InteriAR

**Final Report**

**Group H**

Cleon Grant

Ethan Newell

Tanzum Begum

Jibril Omar Ali

Ifrah Shahid

Liban Farah

Shah Ali

Project supervisor: Frederic Fol Leymarie

Goldsmiths College, University of London

Course code: IS52018C

Software Projects

30/04/2018

## Table of Contents

1. User Need Overview & Concept Introduction 1

2. Data gathering and requirements 1

3. Functional Specification 2

4. Ethical Audit 2

4.1 Privacy and data protection 2

4.1 Intellectual property 2

5. Design 3

6. Prototyping 4

6.1 Conceptual prototyping 4

6.2 Functional prototyping 5

Wall colouring 5

Augmented Reality Objects 6

Database 6

7. Technical Architecture 7

7.1 Database 7

7.2 Augmented Reality Implementation 7

7.3 Messaging 8

7.4 Payment 8

8. Evaluation Plan 8

9. Project Management 9

10. Conclusion 9

11. Bibliography 10

12. Appendices 12

12.1 Appendix A: Data gathering 12

12.2 Appendix B: Functional Specification 19

12.3 Appendix C: Design 20

12.3 Appendix D: Conceptual Prototyping 25

12.4 Appendix E: Functional Prototyping 29

12.5 Appendix F: Technical Specification 30

12.6 Appendix G: Project Management 32

## 

## Introduction

InteriAR, connecting decorator to designer. Our application will help users draft their dream room; combining augmented reality techniques with carefully selected decorators that can make their augmentation a reality.  Users will be able to choose 3D models of furniture and place it in their living space, complementing their existing layout to explore new designs. InteriAR takes it a step further by allowing its users to also change the colour of their walls to really get the full picture.

## Development Record

After conducting extensive market research, we approached companies and the general public in order to identify persons of interest, outlining them in a stakeholder diagram (Appendix A.1). The primary stakeholders were users who want an easier way to visualise a space before investing financially into decorating and regretting the outcome. InteriAR offers them a way to envision their project in detail, avoiding said risk. We conducted a survey, asking potential users “Does this idea appeal to you?”, of those who answered “yes”, the majority were in the age group 26-45 (Appendix A.5). This survey derived the users in that age range as major stakeholders as they were also more likely to be homeowners or long term renters compared to students for example (1). We created two user personas to showcase our general findings (Appendix A.2-A.3). We then gathered more data among users involving functions of the app; this data showed that 82.6% of users found the 3D viewing of the furniture to be the most appealing aspect of the application (Appendix A.5). This meant that that we listed 3D functionality of the application as a project requirement.

One other group of significant stakeholders identified were retailers of furniture. Retailers want the data on industry trends; for instance, what pieces of furniture are popular within the app, and which designs are liked the most. This would enable them to produce the type of furniture wanted by the public and keep a closer eye on industry patterns. We concluded that IKEA, being one of the world’s largest furniture producers with a market share of 8.2% in the UK in 2016 (2), was also a potential stakeholder. This gave rise to the idea that we would use IKEA’s furniture database within our application to allow users to design their home using their extensive furniture catalogue.

The final group of major stakeholders noted were decorating companies, which we gathered data from within the London region. In our sample, 72% of respondents said they would like to see such an application and 61% said that they ‘would be willing to pay a small percentage of job fees for the matching service’ (Appendix A.6).  The results confirmed that decorators are major stakeholders in the app which lead us to create a decorator persona (Appendix A.4).

The primary computational problems we will encounter include the following:

* Accurately projecting 3D objects into an augmented reality space.
* Coding the ability to virtually paint walls which will include image segmentation, colour distance calculating and more.

An explanation of the above with details can be found in the appendix (Appendix A.7).

## Formative Evaluation

The procedures below are illustrated in a functional architecture diagram which outlines the individual components and data paths (Appendix B).

InteriAR would have its users log in using an external API, eliminating the need for us to store sensitive data such as passwords. From that we will have a collection of decorators in a database that will hold reviews, location, and optionally a portfolio of past work. For the users we will store: user names, snapshots of designs, current orders and more.

We will utilise AR libraries combined with computer vision techniques to correctly project the orientation and position of 3D objects in an augmented space. Various image segmentation algorithms will aid in changing the colour of walls within the app. The user would potentially have to make changes to assist the projection and to input additional data where necessary. Once the user has finalised a design, they will be connected with decorators in our database over a long range wireless network. Their profiles will be displayed by area to the user who can contact them directly with our in-app messaging system. The messaging system will likely be using pre-existing libraries as a starting point.

Following the decorator accepting a design and agreeing a quote, an escrow payment system will be shown. This will also include existing API’s to aid in the structure of the transaction. Consequently, there will be an agreement which users must abide to, stating that the money will be released once the job has been completed. Upon completion, the user will have the option of reviewing the decorator and allowing them to use the captured augmentation on their profile to help them build their reputation.

## Design and Implementation

### 4.1 Privacy and data protection

Customer privacy and data protection is essential to maintaining an ethical project. We will be adhering to the Data Protection Act 1998 and any new laws coming into place (3, 4). Acknowledge our role as both the data controller and processor. The primary way we will deal with this responsibility will be to minimise the sensitive data we store and process. One method for achieving this will be utilising external authentication API as outlined in the functional specification to avoid storing user passwords.  In terms of privacy, all public sharing of design snapshots will be strictly opt-in.

### 4.1 Intellectual property

We will make sure to adhere to all licensing on any software and assets we utilise. For example, our database of 3D objects for an initial version is free to use providing we cite their paper (5). Regarding the software development kits, we will be using non-commercial or educational licenses, which would need to be updated should the project ever launch commercially.

We confirm that we will not be working with nor providing our app to minors or vulnerable adults.

## Quality Assurance

### 5.1 Initial Requirements

Functional testing:

* All error messages should display when needed
* All buttons should function accordingly
* Gallery should display all images taken by user and fill up all available slots. Screenshots that are taken after should not overwrite the images within the slots

Non-functional testing:

* Our database should be able to keep user password secure
* All screens within the application should be user-friendly
* Our database should be able to store over 100,000 user accounts
* AR camera should be compatible with majority android devices
* Our app should be able to handle multiple users at once

Static and dynamic testing:

* There should be no defects in any of the C# scripts
* All unit tests should produce no errors
* Integration tests should work successfully and produce no errors

### 5.2 Our approach to quality assurance

To maintain the desired level of quality in our application, we came up with various scenarios that we tested for our functional and non-functional tests. These scenarios are based on areas that we didn’t really focus on as much, areas we didn’t develop well or areas that we were unsure about.

Another way we maintained the desired level of quality is by coming up with various defects that we tested for our static and dynamic tests. The aim of this was to find bugs within our C# scripts and compare expected outputs with our actual outputs to see if they were the same. This helped clean up our code, remove errors and prevent more bugs from being made.

### 5.3 How well our final system conforms to our initial requirements

From the results of our tests, we can see that some failed to work and meet our initial requirements. Our functional tests suffered a few failures. One of them was failing to display an error message within the login screen whenever the user inputted an invalid username and/or invalid password. We were able to fix this error at the end of it, thus justifying the importance of software testing as we were able to identify our failure, fix the failure and deem it as successful, but I digress. The other failure was in our profile page, where screenshots taken by the user within the AR camera would overwrite previously taken screenshots stored in the gallery. We expected the images to not overwrite.  Unfortunately, this is something we could not fix as we did not have enough time.

Our non-functional tests also suffered a few failures. One of them failed our usability tests, where we tested to see if all our screens were user-friendly. We found that our create account screen forced users to input a lot of text, making it difficult to use. We fixed this issue by simply reducing the number of input fields. The other failure involved our compatibility test, where we found that top phones like Samsung Galaxy S7 Edge were not able to render the AR camera properly. We were able to fix the rendering issue for most Samsung phones, but there are other mobile devices that have the same issue, which we couldn’t entirely fix.

In some of our static tests, we were able to find defects within our C# scripts. Generally, it would be defects such as unused variables, dead code and variables with undefined values which we were able to easily fix. But in rare instances, we found string variables that we know would produce a NullPointerException when they are equal to null. These string variables query our MongoDB database for specific reasons (e.g. to find whether the inputted username and password exists in the database), and if it doesn’t find anything in the database, then it returns null. This is something that could be easily fixed with a try-catch block, but the query does not work when I use this. Hence, I had to leave this defect in the program.

### 5.4 Any changes made to initial requirements?

The only change I made to the initial requirements was in the non-functional testing section, where I initially stated that there should be a backup database. I found out that we would need to pay over twenty pounds to have a backup in MLab.com (where our Mongo database is stored.) Paying for resources is something we all agreed to not do, and as a result I erased this from the initial requirements.

### 5.5 Software Testing

All tests collected via a table can be found in **12.5 Appendix E** All results/evidence of the tests can be found in **12.6 Appendix E**

## Summative Evaluation

### 6.1 Conceptual prototyping

For our conceptual prototype, we made a preliminary wireframe of our app (Appendix D.1-D.2). We created a low fidelity prototype using MarvelApp (6) which outlines the steps that the users would follow throughout the app.

Upon completing the conceptual prototype, we constructed a survey to gain an insight into user opinion (Appendix D.3).This feedback was vital, as we gained a greater understanding on the preferences of our users, causing us to rethink and adapt our prototype.

Below are all the changes we made based on the user feedback:

* The budget filter was pointed out by a decorator as unfeasible due to most decorators not charging fixed prices for their service.
* Added the options of logging in via Google and Facebook. It was gathered that it may be tedious to create a new account and password for a new app. We have adjusted our design by replacing this option with popular authentication methods.
* Changed the camera screen orientation to landscape.
* Added an FAQ section to the help screen.

### 6.2 Functional prototyping

The functional prototyping for InteriAR consisted of three main technical questions:

* Is virtual wall colouring feasible to implement and what is the best method of doing so?
* Is “marked tracking” a viable method of us displaying and moving 3D objects in our augmented reality space?
* Will MongoDB be able to handle the volume of users and transactions the app may need in future?

