**Functional testing (testing the app e.g. buttons)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test no. | Test Scenario | Example test case | Pre-condition | Test steps | Example test data | Expected outcome | Actual Results | Pass/Fail |
| 1 | Test if error message displays when an incorrect username or password is inputted within login screen | Test case 1: invalid username and valid password  Test case 2:  valid username and invalid password  Test case 3: invalid username and password | Requires an Internet connection that doesn’t have any restrictions (e.g. home network) | 1. go to login screen  2. input data within username and password fields  3. press log in button  4. see if an error message is displayed | Test case 1: Username: “jzard420” password: “hello123”  Test case 2: Username: martin007 password: “given456”  Test case 3: Username: “nelloih43” password: “seven321” | should display an error message when user inputs a wrongs username or/and password | Does not display anything | Fail |
| 2 | Within create account, test if error message displays when user inputs a username that has already been taken | Test case 1:  Input a username that is already taken | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must go to the create account screen | 1. go to the create account screen  2. input an already taken username  3. see if it displays an error message | Test case 1:  Username:  “jmartinez” | Should display an error message when user inputs an already taken username | An error message displays | Pass |
| 3 | Within create account, test if error message displays when user inputs an email address in the wrong format (correct format – Text@emailProvider.com) | Test case 1:  Input an email address in the wrong format | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must go to the create account screen | 1. go to the create account screen  2. input an email address in the wrong format  3. see if it displays an error message | Test case 1:  Email Address:  “unknown@gt” | Should display an error message when user inputs the email address in the example test data | An error message displays | Pass |
| 4 | Within create account, test if the button works. It should display all the information user inputted within a newly created profile (e.g. first name: Martin, last name: Stein) | Test case 1:  Input all fields in create account and see if it displays in profile screen | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must go to the create account screen | 1. go to the create account screen  2. input all fields correctly  3. click create account button  4. see if profile displays correct information about user | Test case 1:  Title: Mr  First name: John  Last name: Smith  Username: jsmith007  Age: 40  Email: [jsmith007@gmail.com](mailto:jsmith007@gmail.com) | Should display a user profile with the fields mentioned in the test example data | Displays all the fields mentioned in the test example date within a newly create user profile | Pass |
| 5 | Within the profile page, test if the camera button opens the AR camera | Test case 1:  Click the camera button | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must have created an account  3. requires you to log in to your account and enter your profile page | 1. log in or create a new account  2. enter your profile page  3. click the camera button  4. see if it opens the AR camera | No data tested | Should display the AR camera after clicking the camera button | Opens the AR camera | Pass |
| 6 | Within the edit profile page, test whether the update profile button can update the user profile | Test case 1:  Old information on profile screen about the user  Test case 2:  Updated information on profile screen about the user | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must have created an account  3. requires you to log in to your account, enter your profile page and then access the edit profile screen | 1. log in or create a new account  2. enter your profile page  3. click the edit profile button  4. see if profile updates after you have entered all the fields correctly and have clicked the update profile button | Test case 1:  Title: Mr  First name: John  Last name: Smith  Username: jsmith007  Age: 40  Email: [jsmith007@gmail.com](mailto:jsmith007@gmail.com)  Test case 2:  Title: Mr  First name: Johnson  Last name: Smith  Username: js9056  Age: 42  Email:  [JS9056@gmail.com](mailto:JS9056@gmail.com) | Should display a user profile with the fields mentioned in test case 2 example data | Displays the correct updated fields within the profile | Pass |
| 7 | Test basic usability of the application. i.e. whether it is possible to navigate to different screens | Test case 1:  Login screen – navigates to profile page and create account screen  Test case 2: create account - navigates to login screen and profile page  Test case 3: profile page - navigates to login screen, edit profile screen and the AR camera  Test case 4:  Edit profile page – navigates to the profile page | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must have created an account | 1. enter each of the mentioned screens mentioned in the test case. See if it navigates correctly | No data tested | Should navigate to the screens specified in the test case column | Navigates to the correct screens | Pass |
| 8 | Add tests of AR Camera | … | …. | … | … | …. | …. | …. |

**Non-functional testing (how app operates. Not how it functions): not all tests will need SCREENSHOTS**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test no. | Test type | Test scenario | Pre-condition | Test steps | Expected outcome | Actual Results | Pass/Fail |
| 1 | Database Security testing | Test if mongo database displays hashed version of password under the password and confirm password field | Requires an Internet connection that doesn’t have any restrictions (e.g. home network) | 1. visit mlab.com and login to account  2. open the mongo database used to store user accounts  3. see if passwords have been securely hashed | All accounts should have a hashed password of length 29 | All accounts successfully have hashed passwords of length 29 | Pass |
| 2 | Usability testing | Test if all screens in the application are user-friendly | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must have created an account | 1. open application  2. visit every screen (e.g. from the login screen to the create account screen to the profile screen etc.) | All usable features in each screen should be easy to use, stress-free and be simple for users to understand what usable features do what | After testing the application, we realised some of our screens were not user-friendly. For example, the create account screen had way too many input fields requiring user input. Another example would be the AR camera. There were buttons present in the screen that didn’t have clear icons to represent what function they carry out | Fail |
| 3 | Database recovery testing | Test if there is a recovery plan if database gets corrupted or erased | Requires an Internet connection that doesn’t have any restrictions (e.g. home network) | 1. visit mlab.com and login to account  2. open the mongo database  3. go under backups  4. see if there is a mongodump or a way to create one | Their should be a recovery (mongodump) of the database containing all user accounts | There is no recovery (mongodump) present. Having a recovery requires payment with MLab. Paying for stuff is something we tried avoiding in the project | Fail |
| 4 | Compatibility testing | Test if AR camera has no compatibility issues when using on any mobile | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must have created an account | 1. login to your account in the app  2. click the camera button in your profile page  3. see if it opens the AR camera without any rendering issues | The AR camera should open up | Most mobile we tested the app with seemed to open the AR camera, but their were issues with certain mobiles such as Samsung Galaxy S7 Edge. The camera screen would not either open or it would display a load error message | Fail |
| 5 | Database scalability testing | Test if our Mongo database allows us to create at least 100,000 user accounts before exceeding 500 MB (we would need to pay £15 or over to add more storage. Under 500 MB is free to use) | Requires an Internet connection that doesn’t have any restrictions (e.g. home network) | 1. visit mlab.com and login to account  2. open the mongo database  3. see how much storage each account roughly takes  4. calculate how many accounts can be created that doesn’t exceed 500 MB | The database should be able to hold over 100,000 user accounts | The database can hold roughly 149,700 user accounts. Each account on average takes 3.34 kilobytes. Multiply this with 149,700 and it should be equal to 499998 kilobytes, equivalent to 499.98 megabytes | Pass |
| 6 | Load testing | Test if app can handle multiple users at once | 1. Requires an Internet connection that doesn’t have any restrictions (e.g. home network)  2. must have created an account | 1. get around 10 to 15 people to login to their account at the same time  2. see if there are any loading issues | The application should work normally. There shouldn’t be any technical issues | The application seems to be running smoothly when exactly 10 users logged in at the same time. The loading speed of the app accessing the user profile seemed normal. Loading speed for other screens also remained consistent | Pass |
| 7 | Add tests of AR Camera | … | …. | … | … | …. | …. | …. |

**Static testing** – reviewing code without executing program to find errors. E.g. check if syntax is correct, code is organized etc.

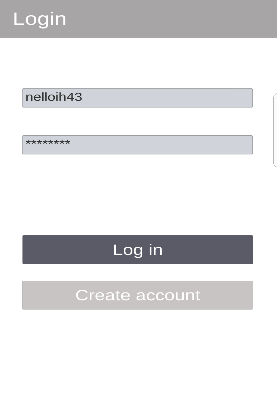
**Dynamic testing** – code is executed to find errors. Result of the code is checked/compared with expected output. For this we can do unit testing (testing parts of the program e.g. login screen, create account etc individually.) and then integration testing (joining all the small parts of the program together and see if it works.)

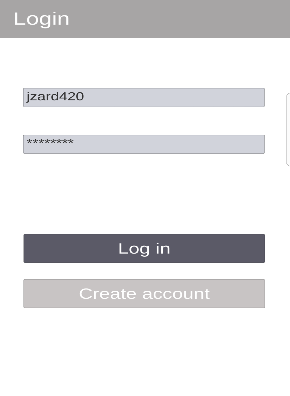
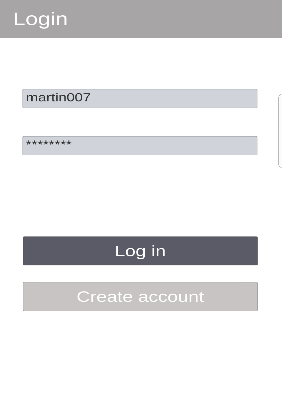
**Static testing and dynamic testing**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test no. | Screen/s | Test type | Defects to search for | Tools used | Test steps | Expected outcome | Actual Results | Pass/Fail |
| 1 | Create account | Static testing | 1. syntax violation  2. dead code  3. unused variables  4. variables with undefined value or values that are rendered useless  5. causes that might’ve made code too long or difficult to understand | Built-in Automated tools | 1. open all relevant C# scripts  2. review each script and find any of the defects mentioned | To find at least one or more defects within each of the relevant C# scripts | **Multiple defects found!**  1. dead code  2. unused variables  3. variable equal to null that’ll produce a NullPointerException. This variable only produces null, if user inputs a username in the username input field that does not exist in the database. This variable essentially query’s the database to find the inputted username. If it doesn’t find the username. It returns null. This is a defect I cannot change | pass |
| 2 | Login Screen | Static testing | 1. syntax violation  2. dead code  3. unused variables  4. variables with undefined value or values that are rendered useless  5. causes that might’ve made code too long or difficult to understand | Built-in Automated tools | 1. open all relevant C# scripts  2. review each script and find any of the defects mentioned | To find at least one or more defects within each of the relevant C# scripts | **Defect found!**  1. their is a variable that produces a NullPointerException. This variable only equals null, when user inputs a username or password that doesn’t exist in our database. This is a defect I can’t change | Pass |
| 3 | Profile Screens | Static testing | 1. syntax violation  2. dead code  3. unused variables  4. variables with undefined value or values that are rendered useless  5. causes that might’ve made code too long or difficult to understand | Built-in Automated tools | 1. open all relevant C# scripts  2. review each script and find any of the defects mentioned | To find at least one or more defects within each of the relevant C# scripts | No defects found! | Fail |
| 4 | AR camera | Static testing | ……………………………………. | …………………………………………. | ………………………………………… | ………………………………………….. | …………………………………………….. | …… |
| 5 | Create account | Dynamic testing – unit testing | 1. unexpected exceptions  2. program not being able to detect text being inputted in the input field each time  3. unexpected errors | Unity console | 1. run the create account scene in Unity  2. enter text in all input fields (e.g. first name, last name etc.)  3. see if inputted text prints & updates in console | To see if the inputted text within each of the input fields is recorded by program. This is something we can use to create our profile pages, hence the importance of this test  **Expected output:**  first name: Gina  last name: Begum  username: gina008  email address: gina123@gmail.com  password: hello123  confirm password: hello123 age: 30 | Expected output matches actual output! No defects found | Fail |
| 6 | Login screen | Dynamic testing – unit testing | 1. unexpected exceptions  2. The Mongo queries placed within C# Script not being able to find a valid username & password in the database  3. unexpected errors | Unity console | 1. run the login scene in Unity  2. first enter a valid username & password (you will need to create an account)  3. then enter an invalid username or password  4. see what the console prints | When user inputs a valid username and password, it should output the whole document extracted from MongoDB in String format  When user inputs a invalid username or password, it should output just NULL in console.  **Expected output for valid username and password:**  { "\_id" : ObjectId("5aa1cde2fee0042de8bd96d6"), "firstname" : "Ishtiyaq", "lastname" : "Ali", "username" : "ish2nv", "password" : "0aa7a662c728b7407c54ae6bfd27d1", "cpassword" : "0aa7a662c728b7407c54ae6bfd27d1", "day" : null, "month" : null, "year" : null, "age" : "44", "eaddress" : "hello@gmail.com", "phonenumber" : null, "title" : "Mr", "account" : "Decorator", "companyname" : "Topps Tiles", "jobtitle" : "Tiler" **}**  **Expected output for invalid username or password:**  Null | Expected outputs matches actual outputs! No defects found! | Fail |
| 7 | Profile Screens | Dynamic testing – unit testing | 1. unexpected exceptions  2. profile screen not displaying user details or displaying incorrect details about user  3. unexpected errors | Unity console | 1. run the login screen or create account screen first  2. navigate your way through the program to the profile screen  3. see what is printed on console and on the profile screen. Is it correct? | All fields within the profile screen should have correct information  **Expected output:**  Title: Mr  First name: Ishtiyaq  Last name: Ali  Username: ish2nv  Age: 30  Email: [ishtiyaq93@gmail.com](mailto:ishtiyaq93@gmail.com) | Expected outputs matches actual outputs! No defects found! | Fail |
| 8 | AR camera | Dynamic testing – unit testing | ……………………………………. | …………………………………………. | ………………………………………… | ………………………………………….. | …………………………………………….. | …… |
| 9 | 1. Create account  2. Login screen  3. Profile Screens  4. Edit profile screens  5. AR camera | Dynamic testing – integration testing | 1. unexpected exceptions  2. any unexpected errors | Unity console | 1. integrate all the scenes in one Unity file  2. see if any errors come about in the Unity console | All scenes should work when put together in one Unity file | All scenes work successfully together in a Unity file | Fail |
| 10 | Add tests of AR Camera | … | …. | … | … | …. | …. | …. |

