IT-485 Hands-On

In my hands-on, I am going to integrate some features to my application, which allow the user to see mobile specific pages when the application is opened using mobile web browser. The reason behind this is that nowadays in this busy world, everyone uses there mobile to do some kind of work and it will help to user to complete his or her tasks on the go. The application will provide an option to see the desktop specific pages on the mobile web browser when requested. The benefit of providing mobile specific pages for the users is to show the application according to the screen size of the device so that the user interface will give the feel to the user that they are using native mobile apps. The functionalities provided for application while using it from a mobile device is much similar to the ones that are available for the desktop users. There are two more use cases that are integrated with the application to increase its functionalities. The first use case is to allow the user for two-factor authentication, which will provide extra security for his or her account. Another use case is to provide URL routing to the application. The benefit of using this is to short the URL and also to remove the .aspx extension from the URL, which the user is requesting. The description for the processes and implementation for the hands-on are as follows:

Firstly, for creating the mobile specific mobile pages, I have created separate mobile pages, which are different from the desktop pages and are responsive. These pages are totally different from the desktop pages as the stylesheets and designing used for these pages are totally different. The pages are shown to the users according to the screen size of the device. So if the user opens the application from the mobile phone or from the IPad then it will render the page on their screen according to the size of the screen. For implementing this feature I have integrated 51.degrees library with my application. The 51.degree library consists of the latest database regarding the mobile devices and allows detecting the device that has requested the page. The request sent from the user is detected in the session_start method available in the Global.aspx and it renders the page according to the description of the device it gets from it. Now, suppose the user has requested from an IPhone. The application will identify that the request is from an iPhone and will provide the user with compatible page. However, if the user wish to see the desktop version of the application then he or she just need to click on the view desktop site option which is available in the footer and he will be provided with desktop specific page.

Secondly, for implementing URL routing, I have used friendlyurl library, which allows doing easier URL routing and switching between mobile and desktop pages. For implementing the features provided by this library I have to call the methods in the Application_Start method available in my Global.aspx so that whenever the user request for accessing the application, it will start the URL routing. The benefit of using this functionality is to shorten the URL (same feature provided by Bitly) as well as to show only limited information in the URL to the user so that hacking the application will become impossible.

Thirdly, the another feature provide to the user is two-factor authentication. The user has the authority to enable as well as disable this feature. An option is provide to the user to enable it and the way through which he or she wants to get the two-authentication code. It can be through email or through message. If the user has disabled this functionality then the while doing login he or she just have to provide the username and password. However, if the user has enabled this feature then while doing login, after providing correct credentials, he or she will be asked to provide a 5-digit unique number which is generated every time they try to login and sent to them. Until and unless users provide the correct two-factor code, they will not be allowed to inter into their profile.

Finally, while doing this hands-on, I faced some problems like testing the mobile pages, which had become so difficult while using android emulator. So, I used opera mini emulator, which allows seeing the application in mobile view like we are using a mobile browser. The problem with using opera mini emulator is that it was not rendering the design for the pages as it was supposed to be. So, I installed an extension for the Google chrome “Chrome User-Agent Spoofer”. Basically the extension allows using the desktop browser as a mobile browser. It provides to select from different device options like android, apple etc. So, when the request is sent to the application using this extension, the application detects the requesting device and renders the page according to it. It also allows retaining the design of the application, which was not possible with the opera mini emulator. The reason for not using android emulator is that it takes too much time to generate the emulator and also to process the request.