#### Wall colouring

#### 

This prototype was created on Processing 3 (7) using the Ketai for Android library (8)to access the camera on a mobile device. The software allows a user to tap a pixel on the live video, grabbing the RGB values from it. It then analyses every pixel on the camera feed and calculates whether they are similar enough to the captured colour. If so, it is repainted red.



Figure 1. InteriAR wall colouring functional prototype v1

The prototype manages to successfully detect part of the surrounding wall, as well as avoid the more distinctive obstacles. Unfortunately, it also misses out large portions and does pick up some unwanted additions. Performance is notably compromised when calculating colour distance on each pixel in the feed, on a mobile-phone processor, as the program starts to stutter.

What we have learned from this prototype:

* It is feasible to implement. Even on a very basic level this functioned reasonably.
* Further research into colour matching is required for improved accuracy.
* We plan to better the performance drastically, either by grouping pixels together or relying on another method of detection such as image segmentation.

Research into software utilising similar features can be seen in the appendix (Appendix E).

#### Augmented Reality Objects

#### 

This prototype was created on Unity3D (9) using the Vuforia AR library (10). The software utilises a database of markers which are images of real objects or surfaces with enough unique features to be distinguishable from the surrounding area. Computer generated 3D objects are then assigned a marker so that when the camera detects it, the object will be superimposed upon it wherever it moves.



Figure 2. InteriAR objects prototype v1 in Unity                                 Figure 3. Marker with “features” highlighted

The close up functionality is impressive, even when dealing with inconsistent lighting. However as soon as the range increases past 2-3 meters, it quickly becomes incapable of consistently tracking the markers.

****

What we have learned from this prototype:

* Recognition based tracking is only viable at close range.
* While very basic, this software ran seamlessly on mobile.
* It may still be useful for some elements within our project, though projection based AR may be a finer alternative.

Figure 4. InteriAR objects prototype v1 utilising Vuforia

#### Database

To assess the feasibility of using MongoDB (11)for our project we implemented a cloud-based database using mLab services (12).We wanted to test if the cloud-based service was capable of handling large quantities of data being interpolated and updated via Pymongo scripts (13).

To do this, we used python to insert 1,000,000 user documents into a collection and ran ‘find’ commands to select users based on field properties. Both the insertion and any interactive find/update script I ran functioned smoothly and in a timely manner.

What we have learned from this prototype:

* We can efficiently add users to our database using python scripts.
* We can update, remove and pull information from the documents on the database in real-time.

## Conclusion

To summarise, we are proposing to develop what we believe to be a genuinely unique and innovative idea. While the individual components of our project may currently exist, there is nothing with the full package of complimenting features we aim to provide. These features have been chosen based on both user-needs gathered from market research and our drive to overcome computational challenges.

We will be implementing augmented reality, computer vision techniques, non-trivial databases and more; whilst maintaining intuitive user interaction and ethical data practices. To ensure feasibility of our concept, we conducted a range of conceptual and functional prototyping procedures, learning a lot in the process to take forwards.

Our team has a clear plan for implementing, testing and evaluating our project to ensure timely progress, resulting in an impressive product.

We firmly believe InteriAR is a concept worth realising.

## Bibliography

1. Barton C. Home ownership and renting: demographics [Internet]. Researchbriefings.parliament.uk. 2017 [cited 3 December 2017]. Available from: <http://researchbriefings.parliament.uk/ResearchBriefing/Summary/CBP-7706>
2. IKEA T, release P, results I. IKEA UK business results - IKEA [Internet]. Ikea.com. 2017 [cited 3 December 2017]. Available from: http://www.ikea.com/gb/en/this-is-ikea/newsroom/press-release/ikea-uk-business-results/
3. Data Protection Act 1998 [Internet]. Legislation.gov.uk. 1998 [cited 3 December 2017]. Available from: <https://www.legislation.gov.uk/ukpga/1998/29>
4. General Data Protection Regulation (GDPR) – Final text neatly arranged [Internet]. General Data Protection Regulation (GDPR). 2017 [cited 3 December 2017]. Available from: https://gdpr-info.eu/
5. J. Lim J, Pirsiavash H, Torralba A. Parsing IKEA Objects: Fine Pose Estimation. ICCV; 2013.
6. Marvel App - Free mobile & web prototyping (iOS, Android) for designers – Marvel [Internet]. Marvel Prototyping. 2017 [cited 6 November 2017]. Available from: <https://marvelapp.com/prototyping/>
7. Processing.org [Internet]. Processing.org. 2017 [cited 12 December 2017]. Available from: https://processing.org/
8. Ketai [Internet]. Ketai.org. 2017 [cited 21 November 2017]. Available from: <http://ketai.org/>
9. Unity software [Internet]. Unity3d. 2017 [cited 13 December 2017]. Available from: <https://unity3d.com/>
10. Vuforia | Augmented Reality [Internet]. Vuforia.com. 2017 [cited 29 November 2017]. Available from: <https://www.vuforia.com/>
11. MongoDB for GIANT Ideas [Internet]. MongoDB. 2017 [cited 11 December 2017]. Available from: <https://www.mongodb.com/>
12. MongoDB Hosting: Database-as-a-Service by mLab [Internet]. mLab. 2017 [cited 11 December 2017]. Available from: <https://mlab.com/>
13. pymongo 3.6.0 : Python Package Index [Internet]. Pypi.python.org. 2017 [cited 11 December 2017]. Available from: <https://pypi.python.org/pypi/pymongo>
14. Facebook Login - Documentation - Facebook for Developers [Internet]. Facebook for Developers. 2017 [cited 11 December 2017]. Available from: <https://developers.facebook.com/docs/facebook-login>
15. Google Identity Platform  |  Google Developers [Internet]. Google Developers. 2017 [cited 11 December 2017]. Available from: https://developers.google.com/identity/
16. OAuth 2.0 — OAuth [Internet]. Oauth.net. 2017 [cited 12 December 2017]. Available from: https://oauth.net/2/
17. Győrödi C, Gyorodi R, Pecherle G, Olah A. A Comparative Study: MongoDB vs. MySQL. University of Oradea; 2015.
18. Android Studio and SDK Tools | Android Studio [Internet]. Developer.android.com. 2017 [cited 12 December 2017]. Available from: <https://developer.android.com/studio/index.html>
19. Barghout L, Sheynin J. Real-world scene perception and perceptual organization: Lessons from Computer Vision. Journal of Vision. 2013;13(9):709-709.
20. Shapiro L, Stockman G. Computer vision. New Jersey, Upper Saddle River: Prentice Hall; 2001.
21. Wikitude Augmented Reality- The World's Leading Cross-Platform AR SDK [Internet]. Wikitude. 2017 [cited 11 December 2017]. Available from: <https://www.wikitude.com/>
22. ARCore - Google Developer  |  ARCore  |  Google Developers [Internet]. Google Developers. 2017 [cited 13 December 2017]. Available from: https://developers.google.com/ar/
23. ARKit - Apple Developer [Internet]. Developer.apple.com. 2017 [cited 13 December 2017]. Available from: https://developer.apple.com/arkit/
24. Messaging SDK and Chat API for Mobile Apps and Websites | SendBird [Internet]. SendBird. 2017 [cited 13 December 2017]. Available from: <http://sendbird.com>
25. Paypal.com. (2017). *PayPal UK: Pay, Send Money and Accept Online Payments*. [online] Available at: https://www.paypal.com/gb/home [Accessed 2 Dec. 2017].
26. balanced/balanced-python [Internet]. GitHub. 2017 [cited 11 December 2017]. Available from: <https://github.com/balanced/balanced-python>
27. Trello [Internet]. Trello.com. 2017 [cited 13 December 2017]. Available from: https://trello.com/
28. Martin R. Agile software development. Harlow, UK: Pearson; 2014.
29. Comport A, Marchand E, Pressigout M, Chaumette F. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. IEEE Transactions on Visualization and Computer Graphics. 2006;12(4):615-628.
30. Klein G. Visual Tracking for Augmented Reality [Ph.D]. University of Cambridge; 2006.
31. P. Mountney, D. Stoyanov, A. Davison, and G.-Z. Yang, Simultaneous stereoscope localization and soft-tissue mapping for minimal invasive surgery, in Proc. Int’l Conf. Medical Image Computing and Computer Assisted Intervention (MICCAI), R. Larsen, M. Nielsen, and J. Sporring, eds., vol. 4190 of Lecture Notes in Computer Science, Springer-Verlag;2006
32. The Dulux Visualizer App [Internet]. Dulux.co.uk. 2017 [cited 15 December 2017]. Available from: https://www.dulux.co.uk/en/articles/dulux-visualizer-app

## 12. Appendices

### 12.5 Appendix E: Software Testing – Data Collection

**Functional testing**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test no. | Test Scenario | Example test case | Precondition | 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 | 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    Test case 2:  Title: Mr  First name: Johnson  Last name: Smith  Username: js9056  Age: 42  Email:  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 |

**Non-functional testing**

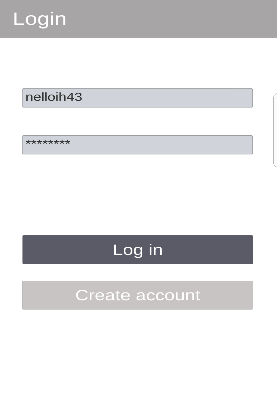
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test no. | Test type | Test scenario | Pre-condition | Test steps | Expected outcome | Actual Results | Pass/Fail |
| 1 | Database Security testing | Test if mongo database displays hashed version of password under the password 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 a 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 by 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 |

