max, you can avoid serving the credentials in multiple DBs and save them in single DB.  
           But still you 3 applications have duplicate code for   
           authentication, authorization or security standards.  
           So, Oauth framework also helps you in these kinds of scenarios by adopting a common authorization server.   
           That means all your authentication and authorization flows will be kept separately in a different server and   
           that server will be leveraged whenever we try to authenticate and authorize inside any of these 3 applications.  
           
     3. **Scenario 3**:  
        
        1. Think of a scenario where you as end user want to use an app “Tweet Analyzer” which is built by a start up.
        2. This Tweet analyzer app will try to analyze all my tweets, retweets, likes and comments inside my Tweet account and then give statistics to me.
        3. If Tweet analyzer app is not using Oauth framework, they will ask me to enter my tweeter credentials inside their own app. I will end up entering my actual Tweeter Credentials which will be used by Twitter Analyzer app to interact with the actual Twitter to get my tweets information.   
           But sharing tweeter credentials is very serious security concern. Such as
           1. What if Tweeter Analyser App misuses my credentials?
           2. What if they post tweets on my behalf using Tweet API?

1. So, all these scenarios can be avoided by using Oauth Framework.