

# **CLOUD ASSIGNMENT- 6X-AZURE APPLICATION**

## **GROUP 16**

Udayin Biswas, 2014CS50296

Ashley Jain, 2014CS50280

Kartar Singh, 2014CS50286

Bipul Kumar, 2014CS50282

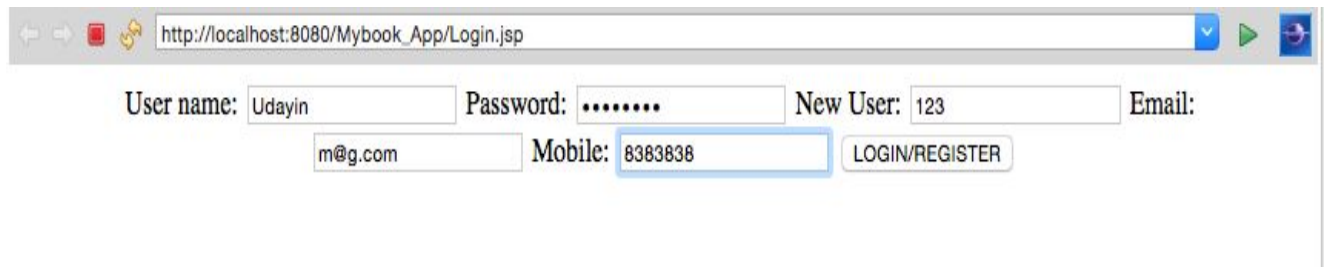
MyBook App feature is implemented using the Cloud features of Microsoft Azure Cloud.

We used Azure Cloud for the following:

Usage of storage account for the NoSQL database: Storing pictures, videos as blobs and other data in tables in the Storage account.

### **Application design:**

#### **(1)Login page**

A screenshot of a web browser displaying the login page of the MyBook App. The browser's address bar shows the URL 'http://localhost:8080/Mybook\_App/Login.jsp'. The login form contains several input fields: 'User name:' with the value 'Udayin', 'Password:' with masked characters '.....', 'New User:' with the value '123', and 'Email:' with the value 'm@g.com'. There is also a 'Mobile:' field with the value '8383838'. A 'LOGIN/REGISTER' button is located to the right of the mobile number field. The browser window has standard navigation buttons (back, forward, stop, reload) and a search icon in the top right corner.

First a user has to login/register in the Login page(Login.jsp).

If he enters x in New User:

(1)x=yes: He registers and the email id and mobile are recorded in PersonallInfoTable corresponding to user name. A new container is also created to store personal objects like images, videos etc in future.

(2)x=no : If userid and password are correct, then redirected to user's profile page.

#### **(2)Profile page**

localhost:8080/Mybook\_App/profile.jsp

Hello kartar  
ka@m.com  
939329

User name:  FOLLOW

PEOPLE I FOLLOW

MY STUFF

FRIEND'S STUFF

Add path:  Object name:  ADD OBJECT

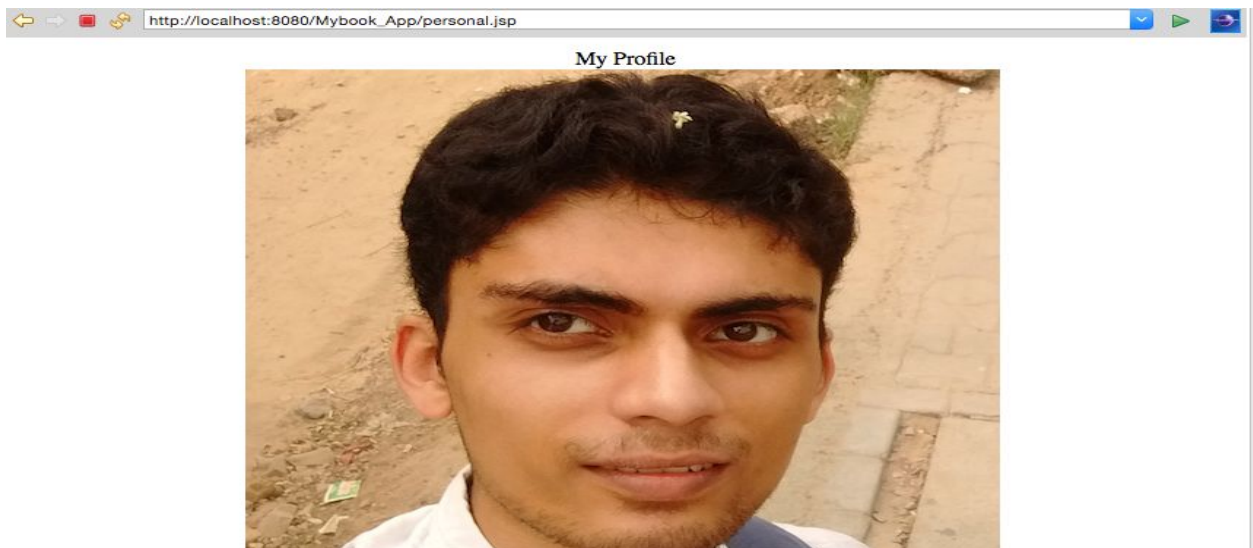
Add message:  ADD message

In profile page, personal info entered during registration displayed at the top. There is an option to follow a person by entering his userid. There are buttons like People I follow, My Stuff (Pictures, Videos, messages), Friends's stuff (Pictures, Videos, messages) which will redirect to different pages. We can upload a picture, video by adding its path and giving it a name. Also we can add messages.

