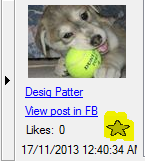
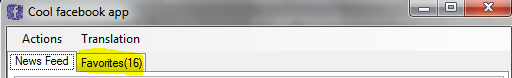
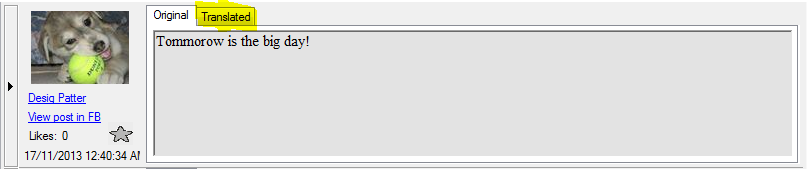
**Exercise 2**

Vainshtein Alexander, 312717218  
Shmuglin Daniel, 305870636

**Implemented features**

1. We implemented “Favorites” feature. Our application allows marking posts in user’s friends news feed as favorites. Those posts are displayed in additional tab in our application’s view. User can remove specific post from favorites later. Favorites post information stored using some data storage provider. We have implemented XmlFileStorage provider but the infrastructure of our application allows adding other implementations and easily switch to use them (for example, cloud storage provider).  
   In order to mark/unmark a post as a favorite user can use context menu (right-click around post area) or click on favorite’s sign – star (image) – that located on the left side of the post message. This sign has two states - yellow (favorite post) and gray (not favorite post). Clicking on the sign inverts post’s status (as favorite):  
     
     
   In order to see the favorite posts user should switch to the “Favorites” tab (which title indicated the current number of favorite posts):  
     
     
   Sources related to this feature are located under two project’s folders:
   1. Favorites
      1. FavoritesManager.cs
      2. FavoriteItem.cs
   2. Storage
      1. DataStorageFactory.cs
      2. IDataStorage.cs
      3. XmlFileStorage.cs
      4. IStorableItem.cs
      5. SimpleStorableItem.cs
2. Second feature that we’ve implemented allows the user instant translating of the specific post to desired language using one of implemented translators. In this exercise we allowed selection of one of three languages – Hebrew, English, and Russian – but it could be extend to dynamic list supported by translators.  
   We have implemented real translator by integration with the Microsoft’s Bing translation service (<http://api.microsofttranslator.com/V1/soap.svc>). We are using free account (that is limited by amount of translated symbols to 1M).  
   For the demonstration purposes we implemented two additional translators
   1. **Dummy** translator (just adds ***[translated]*** prefix to the original message)
   2. **Base64** translator – represents the text of the original message as Base64 encoded string

Both translators and supported languages options are loaded dynamically to the applications menu (***Translation***) based on defined enums. It makes adding and using new translator (e.g. google translate) matter of implementation and update of the appropriate enum.  
  
In order to use this feature user should simply click on the “Translated” tab in the post area:  


User can select desired language and translator in the application menu.