**Appendix**

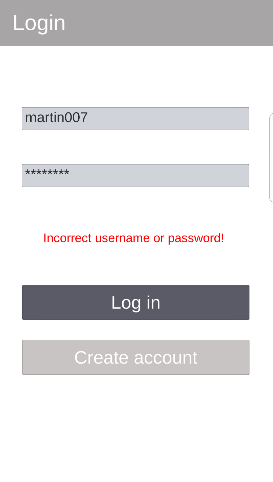
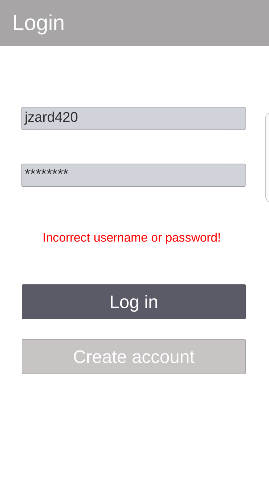
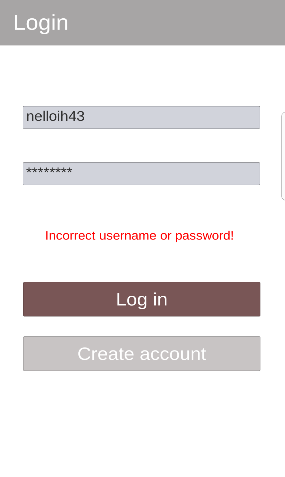
**Functional testing**

**Test 1**



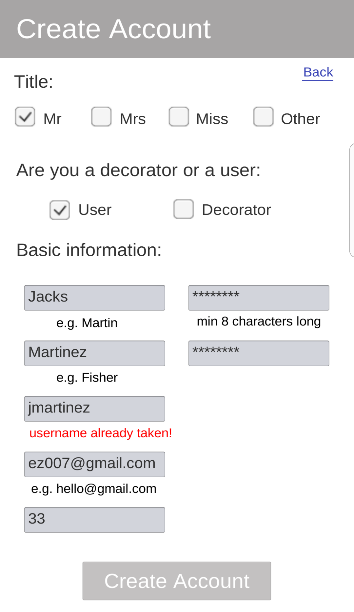
No error message displayed in any of the screenshots when entering incorrect username and/or password. this is a problem, since user will not know whether they are making a mistake or whether the app is not responding

**Changes to fix error**



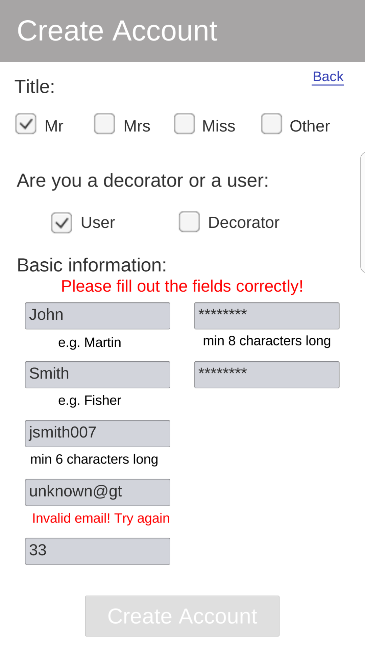
When the log in button is pressed it now displays an error message in red. The awareness from this functional test enabled me to notice this mistake and rightly fix it immediately

**Test 2**



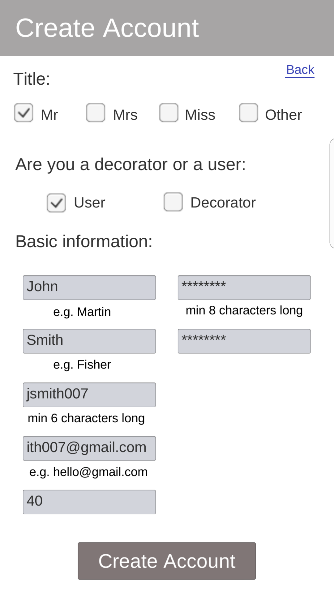
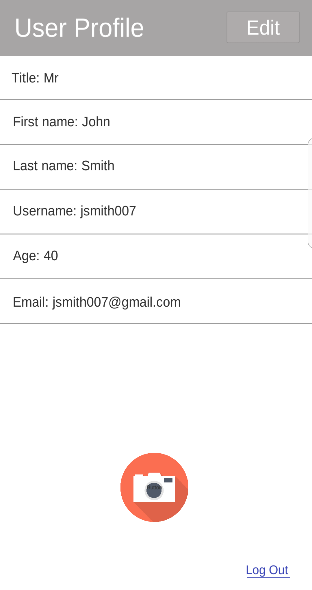
The example data “jmartinez” produces the following error message “username already taken.” This is proof that the error message pops-up when user inputs a username that has already been created

**Test 3**

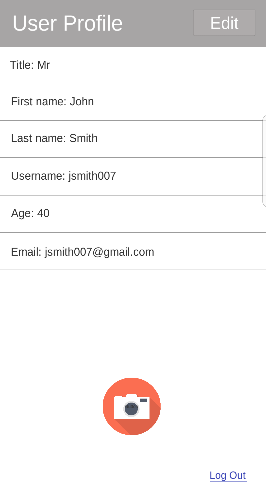


The example data “unknown@gt” produces the following error message “invalid email! Try again.” This is proof that the error message pops-up when user inputs an email address in the wrong format. This is a way of validating the reliability of the email addresses users input

**Test 4**

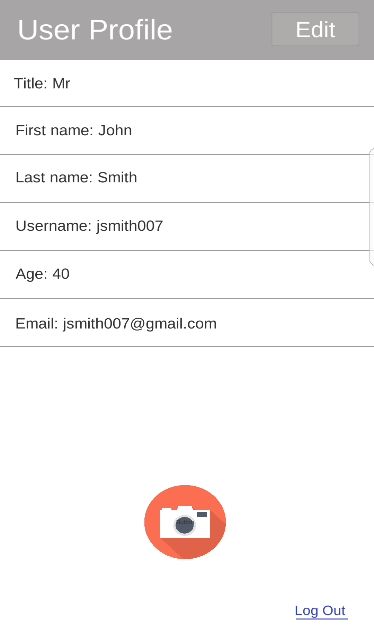
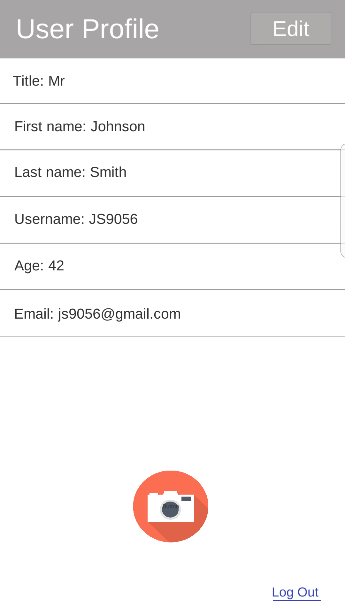


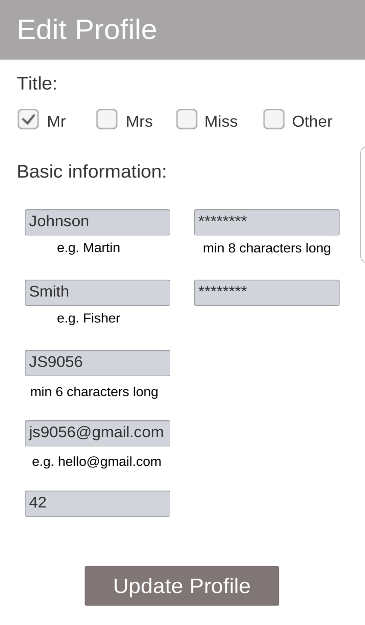
First, we create an account. Then enter our user profile. As you can see, the text inputted in the fields within the create account screen display within the user profile, meaning the user profile was able to extract information about the user correctly and display it