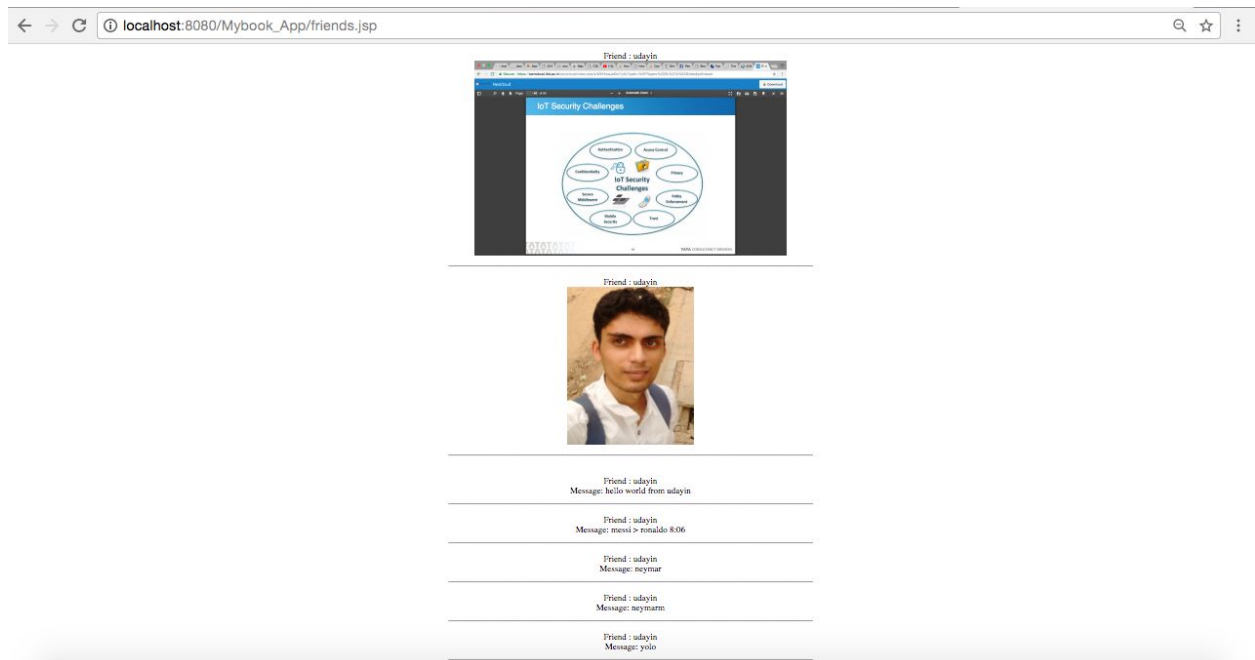
http://localhost:8080/Mybook\_App/followers.jsp

hello  
ashley, udayin,

This is the simple page which shows the people the user is following.



This is the personal.jsp page which is accessed by 'MyStuff' button. Shows all the personal content uploaded by mentioning path and which are stored as blobs in the user's container.



This is the page which will show all the content of the people whom a user is following. The content will be displayed below the person's name.

## **Implementation details**

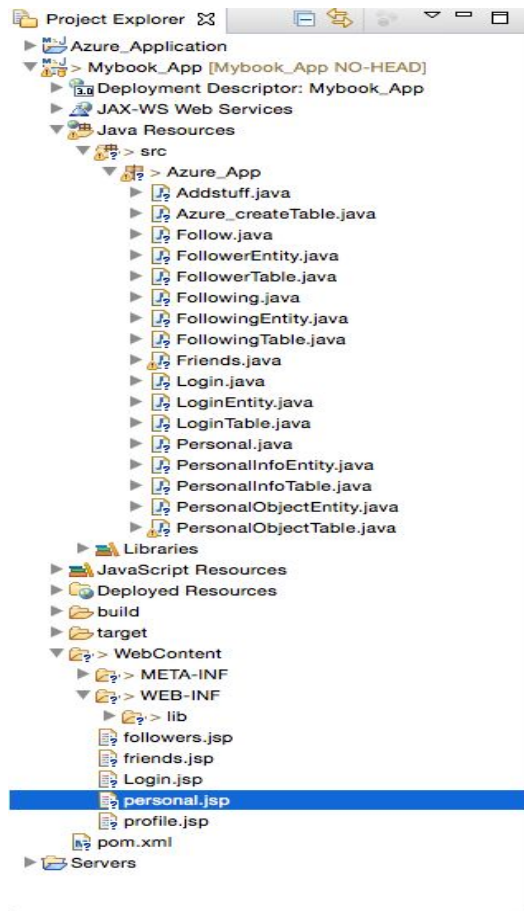
We made the Azure App in Eclipse IDE using Java and JSP. Java files were created to define the methods for interacting with the Storage account- inserting data/blobs or fetching them. JSP files were used to design the web pages of the Application.

Java servlets helped to transfer control from one jsp page to another and interact with the backend i.e. interacting with the storage account.

### **Links used:**

<https://docs.microsoft.com/en-us/azure/cosmos-db/table-storage-how-to-use-java>

<https://docs.microsoft.com/en-us/azure/storage/blobs/storage-java-how-to-use-blob-storage>



The above is the directory structure for the myBook App. The application was run using Apache Tomcat 8.0 server and the starting page is the Login.jsp page.

Web and other dependencies were added in pom.xml file and servlet mappings were done in web.xml file.

The Entity pages define the basic structure of the entity in the respective tables.

Then the Table java files will implement methods used in the above links to access the Storage account tables and containers.

The jsp pages are designed and connected via servlet java files which make a decision when there is a post action in jsp pages. These servlets use the Table java file's methods directly.

Start page is [http://localhost:8080/Mybook\\_App/Login.jsp](http://localhost:8080/Mybook_App/Login.jsp) after running Tomcat server.

---