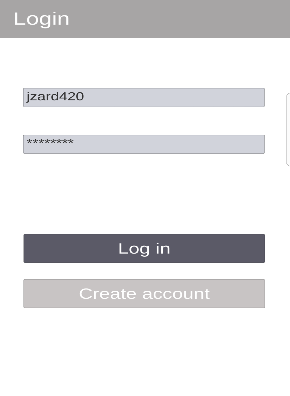
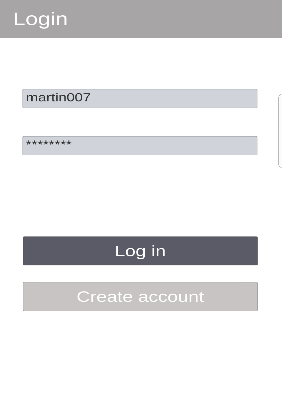
**Static and dynamic testing**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 queries 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. variable equal to null that’ll produce a NullPointerException. This variable only produces null, if user inputs a username & password that doesn’t exist in our database i.e. invalid username or password. 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 | 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 c  onsole | 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. And to see if the program updates each time the input fields get updated with new text    **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 |
| 5 | 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 |
| 6 | 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 | Expected outputs matches actual outputs! No defects found! | Fail |
| 7 | 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 |

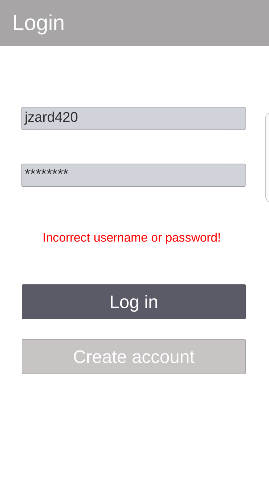
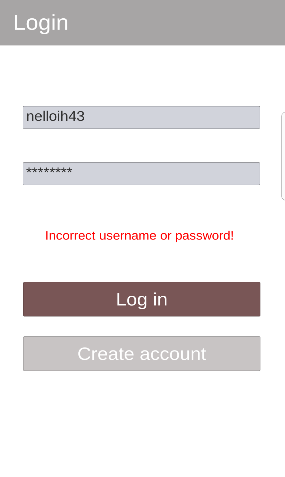
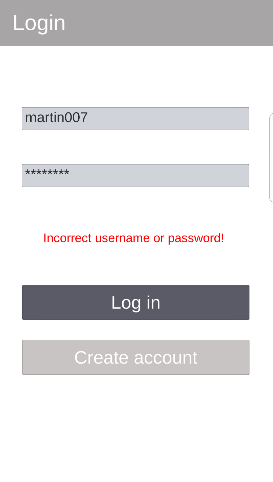
### 12.6 Appendix E: Software Testing – Evidence/Results

**Functional Testing**

**Test 1**

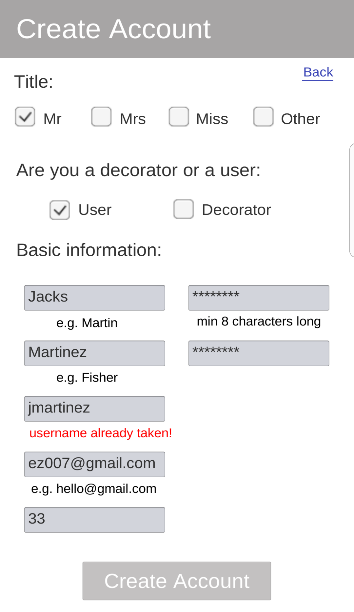


No error message displayed in any of the screenshots when entering incorrect username and/or password. this is a problem, since user will not know whether they are making a mistake or whether the app is not responding

**Changes to fix error**

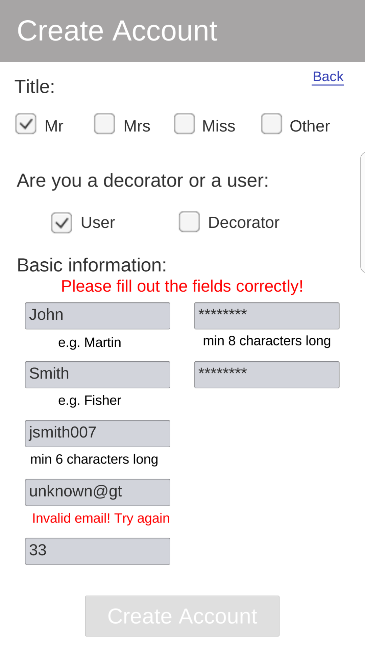
When the log in button is pressed it now displays an error message in red. The awareness from this functional test enabled me to notice this mistake and rightly fix it immediately

**Test 2**

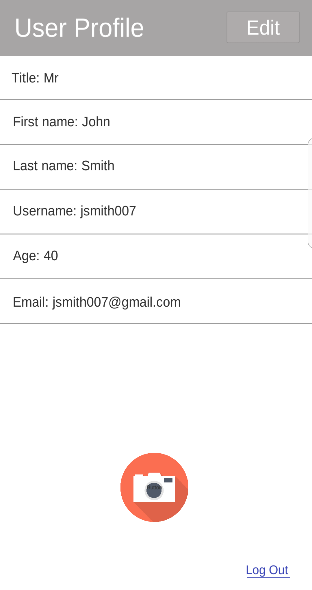
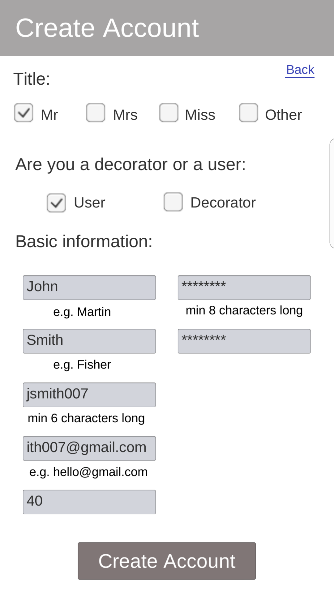


The example data “jmartinez” produces the following error message “username already taken.” This is proof that the error message pops-up when user inputs a username that has already been created

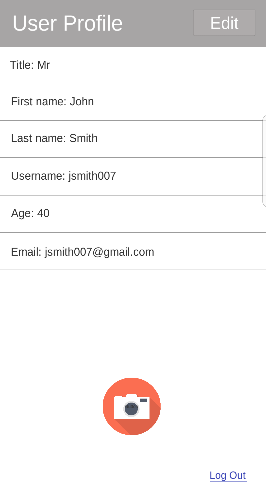
**Test 3**



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

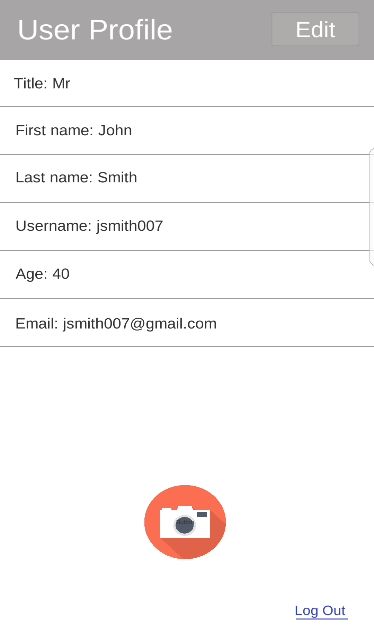
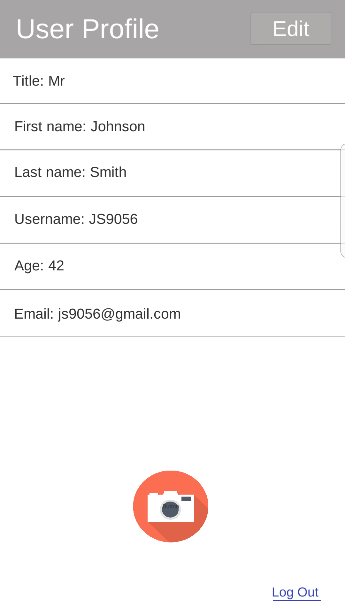
**Test 4**

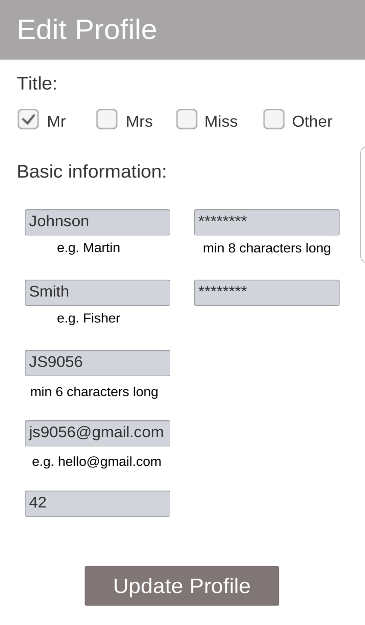
First, we create an account. Then enter our user profile. As you can see, the text inputted in the fields within the create account screen display within the user profile, meaning the user profile was able to extract information about the user correctly and display it