**Test 5**



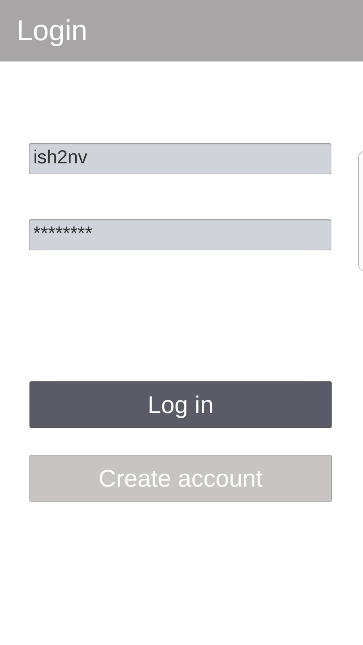
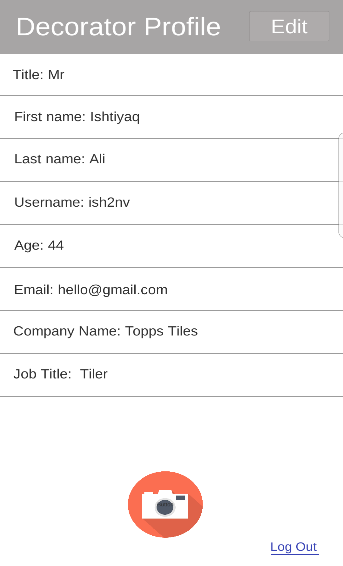
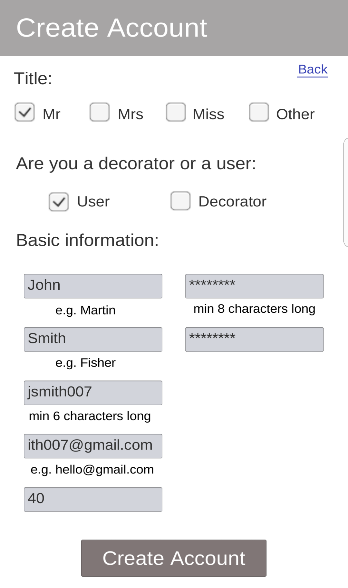
Camera button within the profile page can successfully load the AR camera and display it on the mobile screen. The load time speed is usually between 1 to 5 seconds, which is not bad considering how technical/advanced an AR camera is

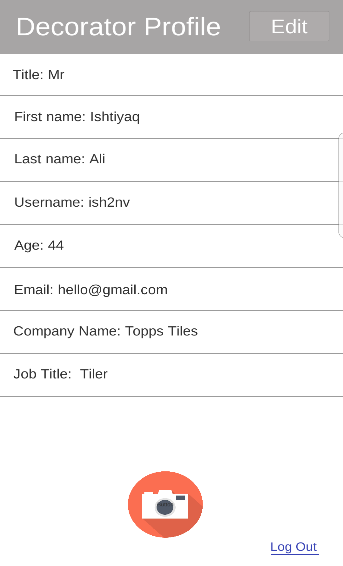
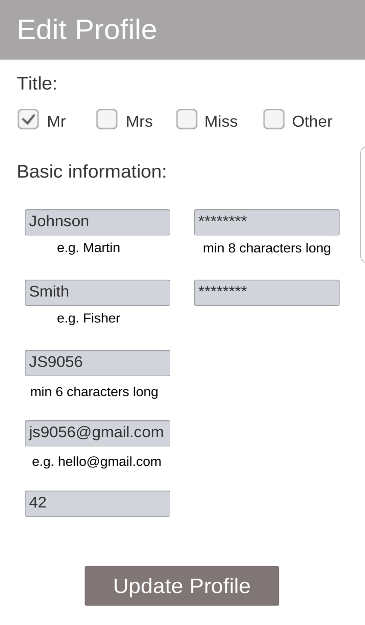
**Test 6**

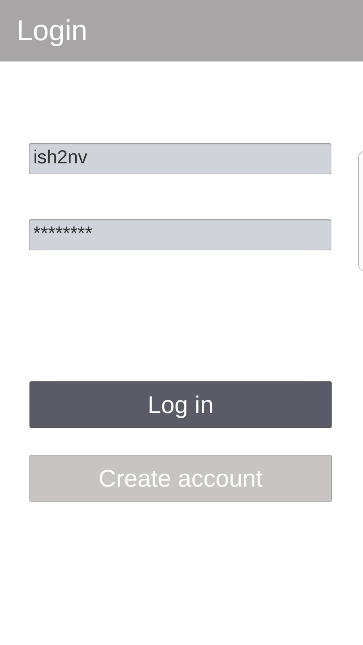


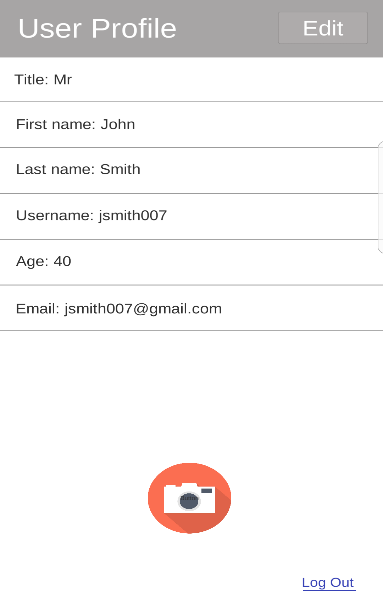
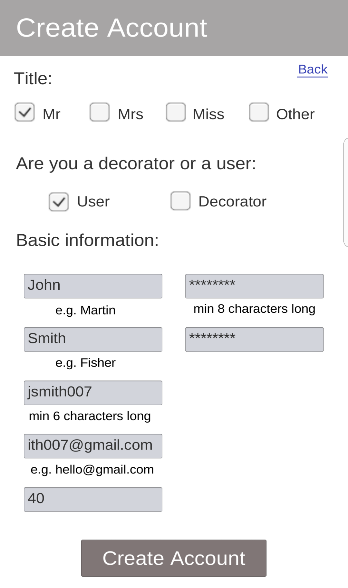
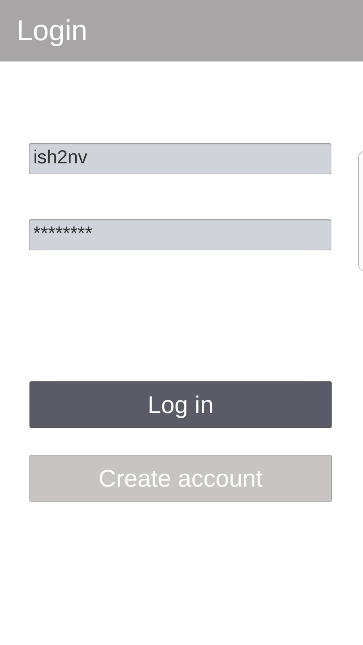
These series of screenshot are trying to illustrate how our application can update a user’s profile. In the first screenshot it shows fields with user information. Second screenshot shows the user going into the edit profile screen. This is where the user updates their profile. Last screenshot shows the user profile displaying the updated user information.

**Test 7**

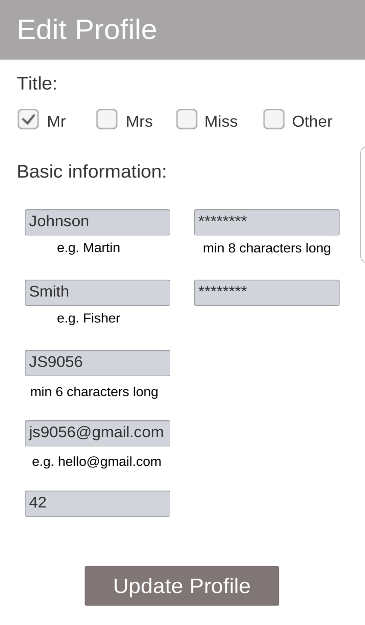
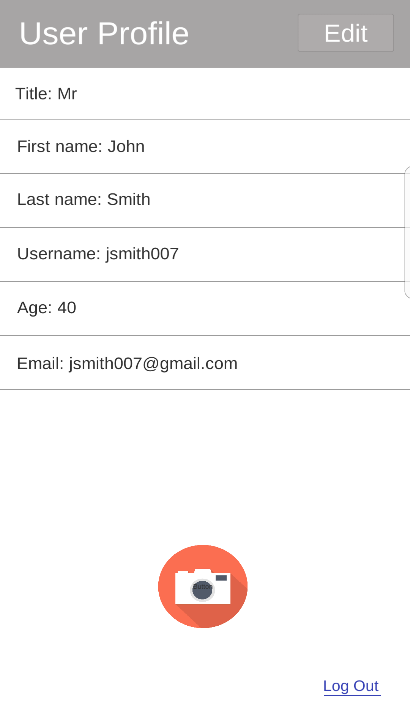
**Login screen – navigates to profile page & create account**

**Profile page – navigates to edit profile screen, AR camera & login screen**

****

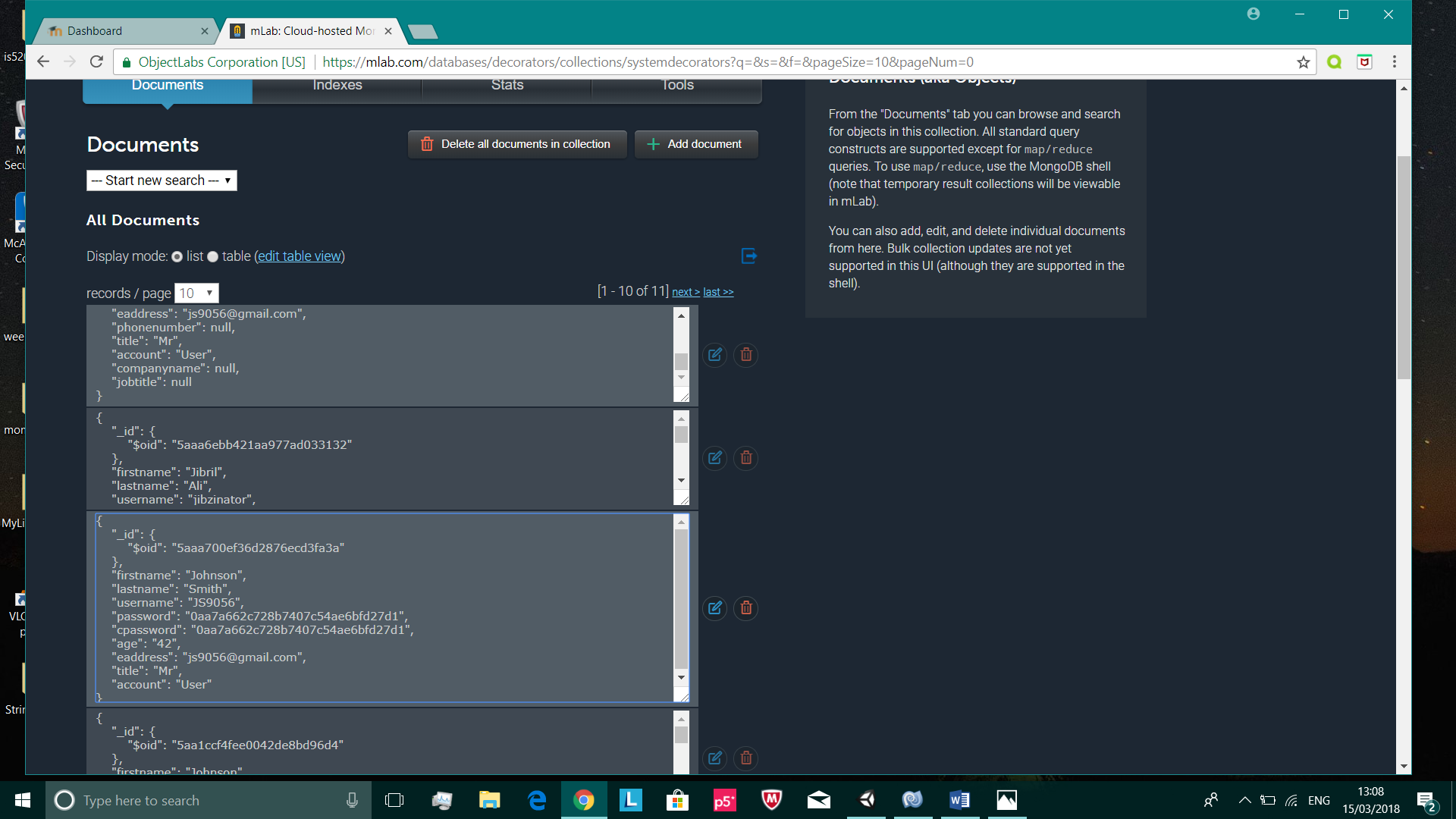
**Create account screen – navigates to the log in screen and the profile page**

