# How to Build Guide for Net Banking Web Application

# (NetBankingApp)

- Munjal Thakkar
- munjal@umd.edu

**Introduction**

- This app was developed on Windows 10 Operating System.
- Developed using Microsoft Visual Studio 2022.
- The language it has been written in is .Net (5.0)
- For Database I have used Microsoft SQL Server Management Studio

**How to Build and Run**

- This app can run in the VM provided by the professor as well.
- You can download this app from here
- Or you can also clone the app using git-clone.
- Once you've opened the app in Visual Studio, delete the 'Migrations' folder and all its contents.
- Open the Package Manager Console.
- In that, type following commands
    - **add-migration Testing** (wait for the build to be succeeded. You can see the Migrations folder will be regenerated).
    - **update-database** (This will create the Database in the SQL Server Management Studio)
- Mock data is stored in Data->SeedData.cs file.

**Logging**

- Logging has been implemented by overriding the ILogger functionality in .Net itself.
- The classes mentioned below help in Logging
    - FileLogger.cs
    - FileLoggerExtensions.cs
    - FileLoggerProvider.cs
- The logs are stored in a folder called "logs" which is present inside the NetBankingApp folder.
- Daily logs are stored.

**Cryptography**

- Hashed Passwords (with randomly generated salts) are stored in the Database.
- The entire hashing logic is implemented in Services->PKDF2Hashing.cs

**Login**

- The app has 3 users with accounts, transactions, etc. all created.
- The mock data is stored in Data->SeedData.cs
- There are 3 users, their credentials are:
    - First User:
        - Userid: 12345678
    - Second User:
        - Userid: 17963428
    - Third User:
        - Userid: 38074569
- The password for all the three users is: **test@123**
- You can see in the Login table that even though the password for all the users is same, the hashes for them are different. This is because of the randomly generated salts.

**Registering as a User**

- Registering as a user is possible, you can click on register and register as a user.
- But once you've registered, the userID you get is of only 4 digits, but to login, you'll need an ID with 8 digits.
- This is because, since this is a Net Banking Application, once you've registered, the admin will have to go into the database and set the userID to an 8 digit one manually.
- This has been done for security purposes.

**Other**

- ATM is from where you can Withdraw, Deposit or Transfer money.
- Deposit is hypothetical (in case Cash transaction at bank physical site).
- After making a transfer of funds in accounts, we can see the history of the transaction under the transaction tab.
- I think the rest of the functionalities are pretty self-explanatory.

**Session Management**

- Session Management has been implemented throughout the website.
- Attributes->AuthoriseCustomerAttributes.cs is also a file being utilized for session management.

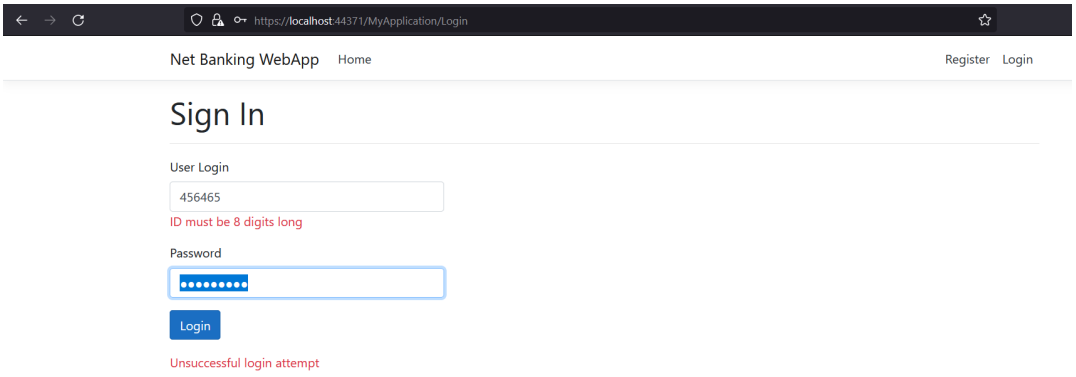
NOTE: In all below screenshots, Billpay has been included in the header of the all the test cases screenshot, but in reality, I have removed that functionality at the last moment, since it was super buggy.