**Test 5**



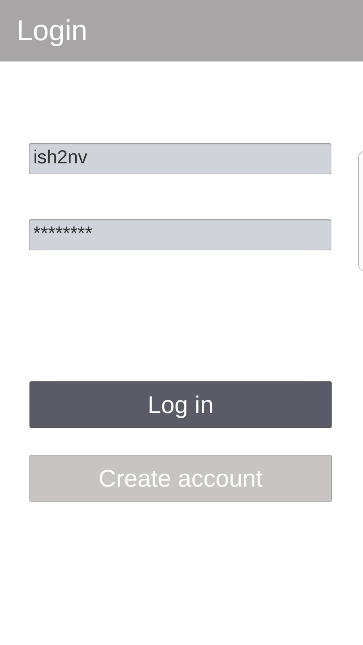
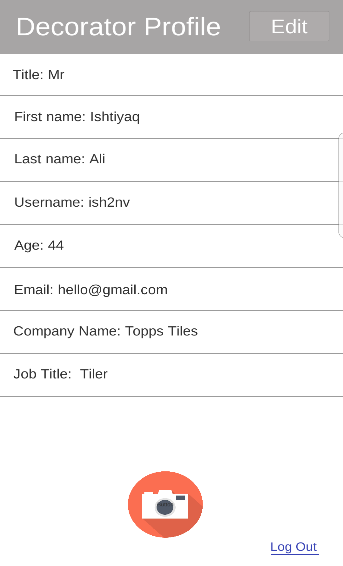
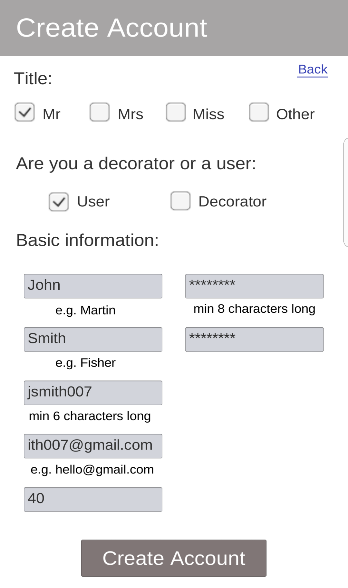
Camera button within the profile page can successfully load the AR camera and display it on the mobile screen. The load time speed is usually between 1 to 5 seconds, which is not bad considering how technical/advanced an AR camera is

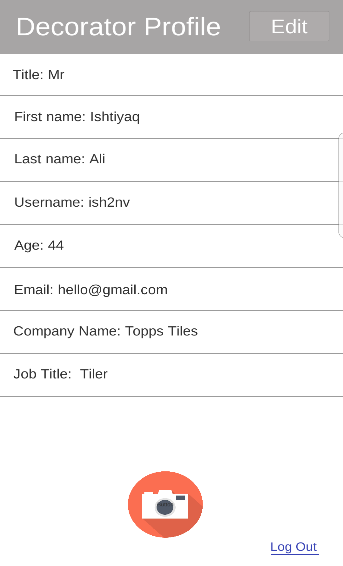
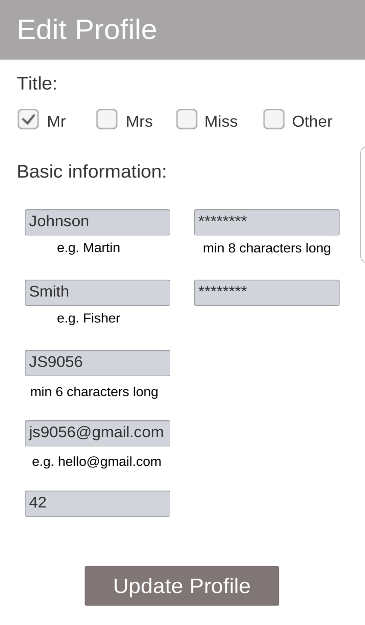
**Test 6**

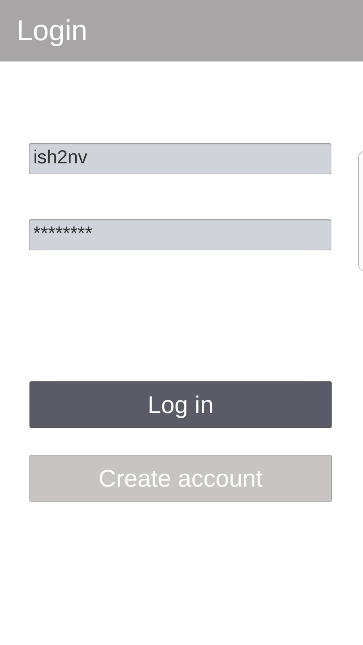


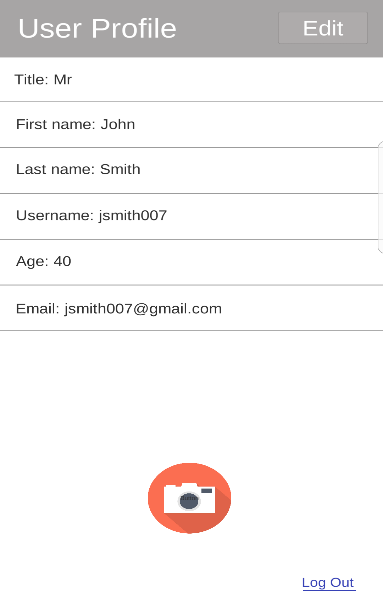
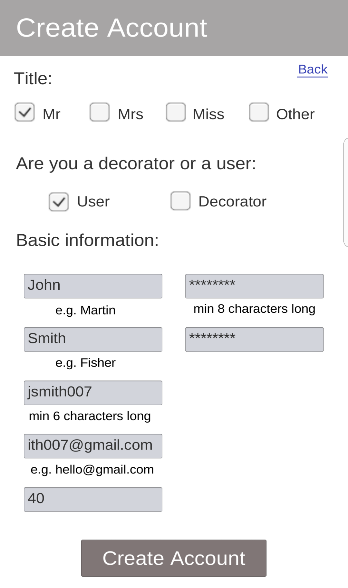
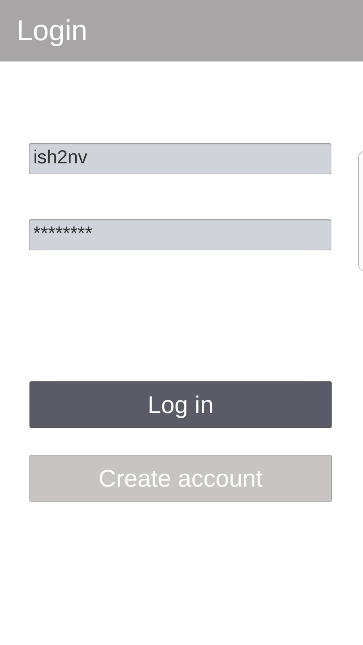
These series of screenshot are trying to illustrate how our application can update a user’s profile. In the first screenshot it shows fields with user information. Second screenshot shows the user going into the edit profile screen. This is where the user updates their profile. Last screenshot shows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

**Test 7**

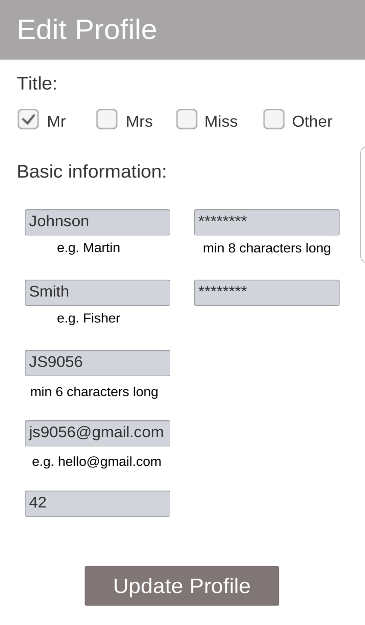
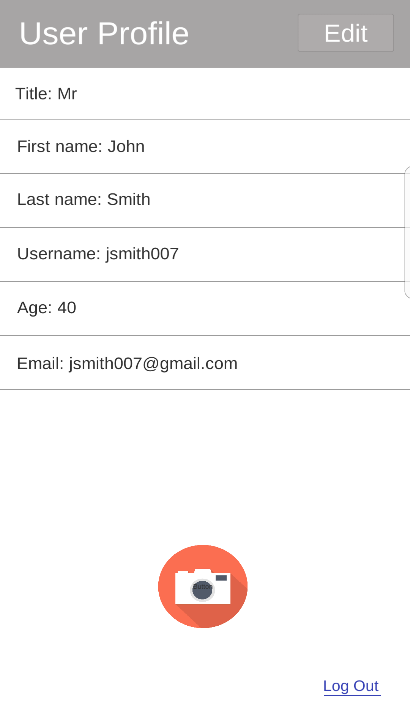
**Login screen – navigates to profile page & create account**