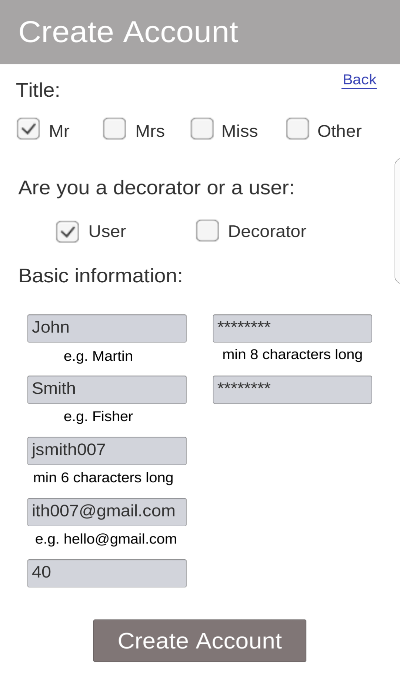
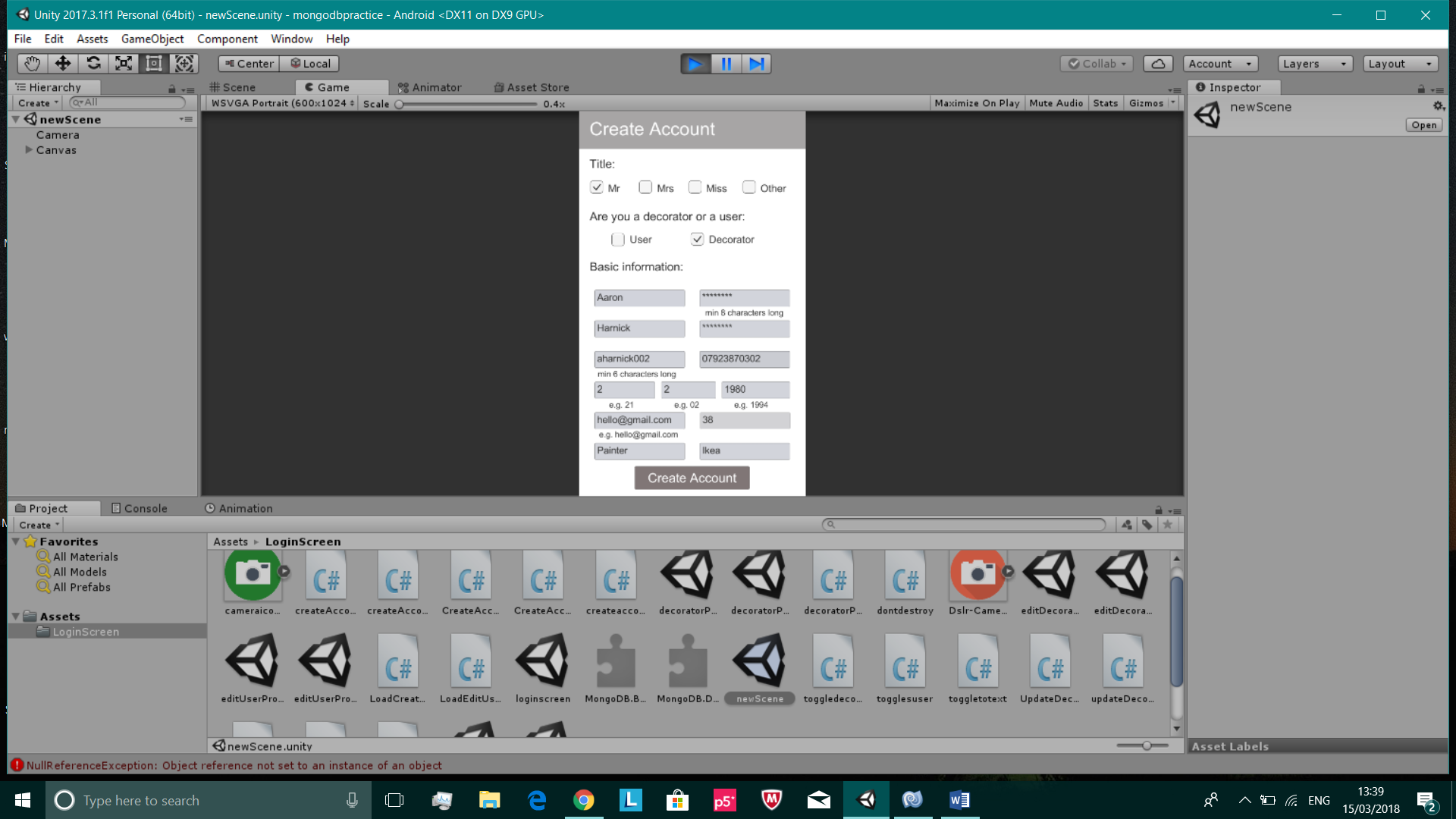
**Edit profile page – navigate back to the profile page (with updated user information)**



**Non-functional testing**

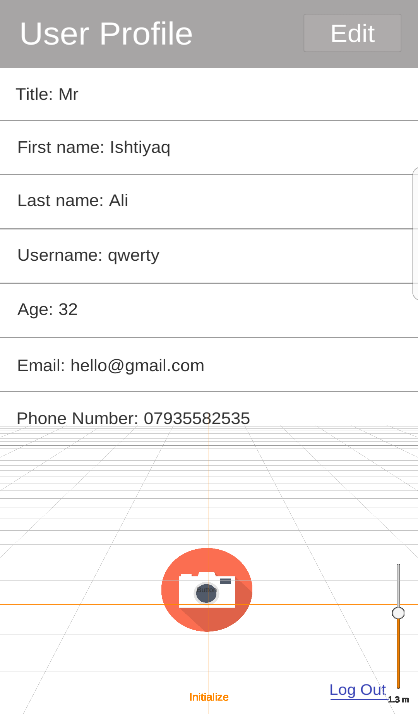
Within our Mongo database, accounts are represented as documents. Within this document we can see proof that the password and confirm password fields have indeed been hashed. This adds a layer of security to our database since hackers will not be able to decrypt the password nor will developers have access to it. Thus, securing users privacy

**Test 1**

**Test 2**

Due to how congested the create account looks with multiple input fields squashed together, we decided to reduce the number of input fields, so it is more user-friendly i.e. preventing users from inputting too much information

Changed the create account layout to..

**Test 4**

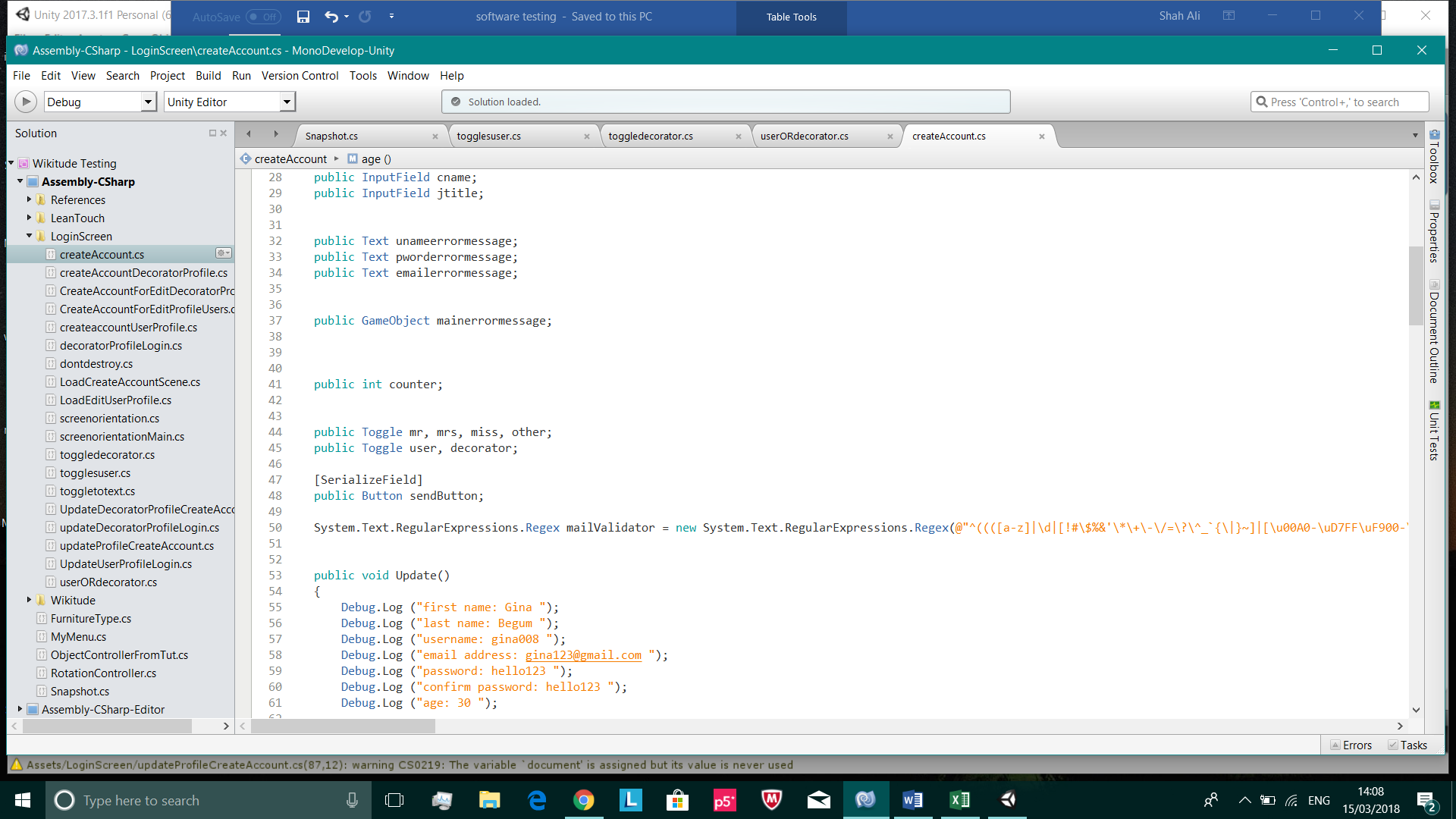
The AR camera in our application opens almost instantly. Loading speed is at a very good rate for most devices. But there are certain devices that have issues in opening the AR camera. Some may not open on the first or second try. And some do not open the AR camera at all. The screenshot on the left was made by a Samsung Galaxy S7 Edge. As you can see, the AR camera has not been rendered properly. This is an issue we did solve for Samsung phones, but there are other mobile devices that we know cannot render/load the AR camera properly.