# Test Cases

I tested the web application with the test cases as shown below:

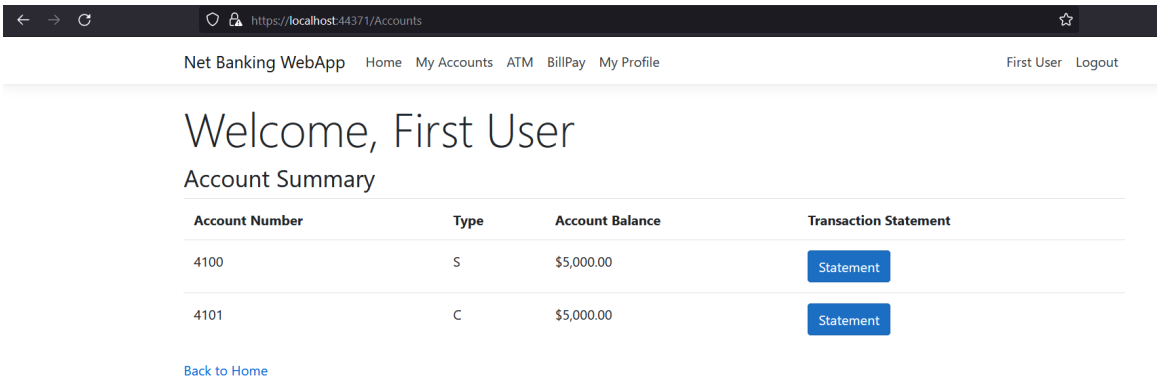
**Testcase**: Providing wrong input credentials.

**Expectation**: Not allowed to login and given appropriate errors.



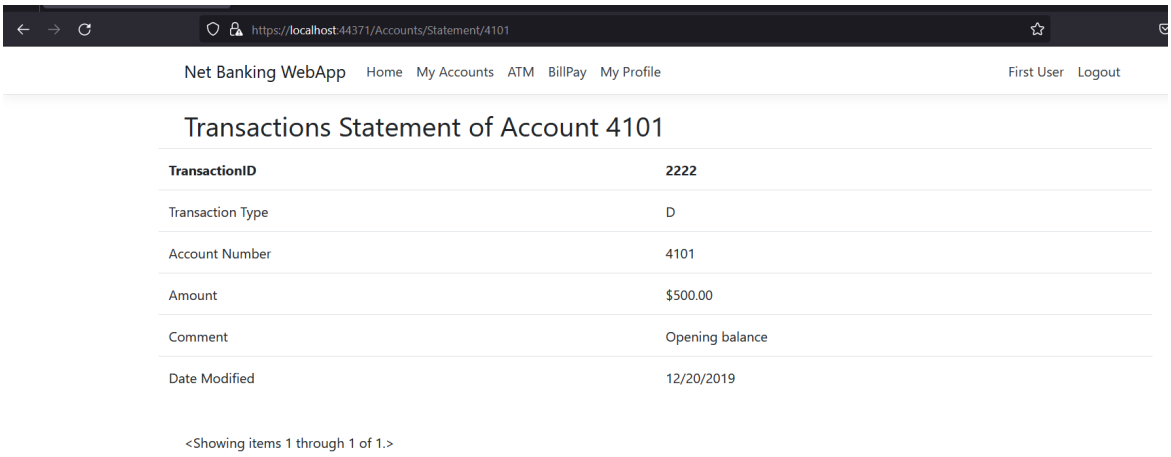

**Testcase**: Providing correct input credentials.

**Expectations**: User gets logged in and is shown the accounts page which by default is set as the entry page for any new user logging in.