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

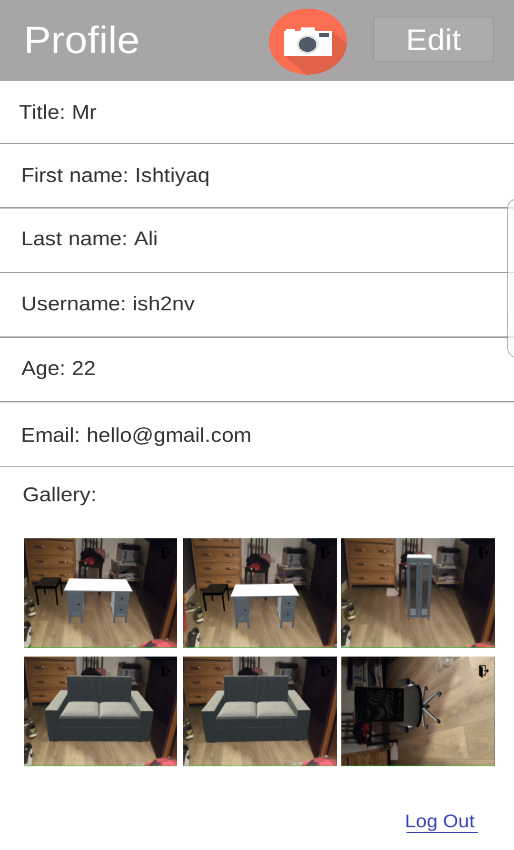
****

**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)**



**Test 8**

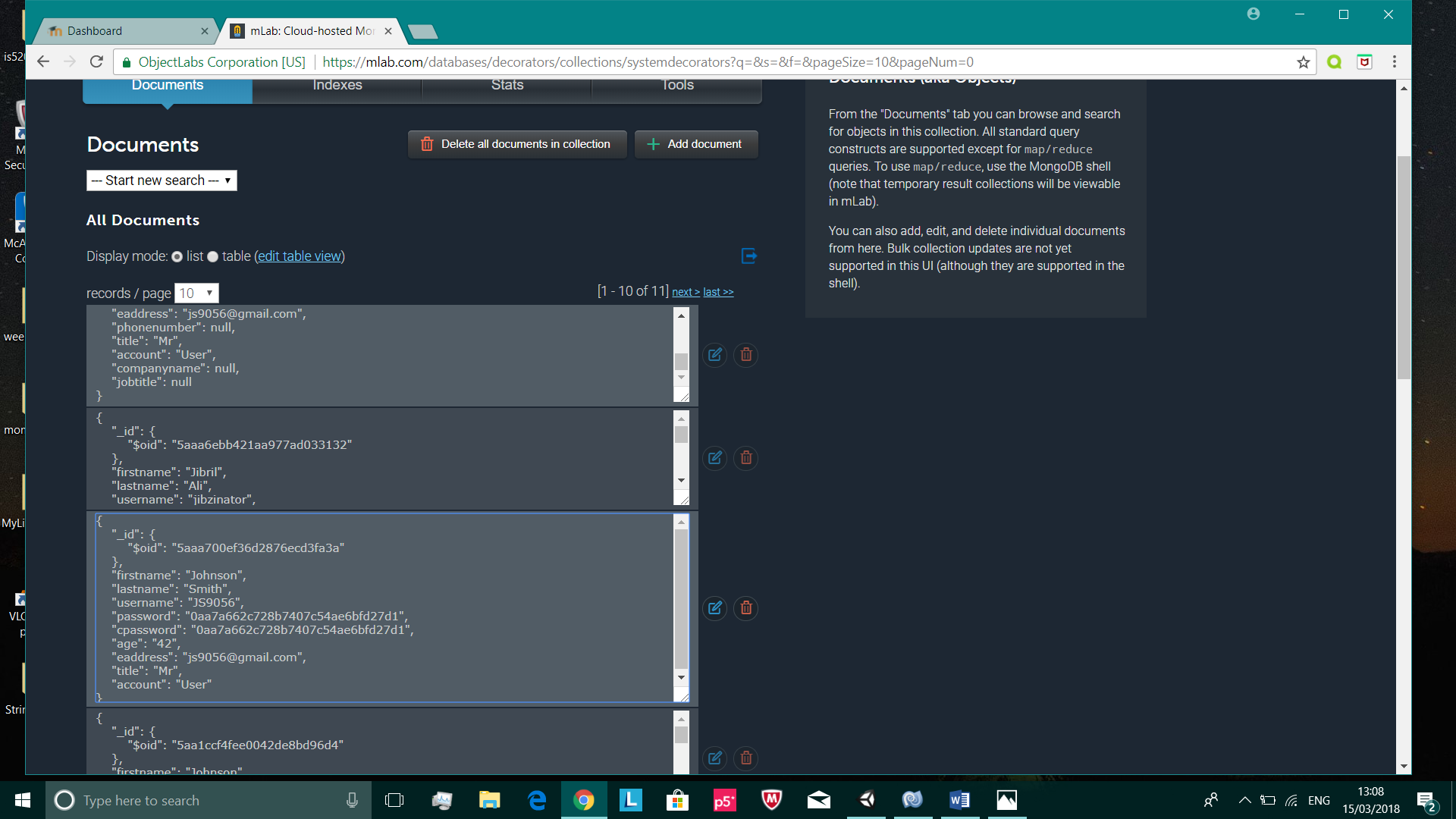


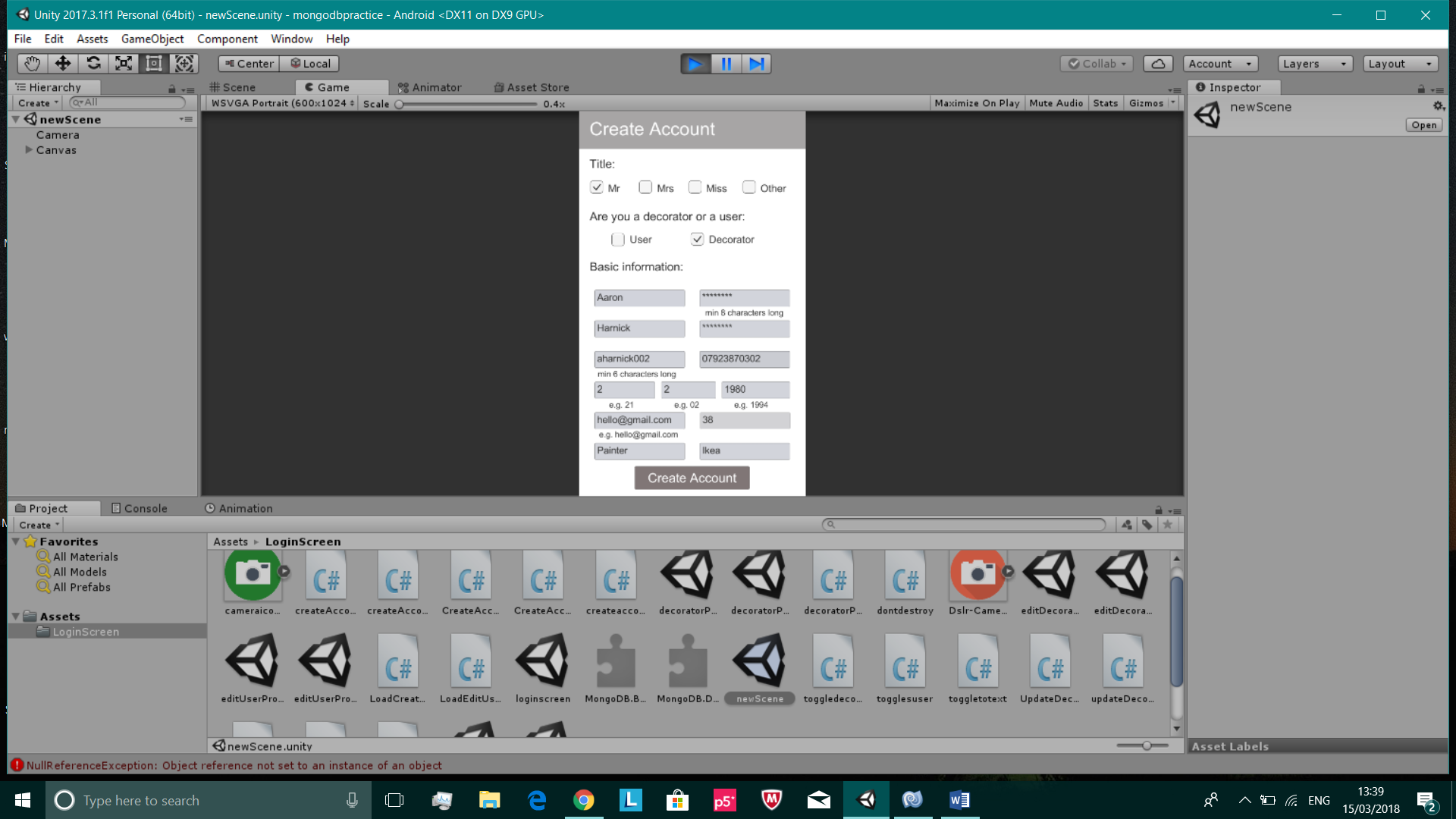
As you can see now images have changed. This is because all images have been overwritten when users take more screenshots. This is something that we didn’t want our app to do, however it is something we couldn’t figure out how to fix in time

