**AUTHENTICATION FLOW IN EMERGE**

**1. When user Login from Login page**

**1.1 Existing Authentication flow in eMerge**

**1.1.1** In the login page, user enter his/her userId and password.

**1.1.2** This userId and password will be verified against the user details from eMerge “User” table.

**1.1.3** Once verified, a session will be created, user details will be saved in the sessionObject and user will be logged in and redirected to the “Home.aspx” page of eMerge and can then access the application.

**1.1.4** If at any point of time, user clicks on logout button of eMerge application or some unhandled exception occurs, his/her session will be cleared.

**1.2** **Proposed Authentication flow in eMergeAPI**

**1.2.1** Create a table ( let’s say USR\_AuthMaster ) in eMerge database with columns “UserId”, “SessionId”, ”TokenId”, “Timestamp”.

**1.2.2 In eMerge Side**

**1.2.2.1** In the login page, user enter his/her userId and password.

**1.2.2.2** Create a session with default values and get a sessionId.

**1.2.2.3** Pass this sessionId in the requestDTO of login Api call along with username and password.

* + 1. **In eMergeAPI Side**

**1.2.3.1** Call the validateLogin() to validate the user against the user details from eMerge “User” table.

**1.2.3.2 CASE** - If user is not a valid user

* + - * 1. **In eMergeAPI Side**

**a)** Don’t generate JWT token

**b)** Return false in the loginResponseDTO

* + - * 1. **In eMerge side**

**a)** Clear the session that we created and show the appropriate message to user on UI

**1.2.3.3 CASE** – If user is a valid user

* + - * 1. **In eMergeAPI Side**

**a)** Generate a JWT token with timeout greater than that of session created in eMerge

**b)** Save the sessionId in claims section of this JWT Token

**c)** Save this JWT token and timestamp against the userId and sessionId in “USR\_AuthMaster” table that we created.

**d)** Create a cache memory, save token, sessionId in cache.

**e)** Send true with user details and JWT token in the loginResponseDTO

* + - * 1. **In eMerge side**

**a)** Set the user details and the JWT token from loginResponseDTO in the sessionObject.

**b)** User will be logged in and redirected to the “Home.aspx” page of eMerge and can then access the application.

**c)** If at any point of time, user clicks on logout button of eMerge application or some unhandled exception occurs, his/her session will be cleared.

**2. Flow for subsequent API calls after successful login API call**

**2.1 eMerge Side**

**2.1.1** Send the data in the requestDTO with sessionObject and token in header section of the request.

**2.2 eMergeAPI Side**

**2.2.1** Validate the sessionId and token from the requestDTO with the sessionId and Token stored in cache memory.

**2.2.2** **CASE** - If vaild api call

**2.2.2.1** Check if the token timeout is between 50% to 100%.

**See the 4th section for regeneration of token in detail**

**2.2.2.1.1 CASE** – If yes

**a) eMergeAPI Side**

**1.** Then regenerate the JWT Token, update the “USR\_AuthMaster” table with this token and timestamp against the sessionId and userId

**2.** Delete those records from the table “USR\_AuthMaster” for which timestamp difference is greater than that of session timestamp.

**3.** Also update this token in cache against the sessionId.

**4.** Return the token with the responseDTO

**b) eMerge Side**

**1.** Show the appropriate message to user on UI and also update the token in the sessionObject

**2.2.2.1.2 CASE** – If No

**a) eMergeAPI Side**

**1.** Return the responseDTO

**b) eMerge Side**

**2.** Show the appropriate message to user on UI

**2.2.3 CASE** – If invalid api call

* + - 1. **eMergeAPI Side**

**2.2.3.1.1** Throw the exception stating invalid token/user

* + - 1. **eMerge Side**

**2.2.3.2.1** clear the session and logout the user

**3. Logout Flow**

* 1. **eMerge Side**

**3.1.1** When user clicks on Logout button, call the logout api call and send the sessionId with the request

* 1. **eMergeAPI Side**

**3.2.1** Delete the record from table “USR\_AuthMaster” which matches with the sessionId, UserId and Token

**3.2.2** Clear the sessionId and Token from the cache memory

**3.2.3** Return true

* 1. **eMerge Side**

**3.3.1** After successful API call, clear the session and redirect user to the login page.

**4. Regeneration of Token in API**

* 1. **eMergeAPI side**

**4.1.1** We will keep the regenerate token time (half the timeout of session) in appsettings.json.

**a)** Let’s say the session timeout is 30 mins, then the token timeout will be slightly greater than session timeout let’s say 35 min.

**b)** Since regenerate token time will be half the session timeout, therefore regenerate token time will be 15 mins.

**4.1.2** When any api call is made after login, we will check the IssuedAt property of the JWT token to know the time the token was created.

**4.1.3** If regenerate token time <= ( DateTime.Now - IssuedAt ) < token timeout then create a new jwt token.

**4.1.4** After creating a jwt token, update the “USR\_AuthMaster” table with this token and timestamp against the sessionId and userId

**4.1.5** Also update this token in cache against the sessionId.

**4.1.6** Return the token with the responseDTO

* 1. **eMerge Side**

**4.2.1** Update the new token in the session.

**5. When Login from Other Page**

Will write this flow later.

**Known Issue with this flow –**

If the user logins and due to some reasons the tab is closed or browser is closed, then the record associated with this sessionId will be there in the table “USR\_AuthMaster” which will keep on adding as for every login new session is created. This will be more problematic in scenario’s where multiple session’s are allowed.

As of now when we regenerate the JWT Token and save it in table “USR\_AuthMaster”, we are also checking and delete those records with timestamp difference greater than that of session timestamp.

**We need to find a better solution for this.**