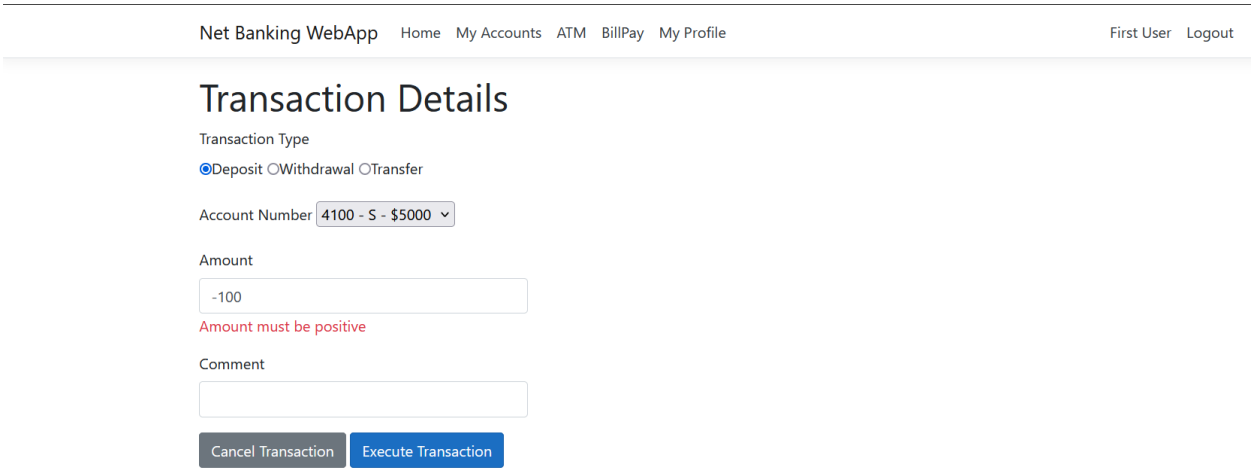
**Testcase**: Clicking Statement for a particular account.

**Expectation**: Shows the hypothetical statement built by me by default. (If you make any transactions or anything, then they will be shown here as well. Done in the later test cases.)