**Test 6**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | User 1 | User 2 | User 3 | User 4 | User 5 | User 6 | User 7 | User 8 | User 9 | User 10 |
| Loading speed of user profile | 1.1 | 1 | 0.8 | 0.9 | 0.9 | 1.5 | 1.7 | 1.3 | 1.8 | 1.1 |
| Loading speed of AR camera | 1.9 | 2.3 | 2.1 | 2.4 | 2.7 | 2.1 | 2.3 | 2.5 | 3.4 | 3 |
| Loading speed of create account screen | 1 | 1.5 | 1.7 | 1.8 | 1.2 | 1.4 | 1.2 | 1.9 | 0.8 | 1.2 |
| Loading speed of edit profile screen | 1.6 | 2.3 | 2.5 | 2.9 | 2.6 | 2.3 | 2.5 | 2.1 | 2.2 | 2.3 |

The ten users we had, whom all logged in at the same time all recorded the time it took (in seconds) to load each screen within the application. From this table I was able to conclude that there were no issues in loading any of the screens. All the loading times seem reasonable.

**Static and dynamic testing**

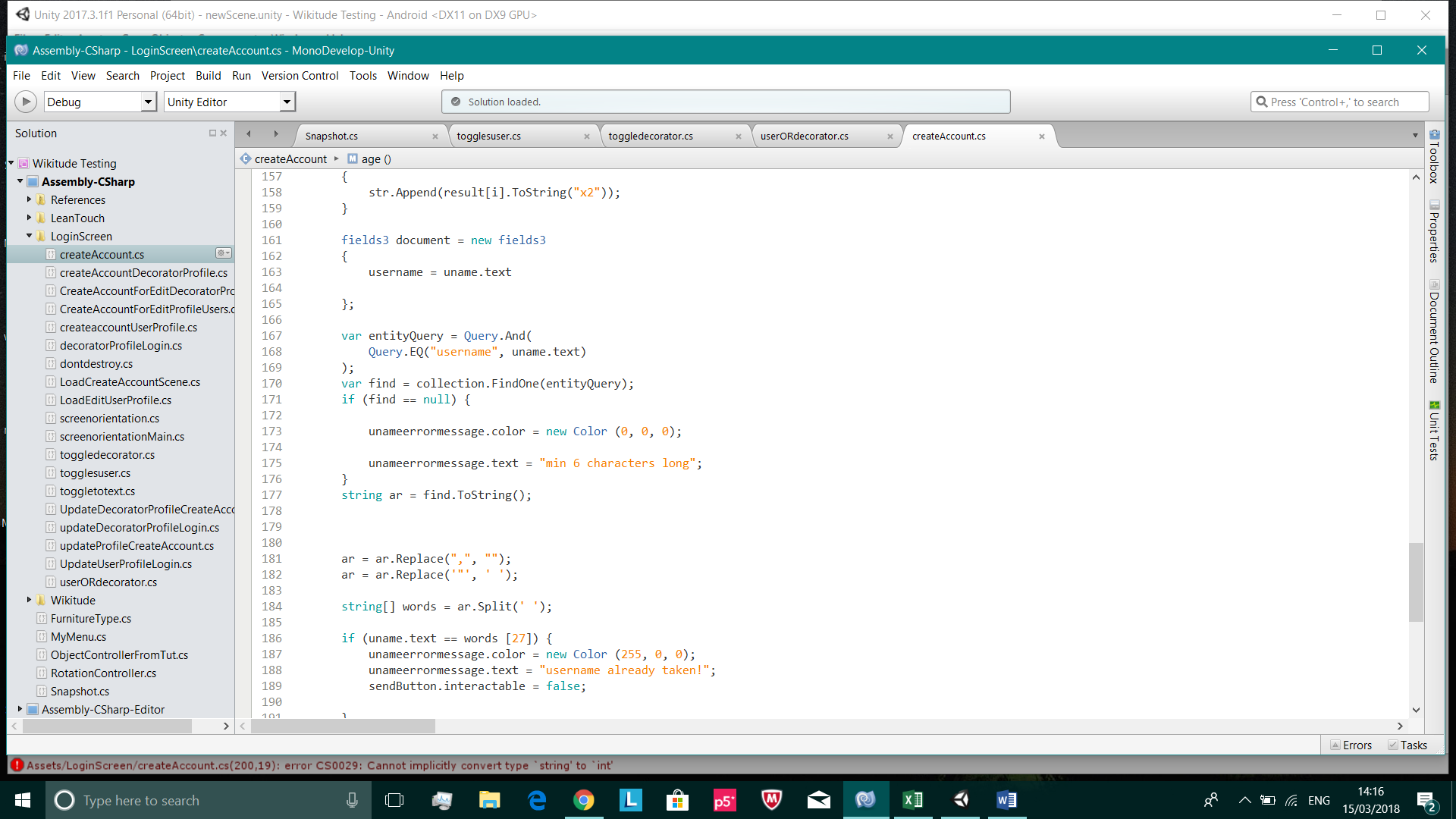
**Test 1**

/

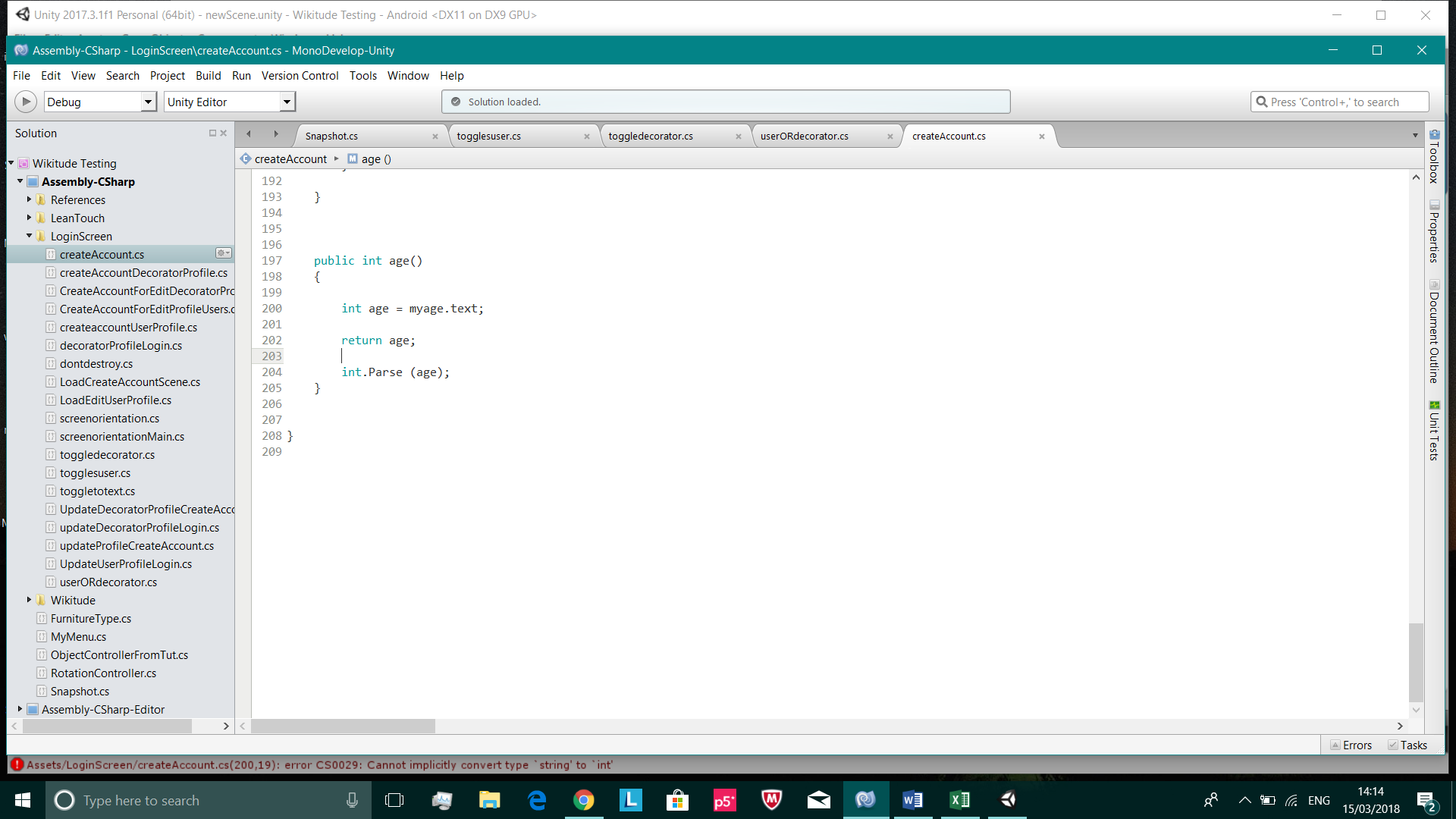
Variable removed

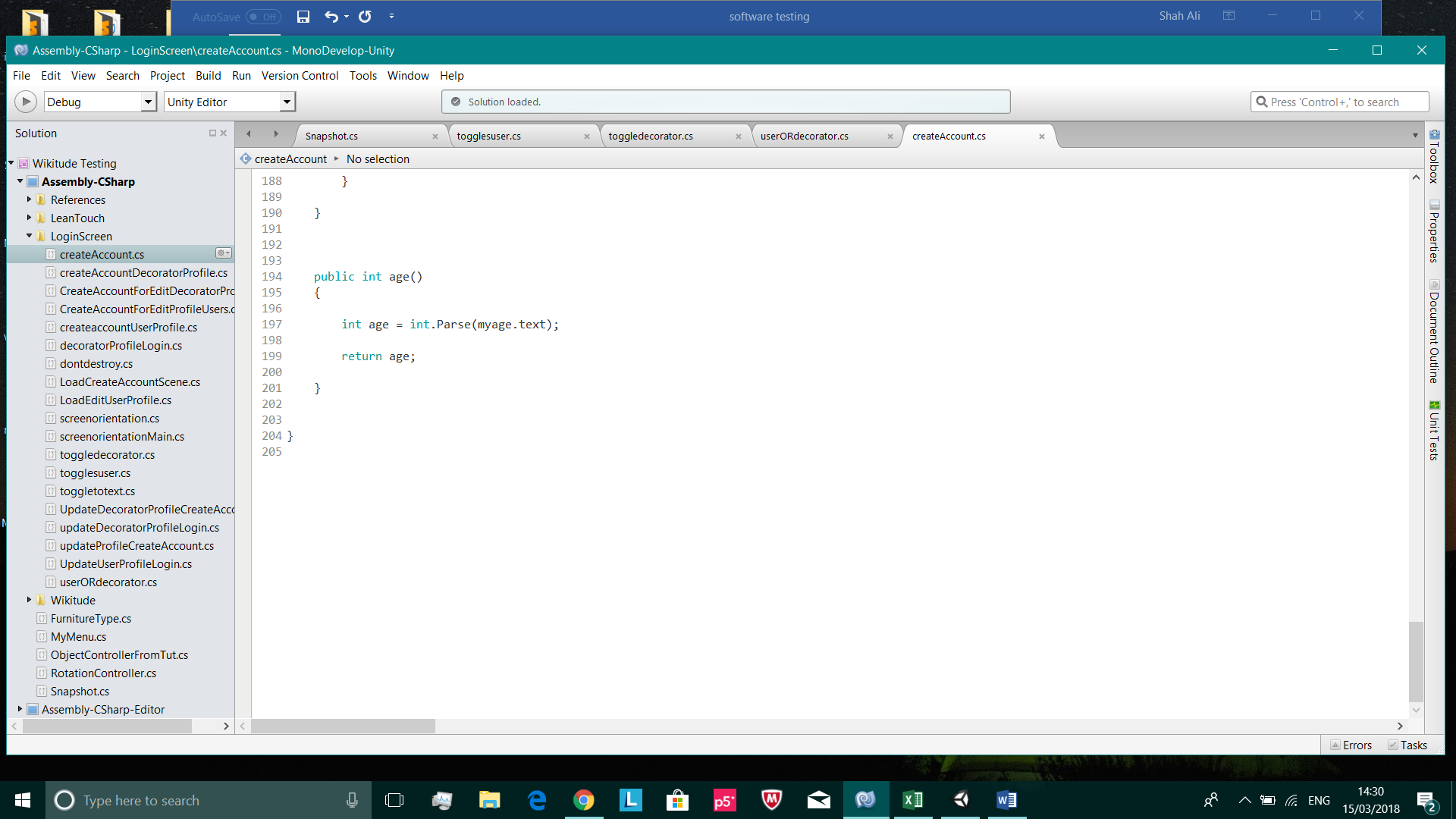
Unused/uninitialized variable

Returns null if it cannot query the inputted username. This generates a NullPointerException. This is a defect I cannot remove since I need this query to search for the inputted username in the database and there is no other way of carrying this out