All images do display in the gallery within all available slots

**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

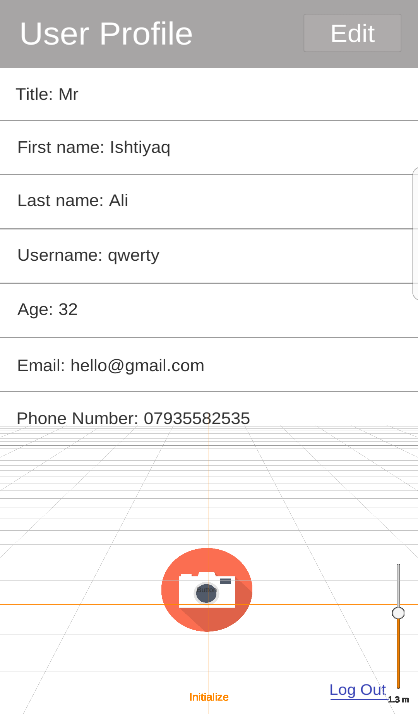
**Test 1**

**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**



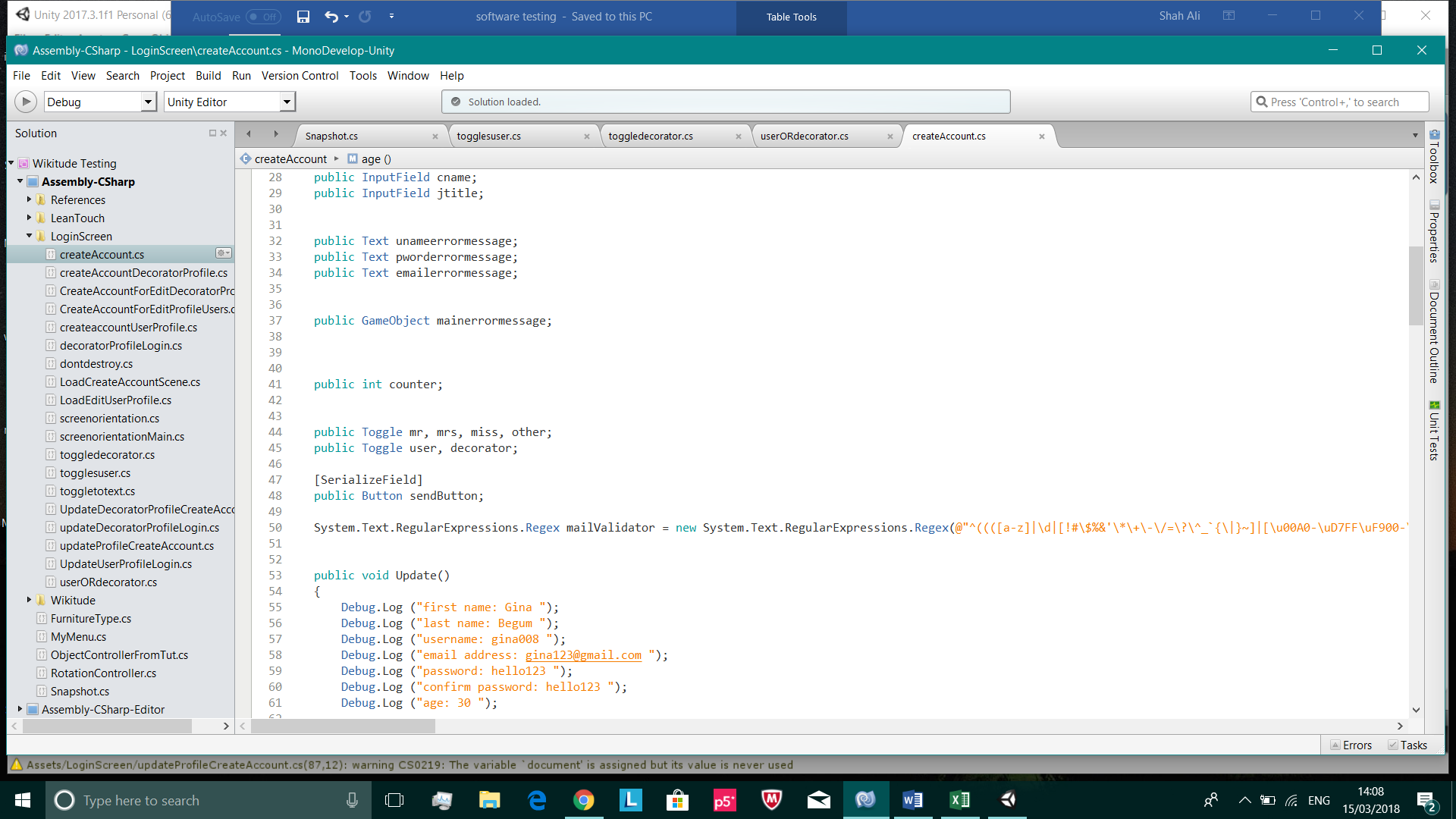
The AR camera in our application opens almost instantly. Loading speed is at a very good rate for most devices. But there are certain devices that have issues in opening the AR camera. Some may not open on the first or second try. And some do not open the AR camera at all. The screenshot on the left 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**