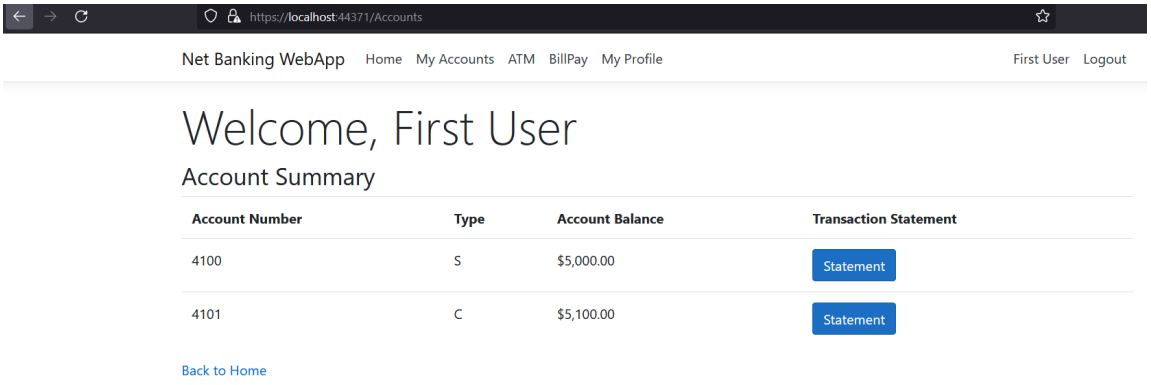

**Testcase**: Going into the ATM tab and trying to deposit or withdraw or transfer a negative amount.

**Expectation**: Amount should be positive error.



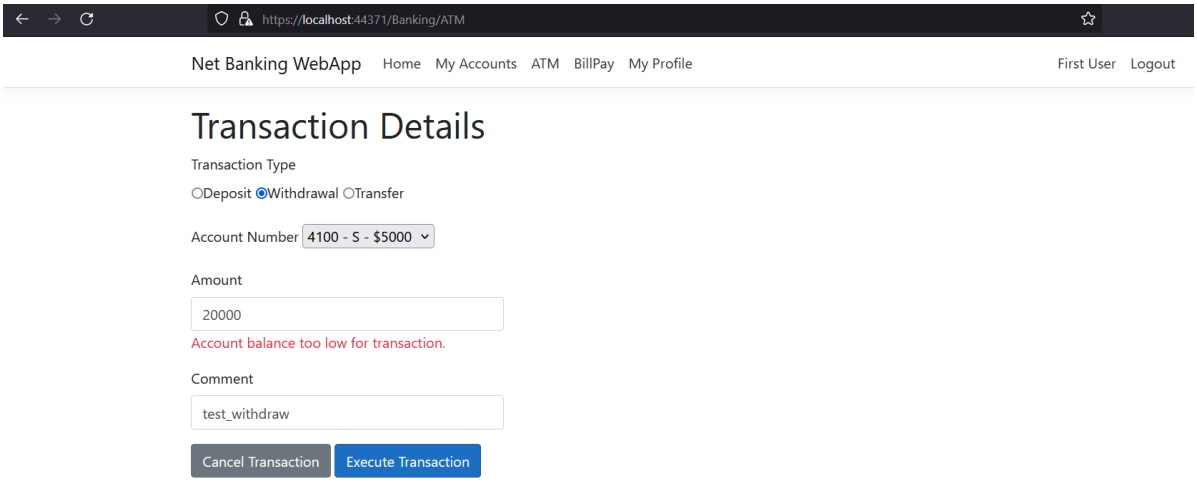
**Testcase**: Depositing 100$ into the account (similar to as shown in the above screenshot)

**Expectation**: The transaction gets reflected in my accounts.



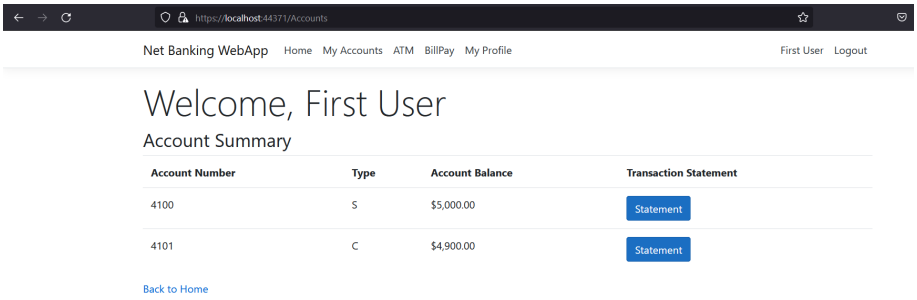

**Testcase**: Trying to withdraw amount more than the original amount present in our account.

**Expectation**: Not being allowed to do so.



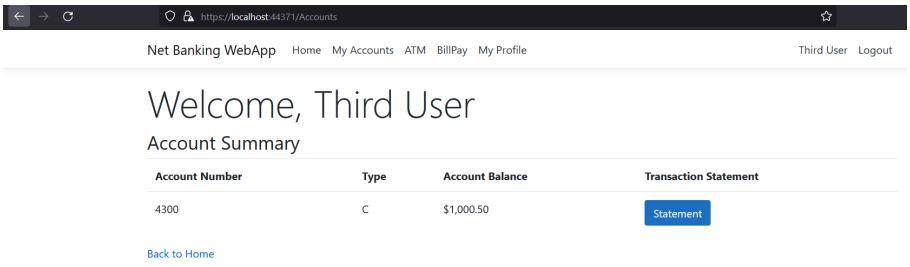
**Testcase**: Withdrawing 200$ from the Checking account.