/



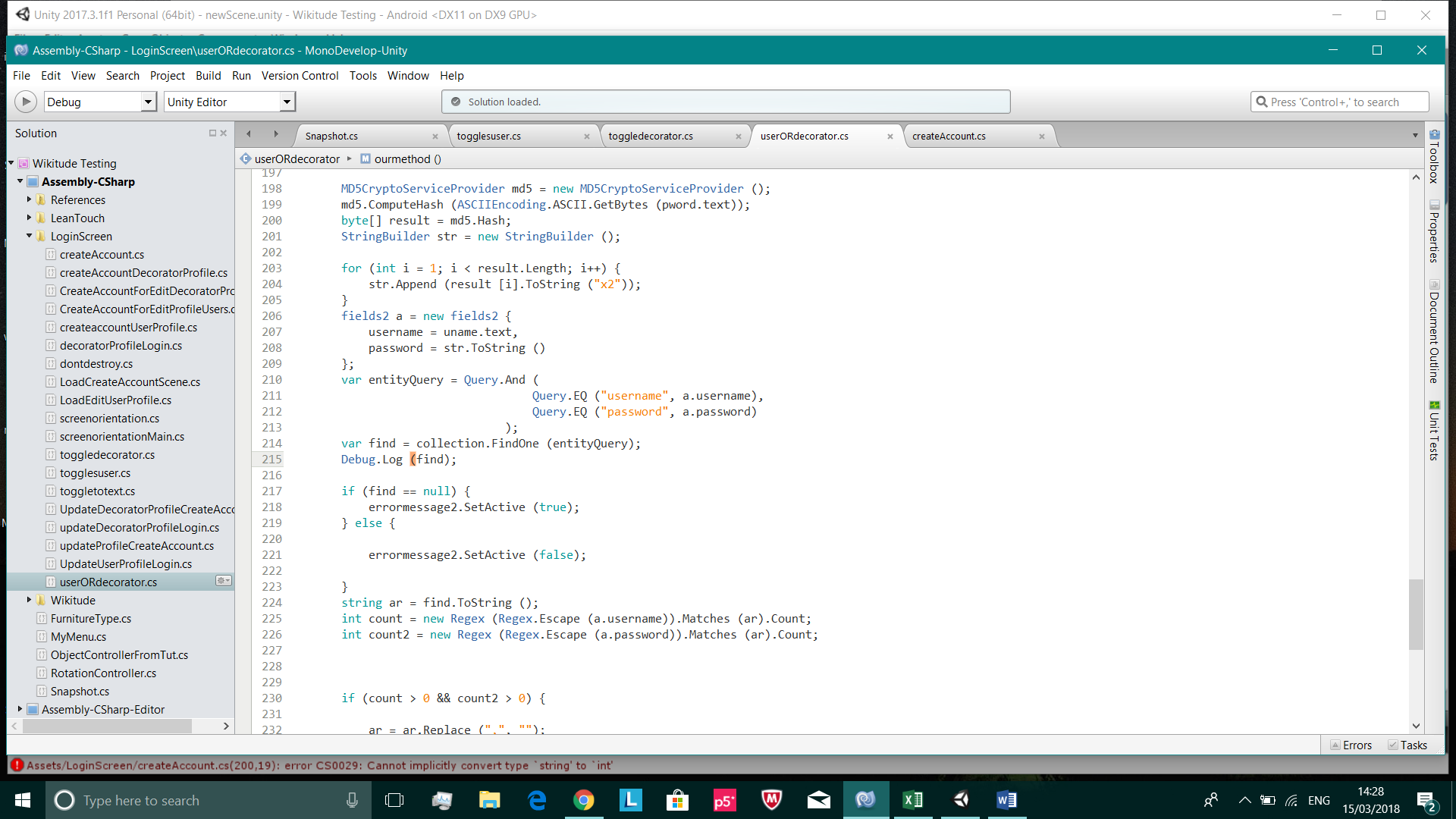


Correctly changed

/

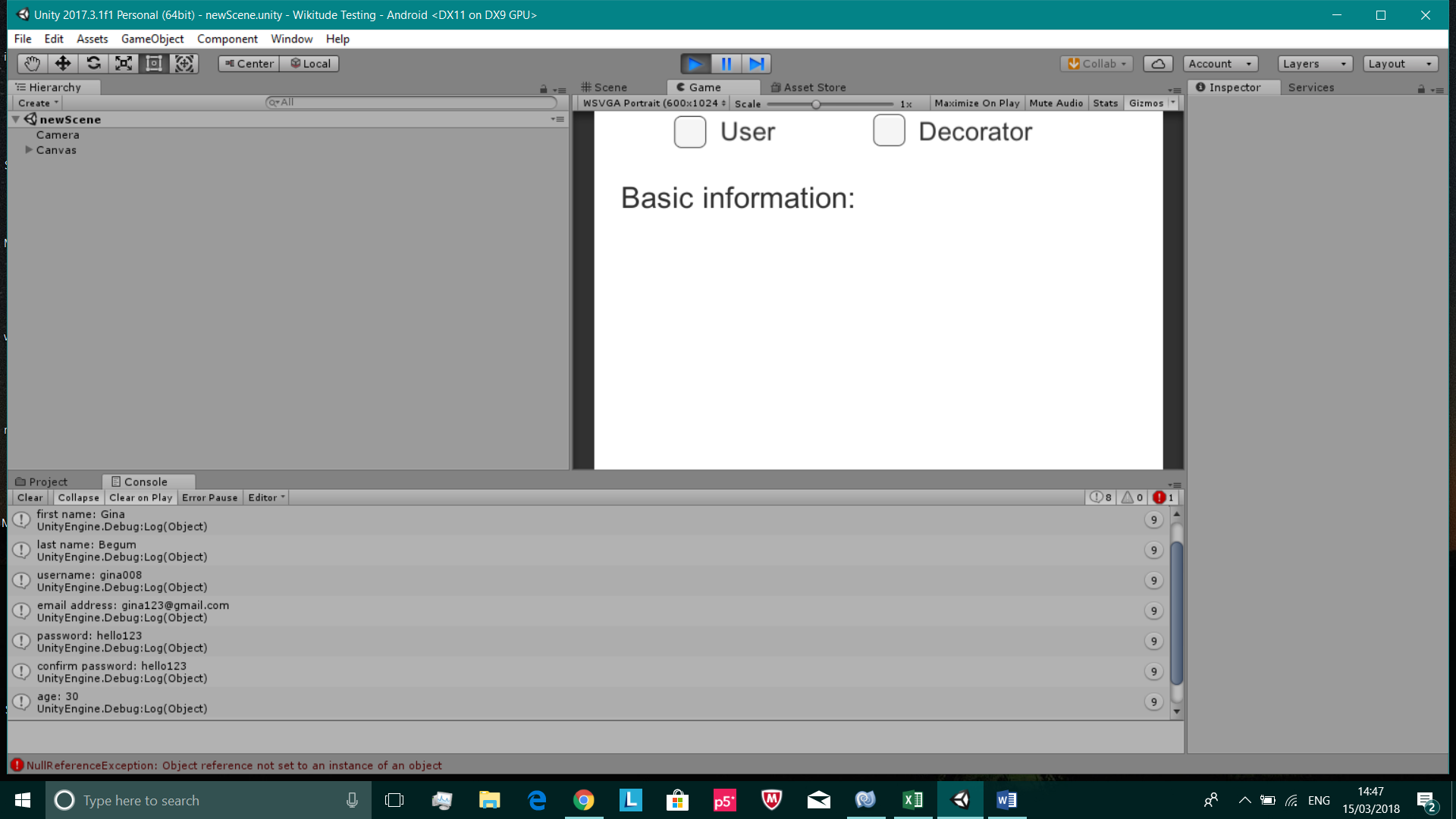
Dead code

**Test 2**



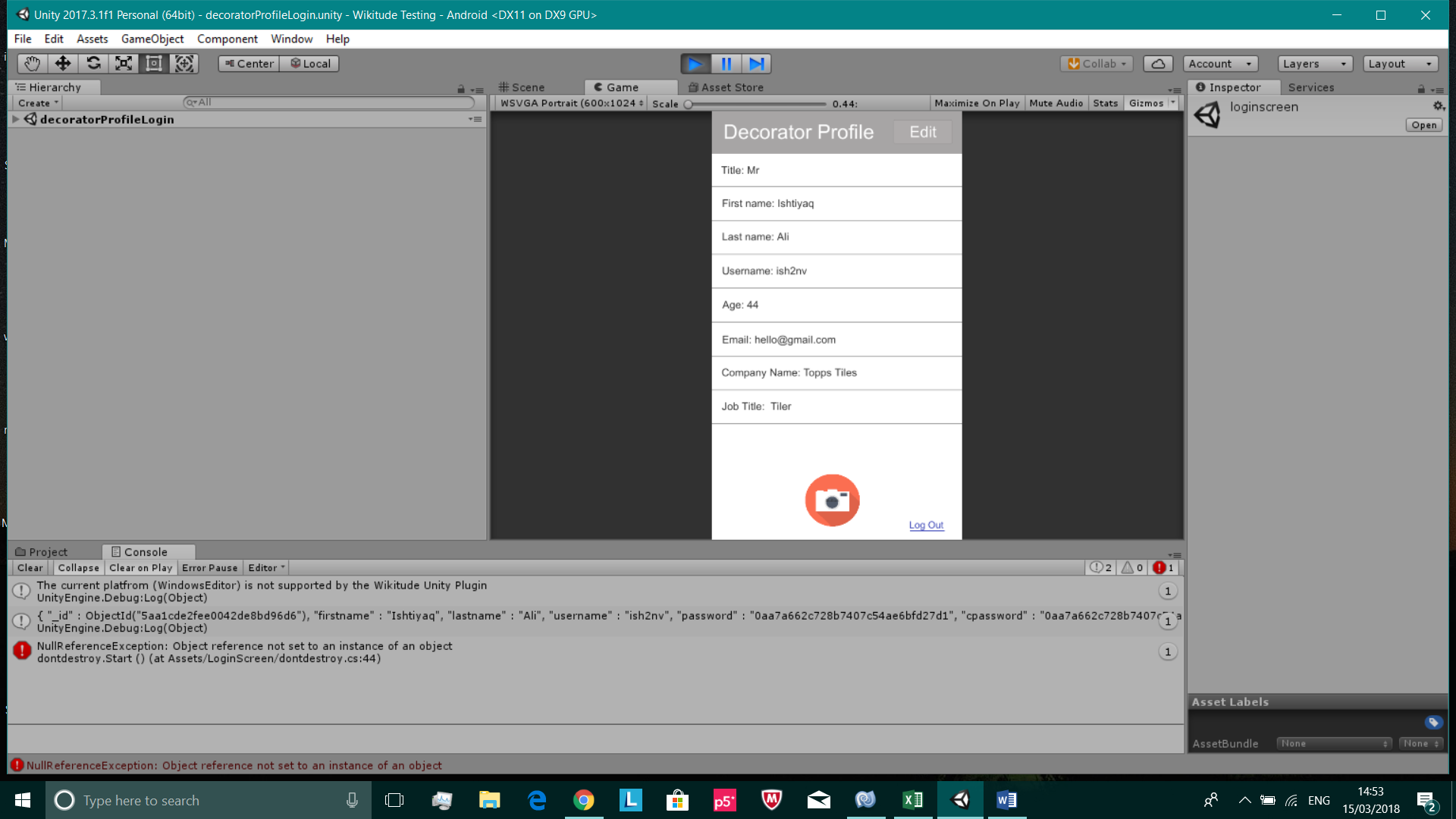
entityQuery is a variable that produces a NullPointerException. This variable only equals null, when user inputs a username