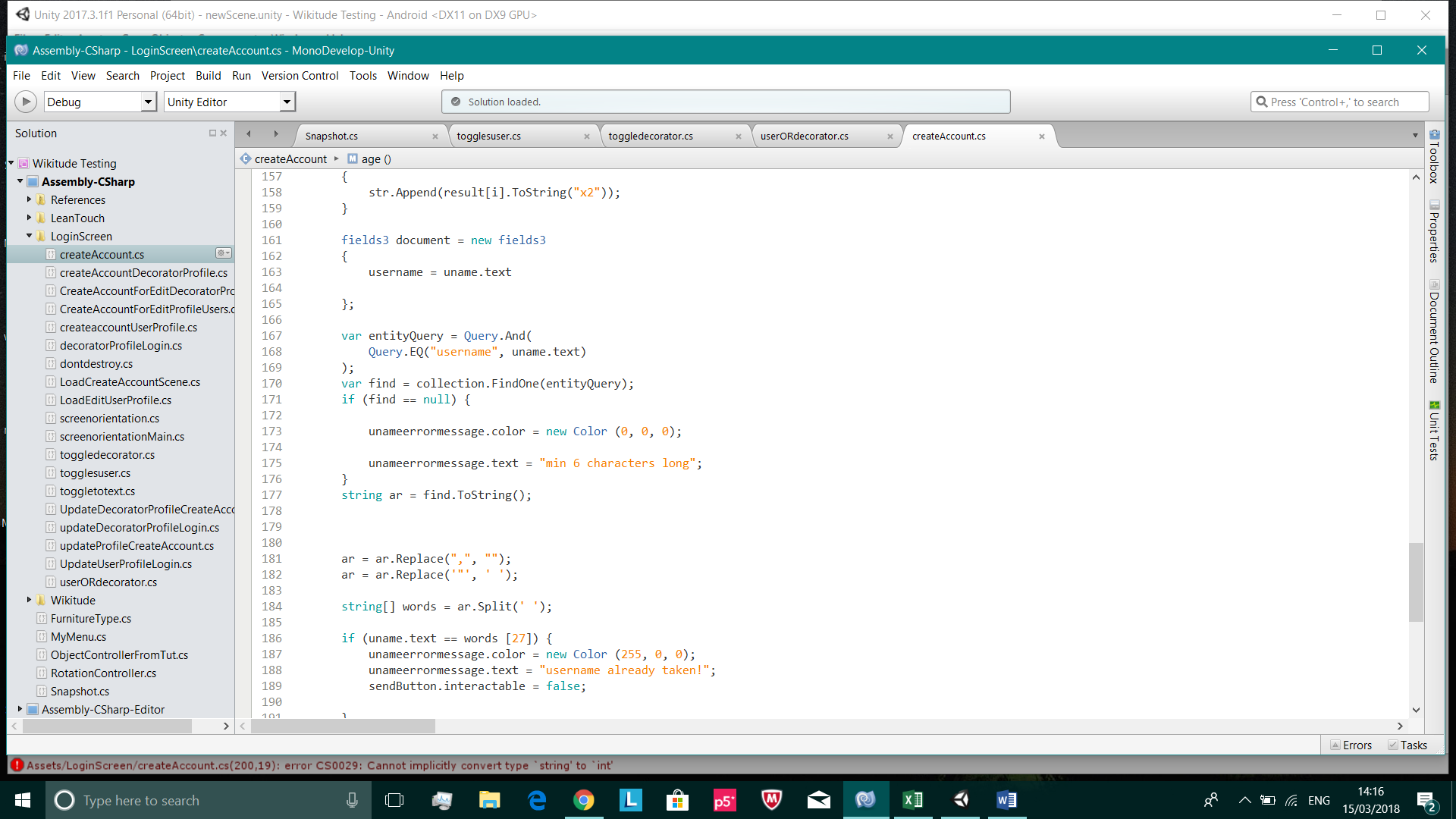
**Test 1**

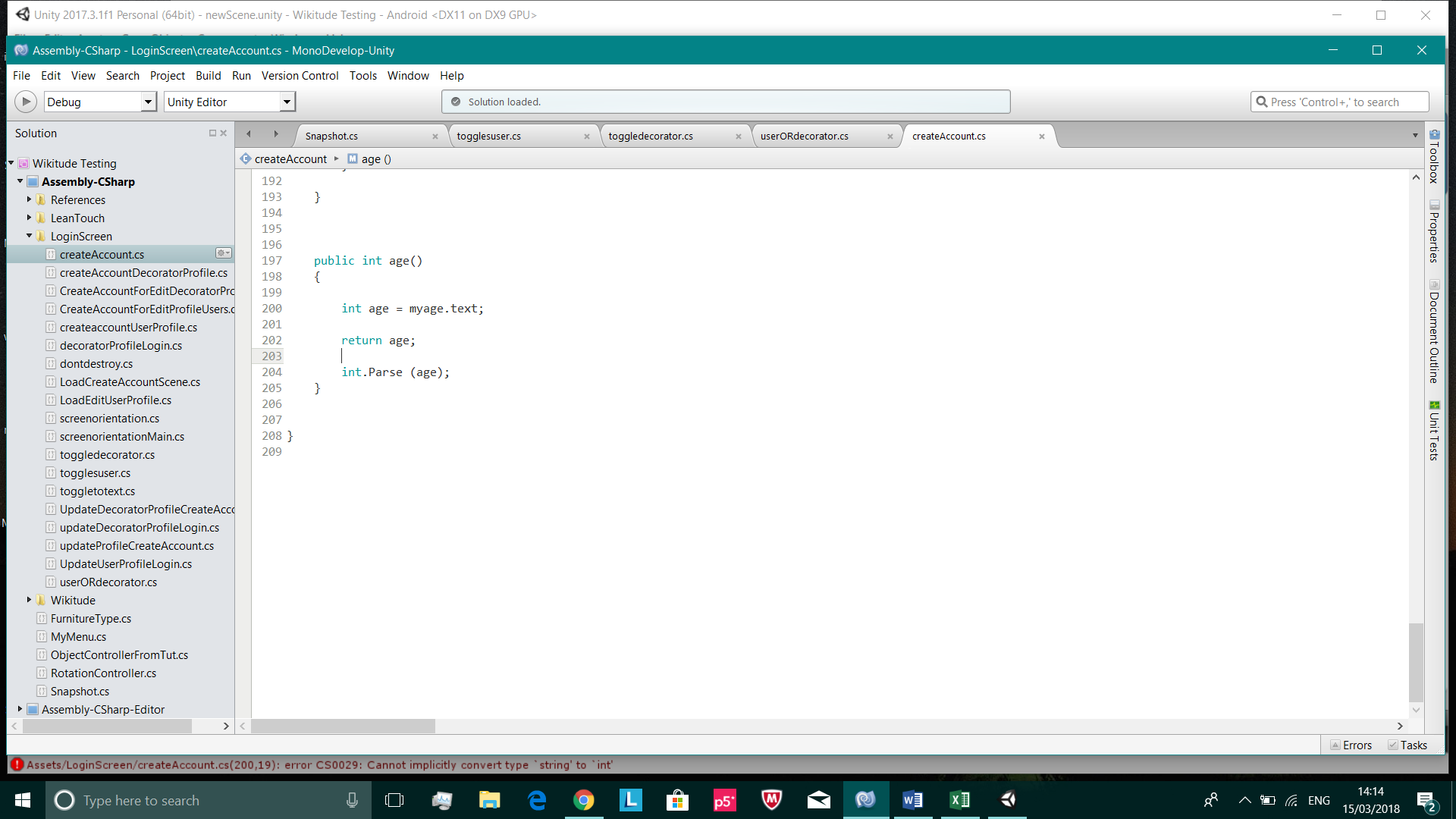
/

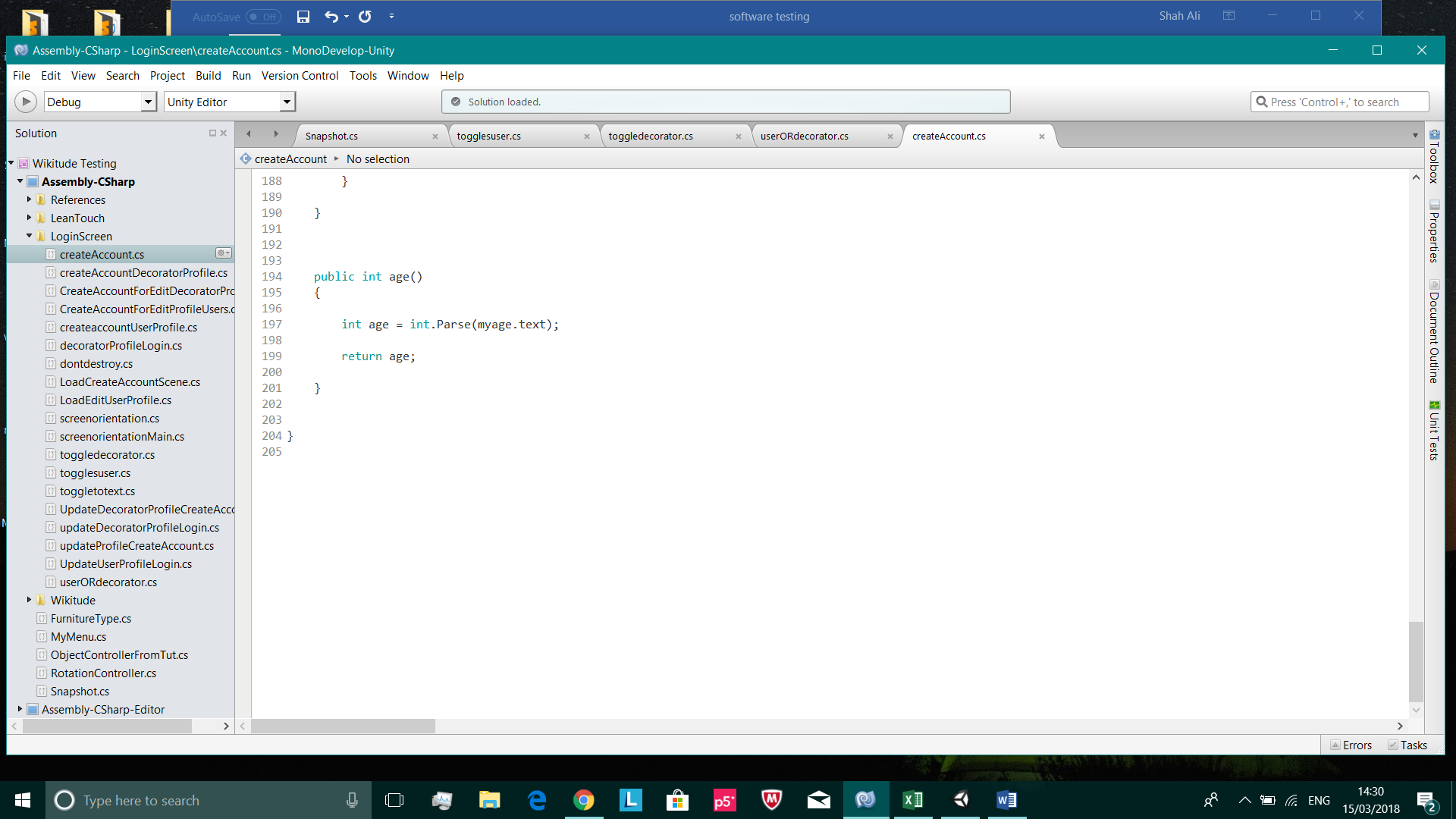
Unused/uninitialized variable

Variable removed

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







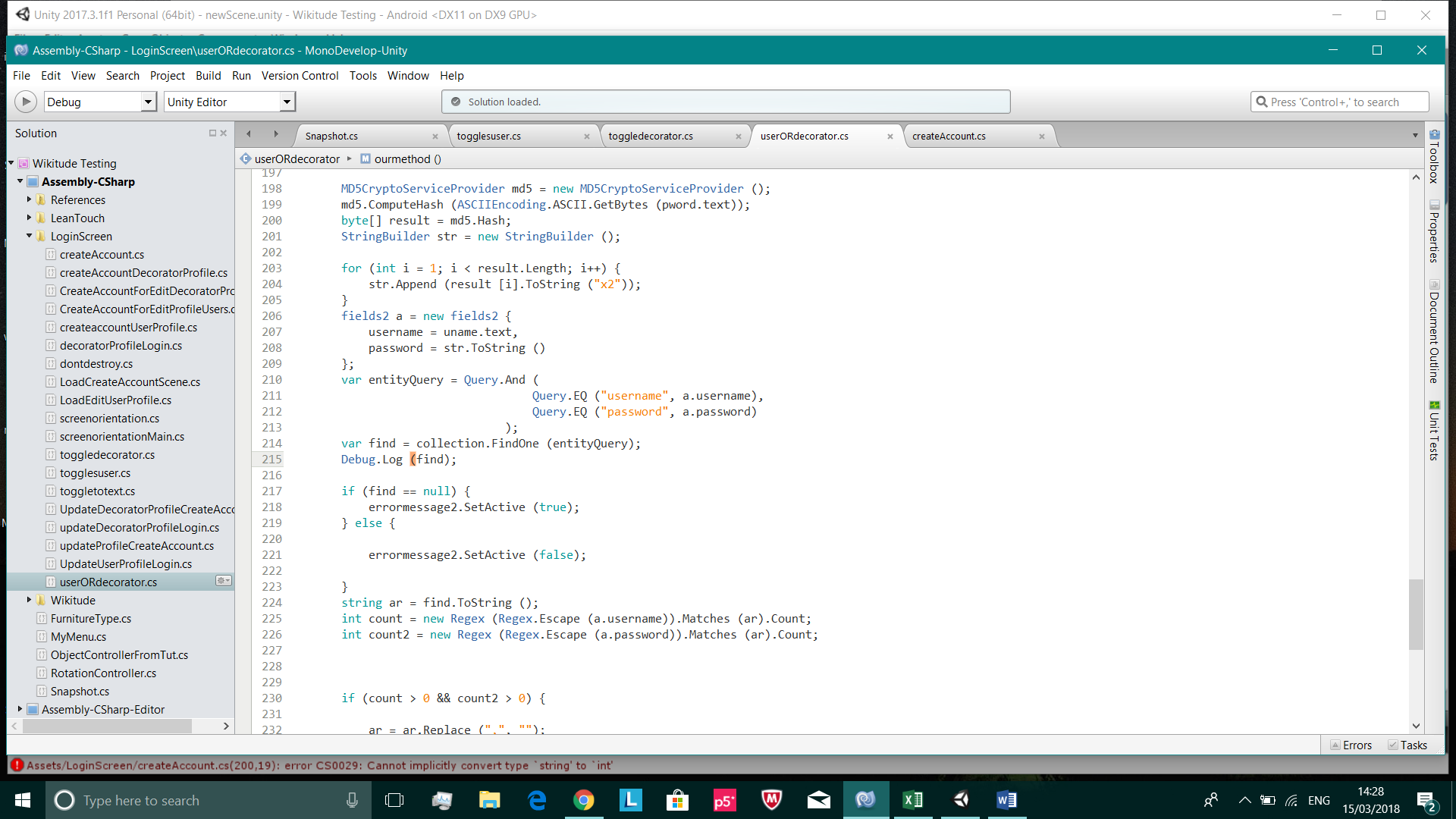
Correctly changed

/

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. I’ve tried the try and catch block to override the default behaviour of the program, but this didn’t really help