**Expectation**: Being allowed to do so, and the change being reflected in the accounts

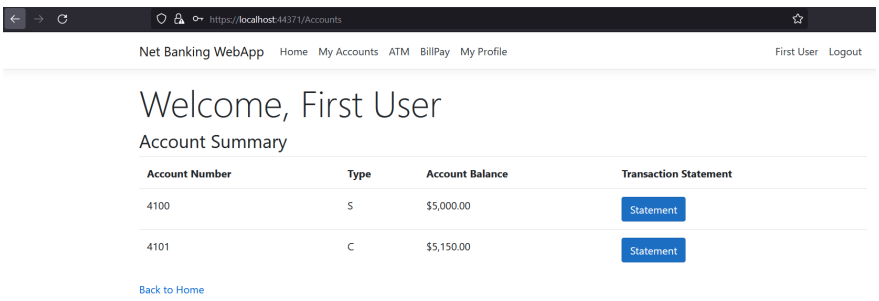



**Testcase**: Logged in as Third User and transferring 250$ to First User's checking account.

**Expectation**: Changes are reflected in both the accounts.



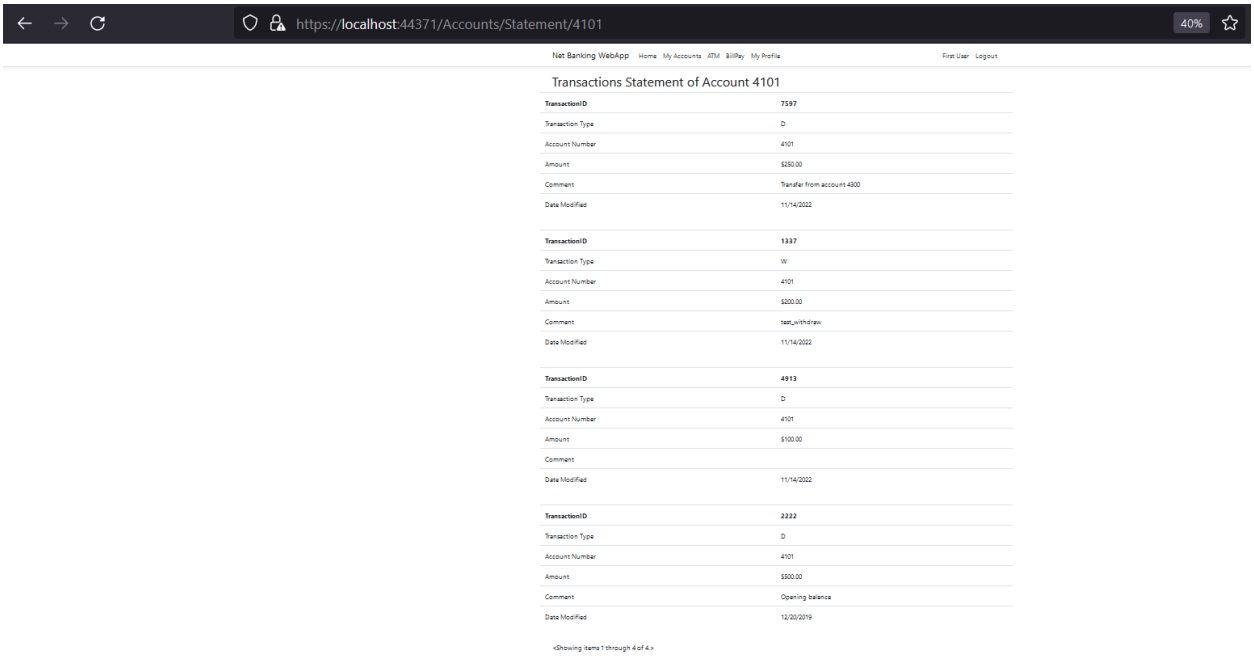

Third User's Account (He had 1250.5$ before the transfer)