or password that doesn’t exist in our database. This is a defect I can’t change since this is the only way in carrying out this important task

/

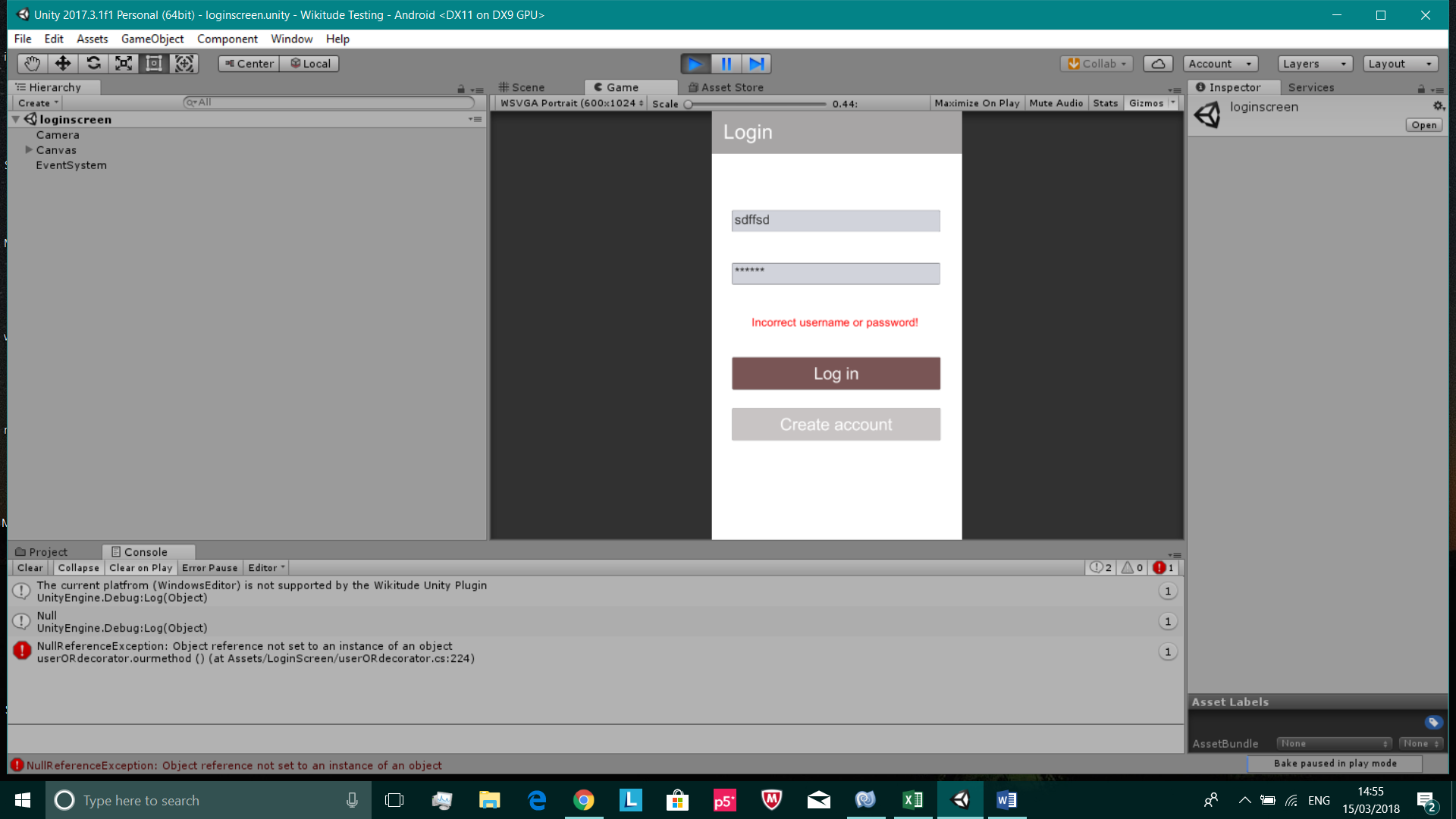
**Test 5**

Proof that this test was successful, since the expected output has been printed into our Unity console