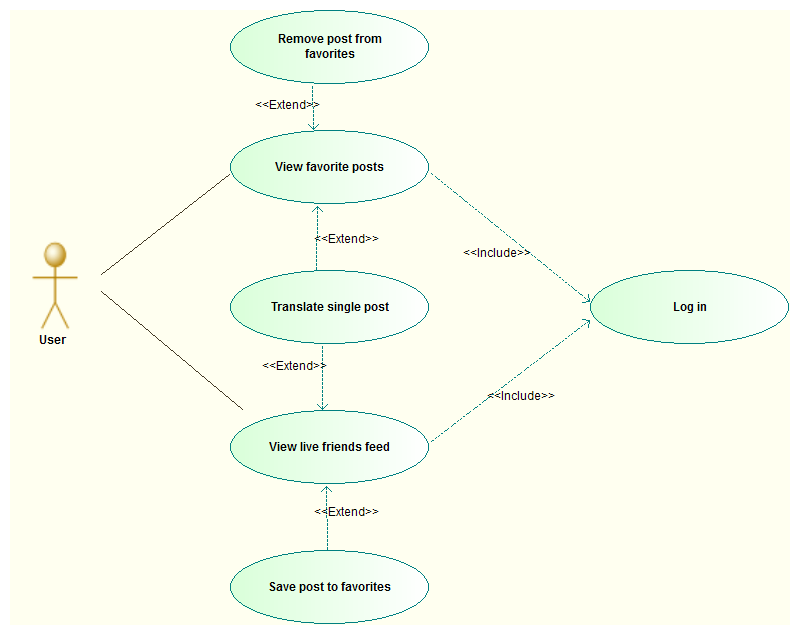
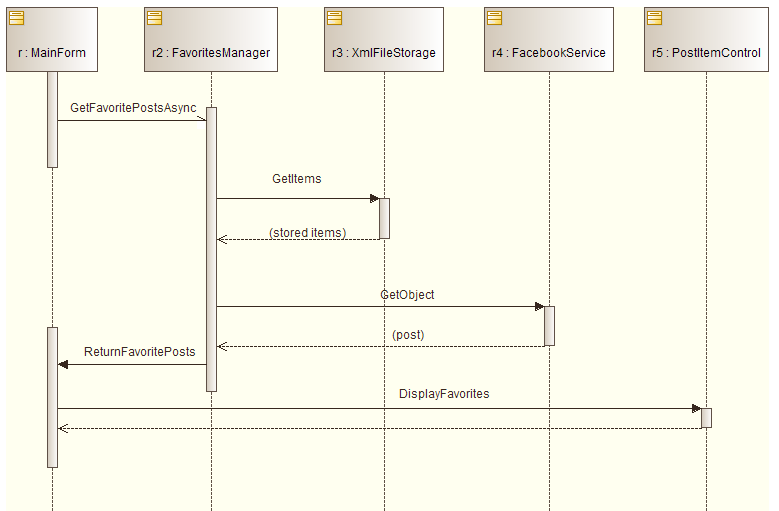
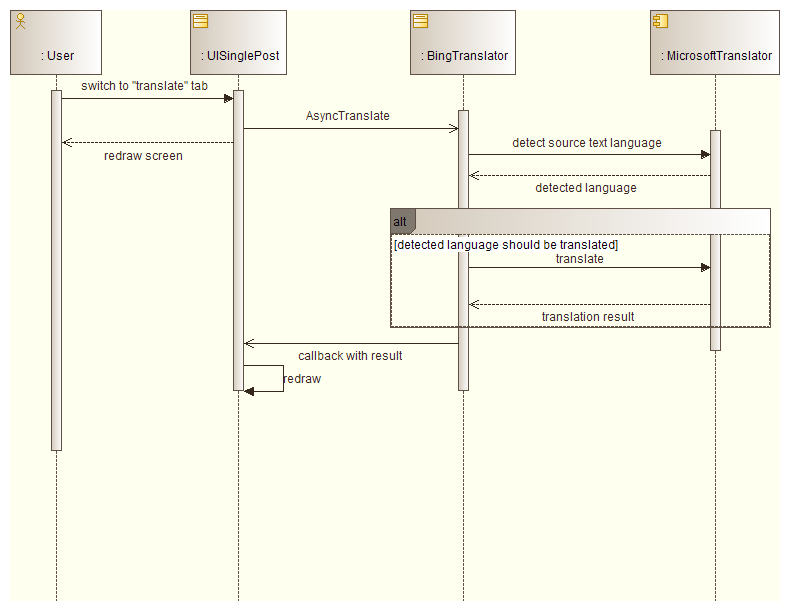
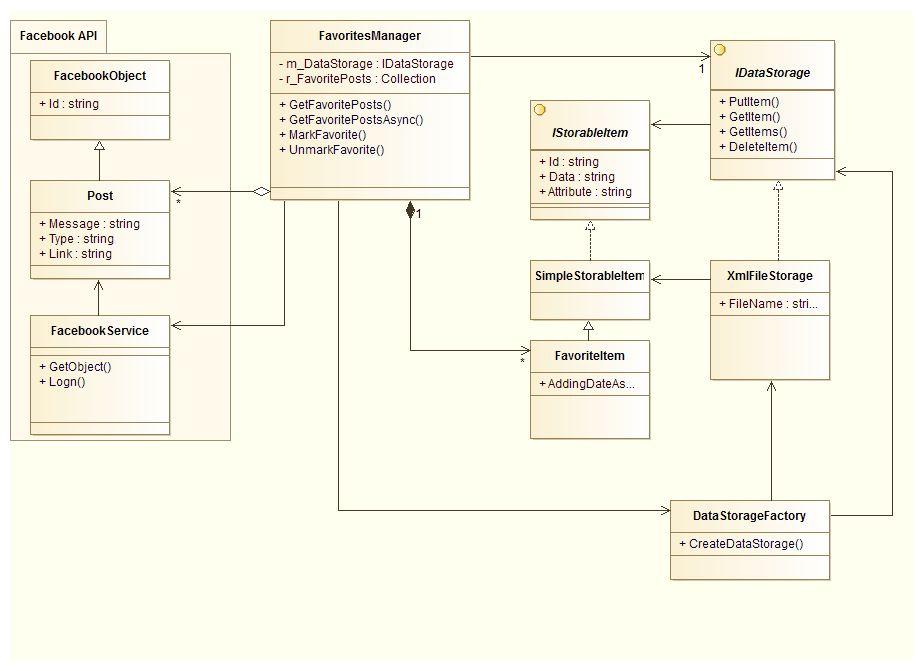
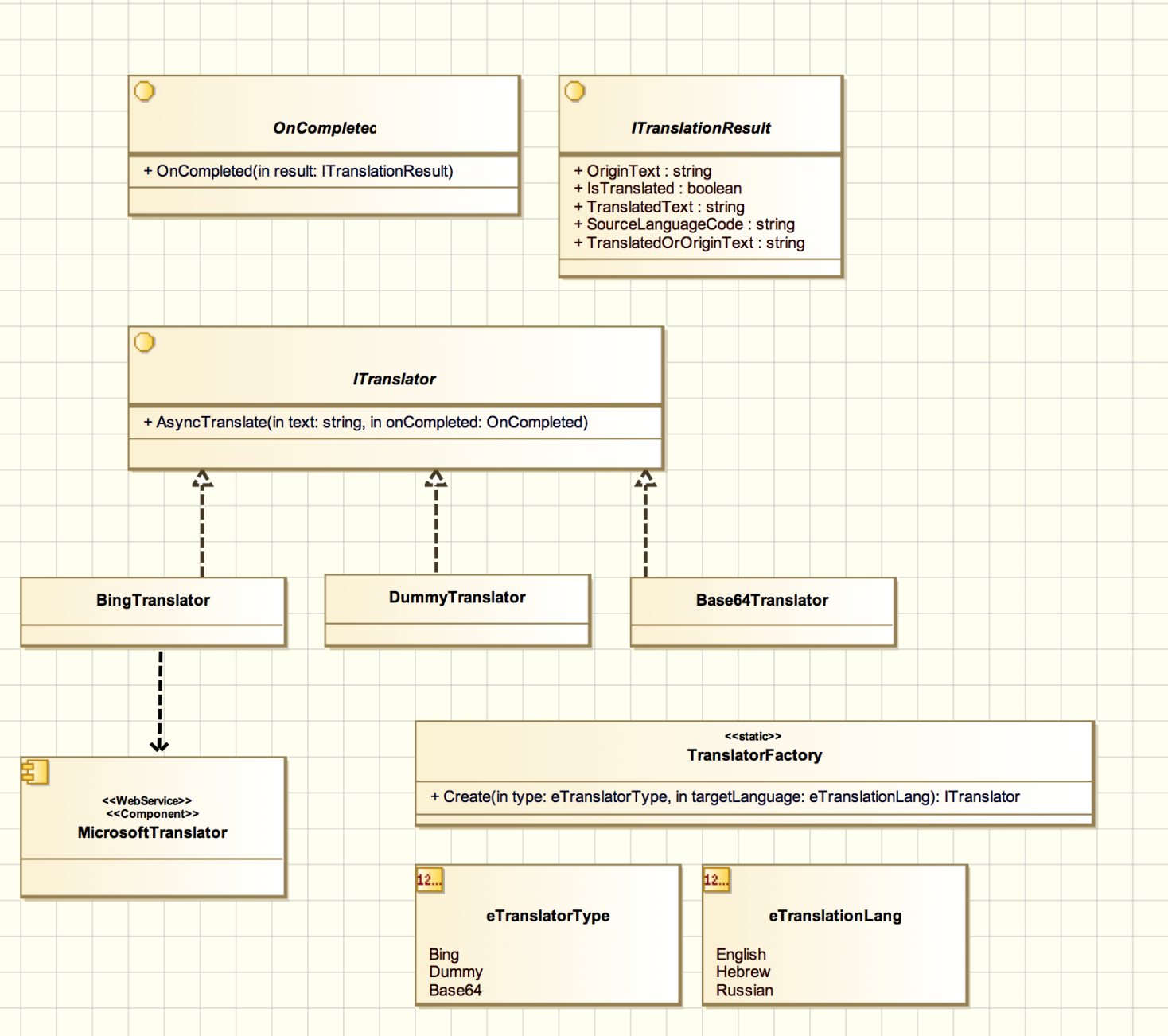
Note that once post is translated, the translation result is cached for the performance sake. Any future changes of the active translator/language will not affect translated posts (until post is reloaded). Post is “lazy” translated, i.e. it will be translated only when user clicks on the “Translated” tab.

Sources related to this feature are located under project’s folder **Translator**:

1. TranslatorFactory.cs
2. ITranslator.cs
3. ITranslationResult.cs
4. ITranslatorHost.cs
5. Bing translator
   1. BingTranslator.cs
   2. AdmAccessToken.cs
   3. AdmAuthentication.cs
6. Base64 translator
   1. Base64Translator.cs
7. Dummy translator
   1. DummyTranslator.cs

NOTE. In our project we used controls from the **Microsoft.VisualBasic.PowerPacks.Vs.dll** assembly which is not part of standard .NET Framework but comes with Visual Studio installation. We don’t submit that dll but on other machine it could be in different location than referenced in the project.

**Diagrams**

1. Use Case Diagram  
   
2. Favorites Sequence Diagram (View favorites posts scenario)  
   
3. Translation Sequence Diagram (Translate scenario)  
   
4. Favorites Class Diagram  
   
5. Translation Class Diagram   
   
6. UI Controls Class Diagram