First User's account

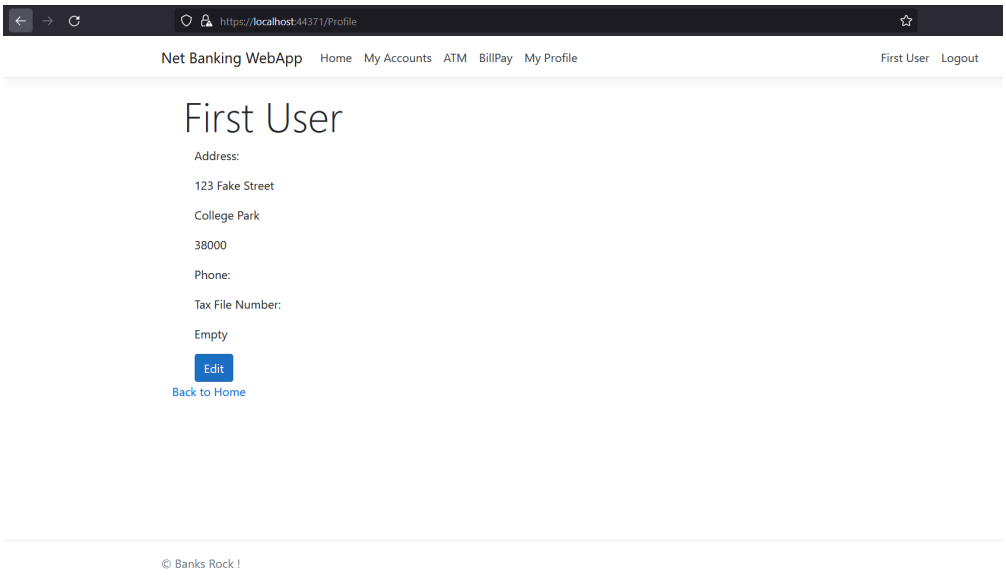
**Testcase**: After doing all these transactions, while taking a look at First User's statement.

**Expectation**: Being shown a description of all of the above transactions we have performed.



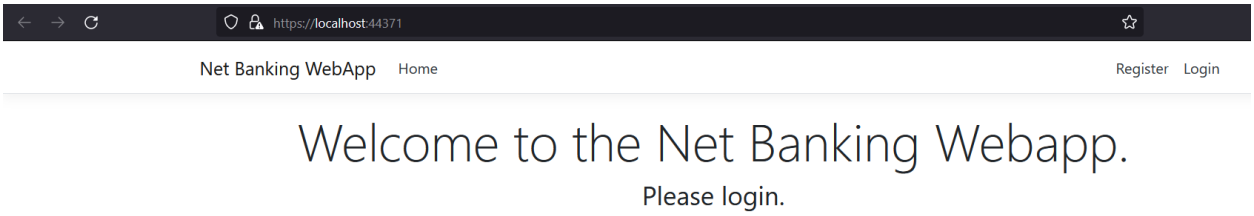

**Testcase**: Clicking on My Profile

**Expectation**: Getting to view all the profile information.



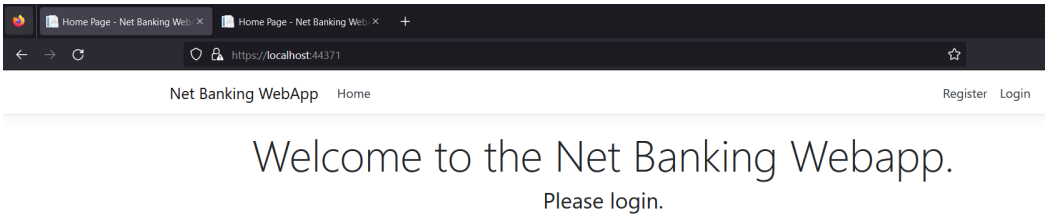
**Testcase**: Clicking on Logout

**Expectation**: Successfully Logging Out.




**Testcase**: Staying logged in, and logging out in a different tab.

**Expectation**: Not being able to perform any functions in either of the tabs.




**THE END**