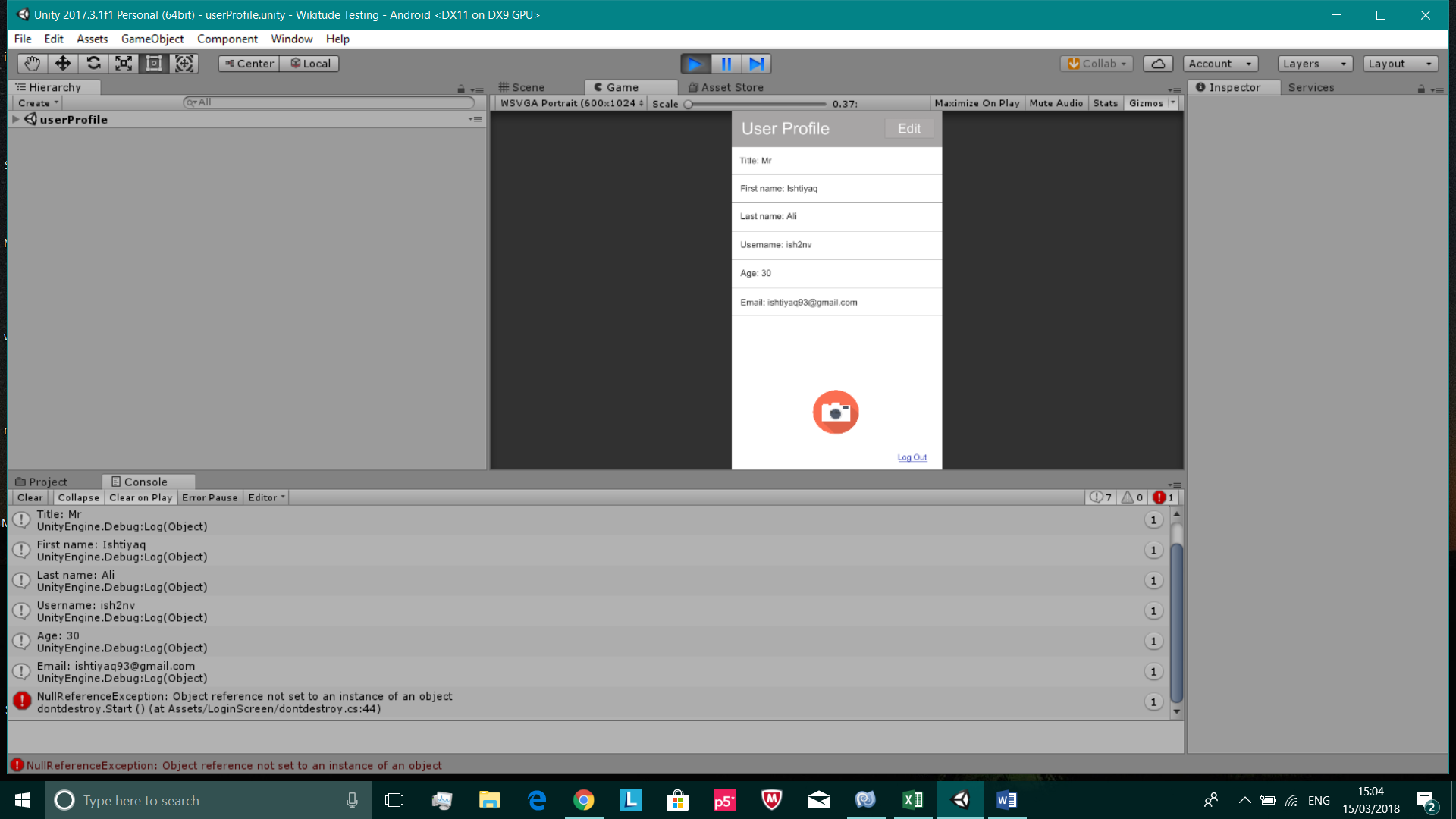
/

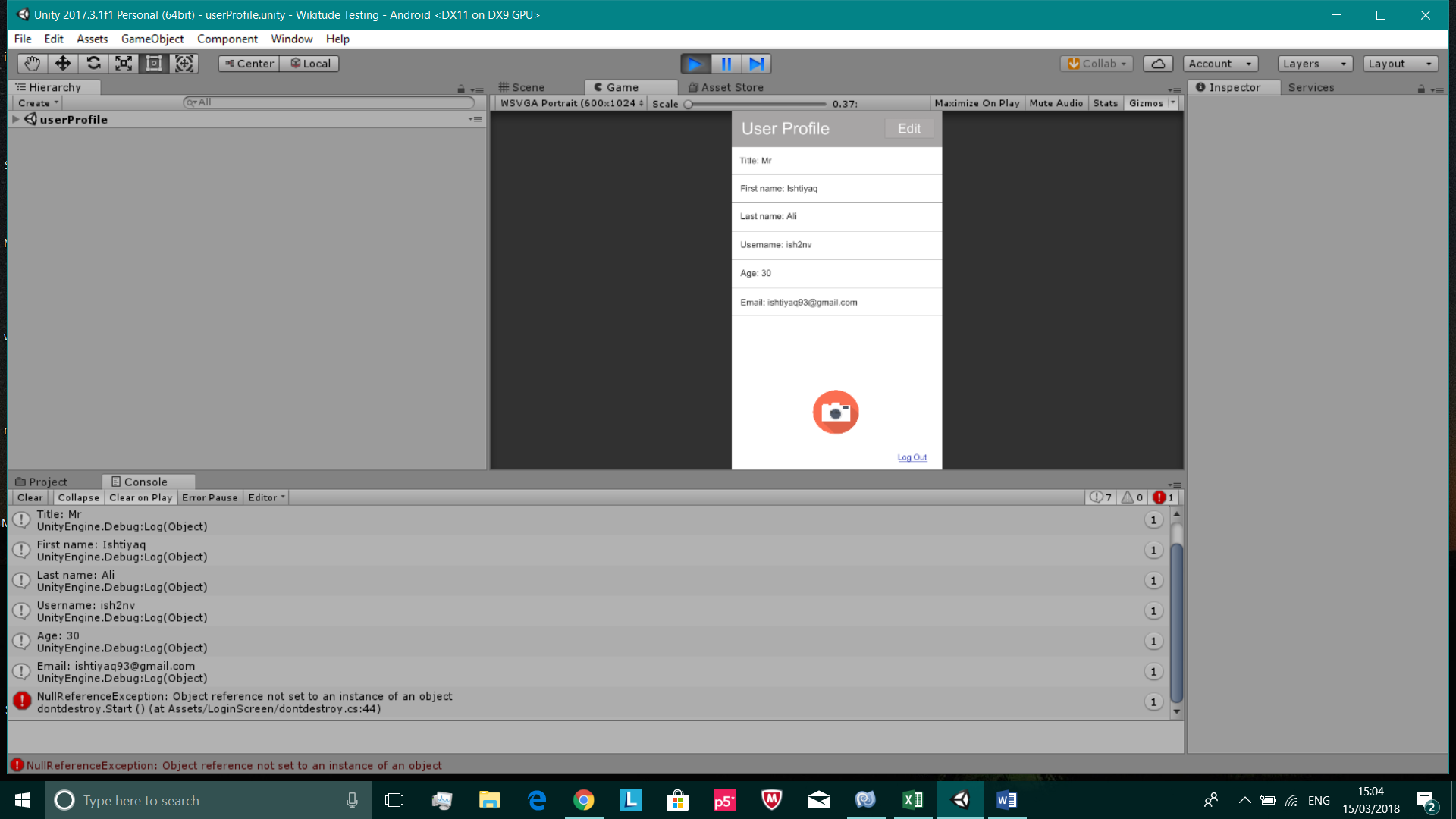
**Test 6**

Unity console is printing out the MongoDB document that was found from inputting a valid username & password in the login screen

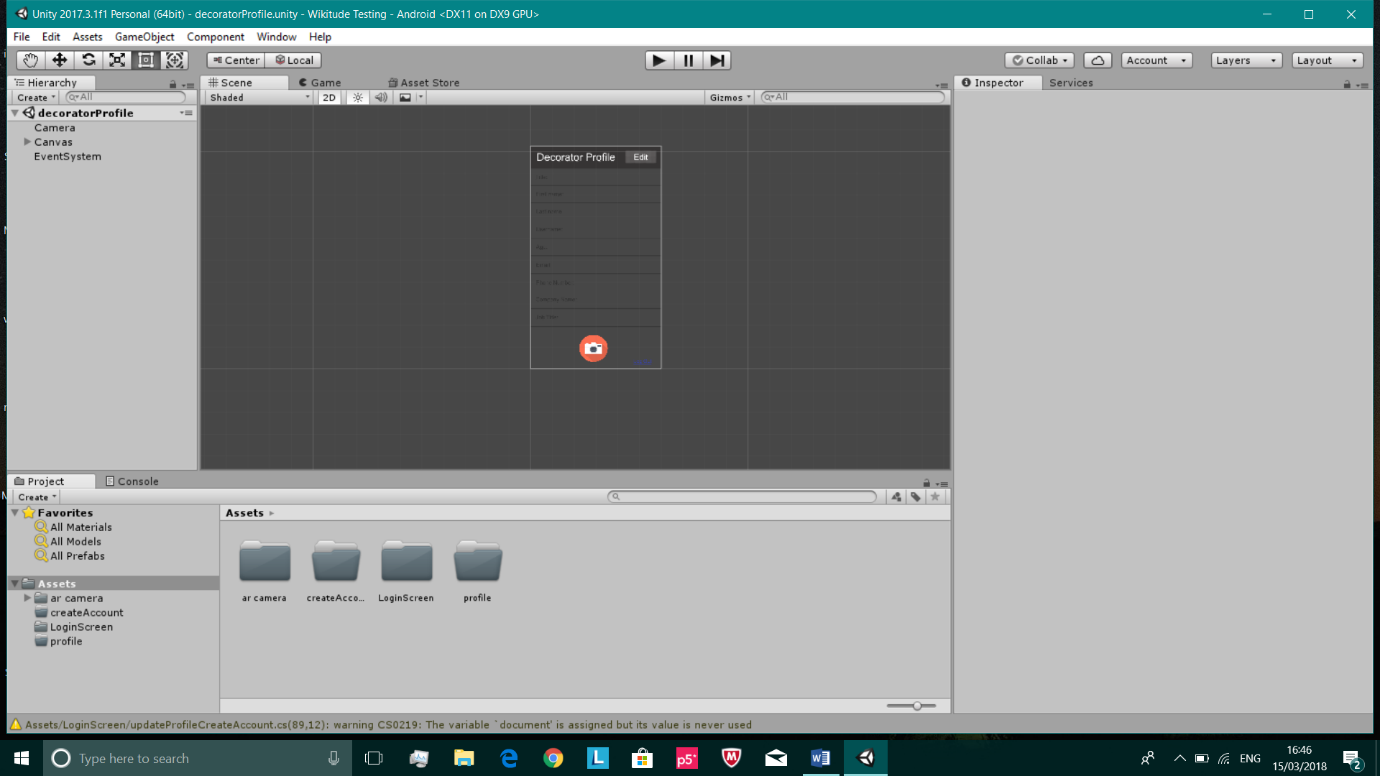


Unity console printing out null, since an invalid username & password was inputted in the login screen

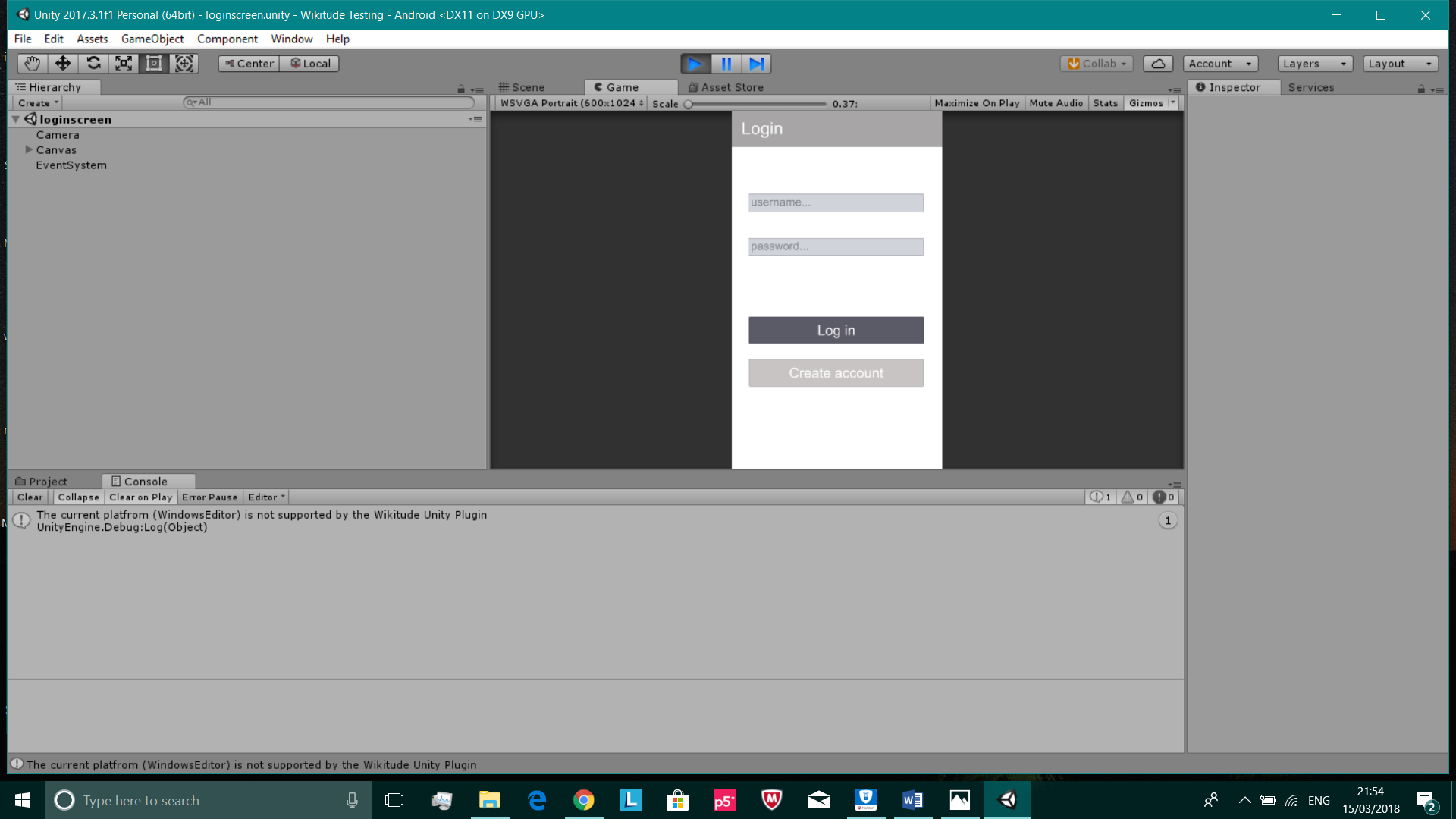
**Test 7**



Expected output has been printed in the Unity console and on the profile page. These screenshots are proof that this test was successful

**Test 9**

All unit tests have been integrated together in one Unity file



Proof that the integration was successful as there are no error messages. The Unity file can also be built to Android without any issues