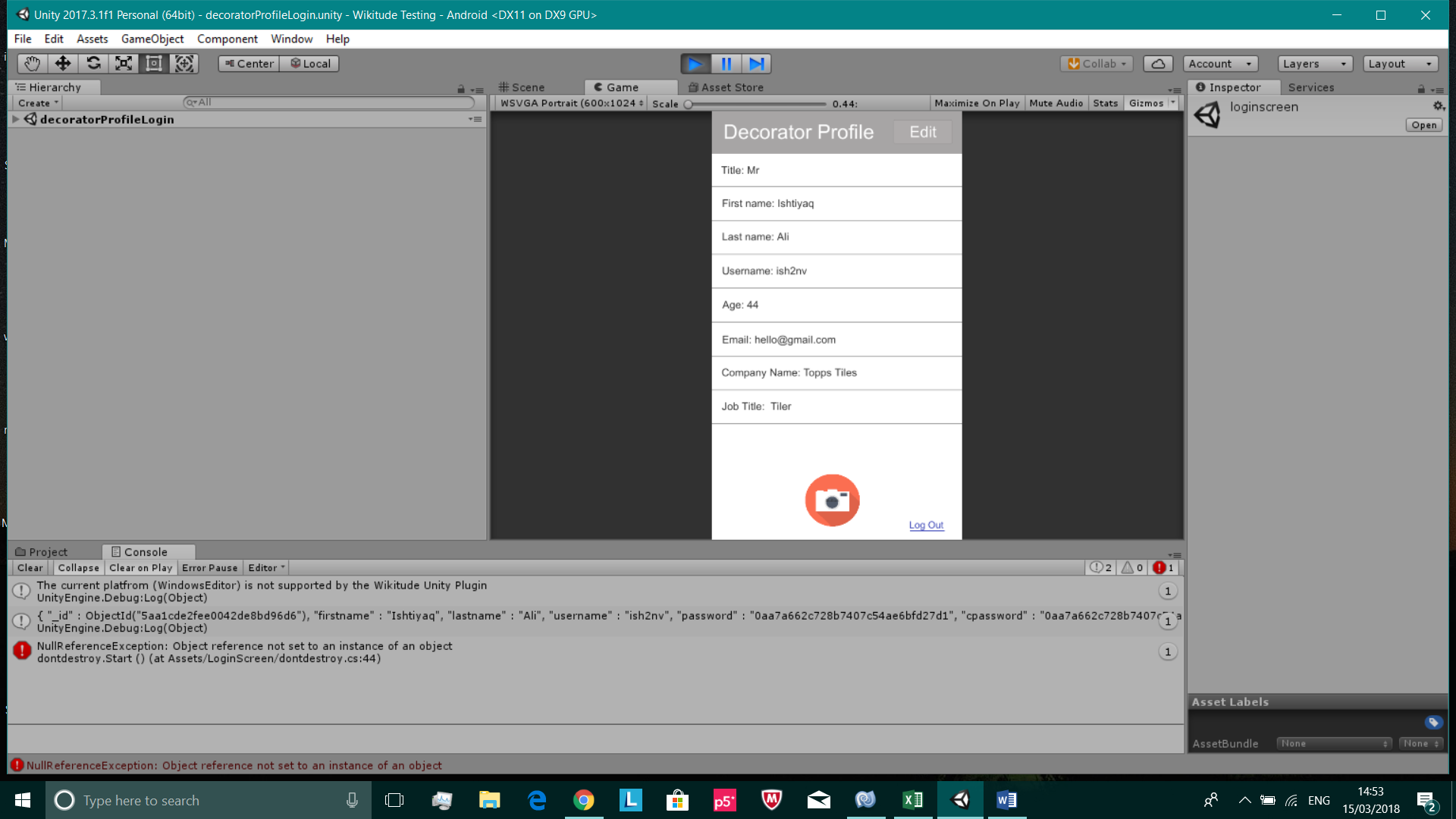


/

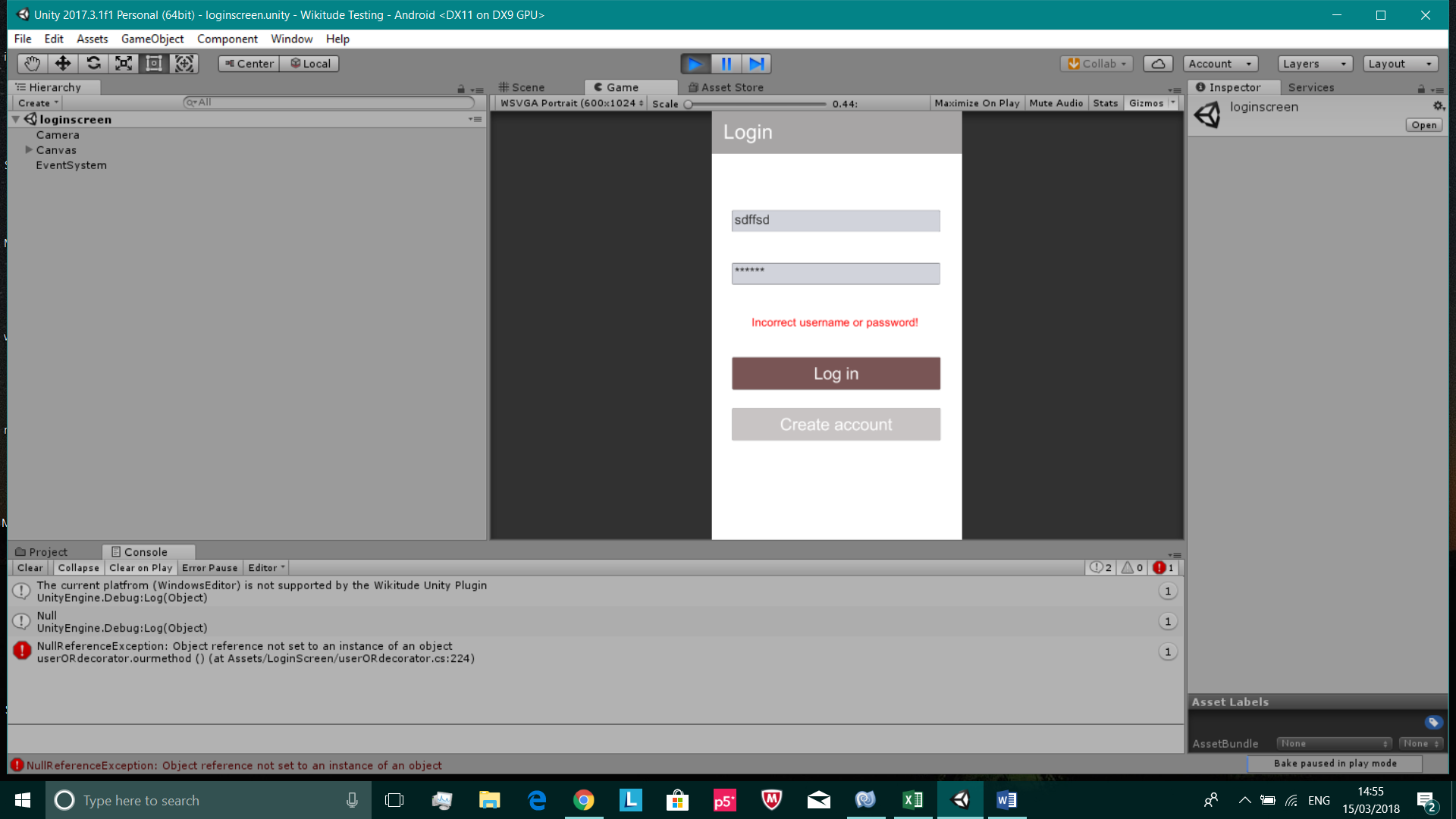
**Test 5**

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

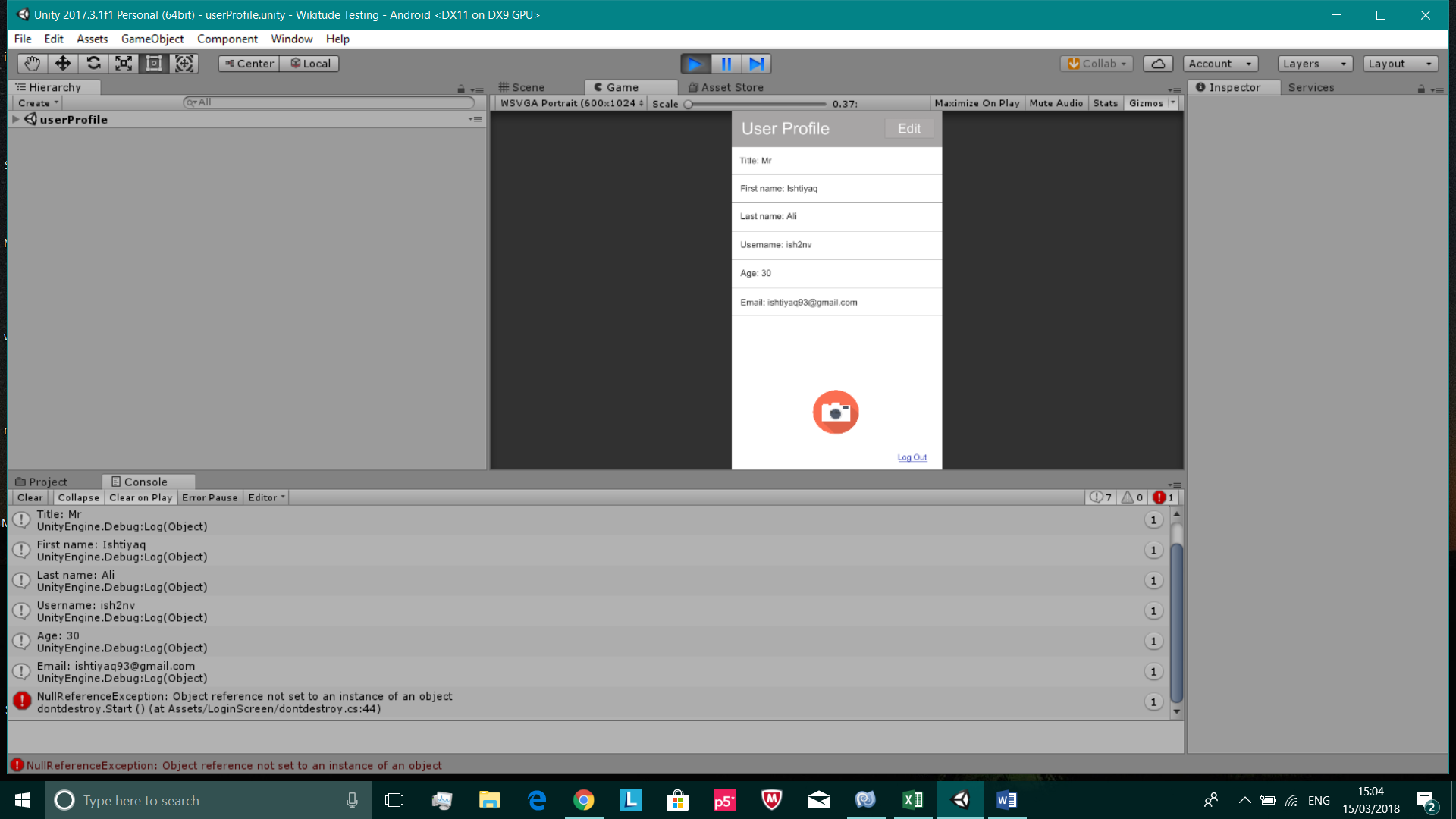
/

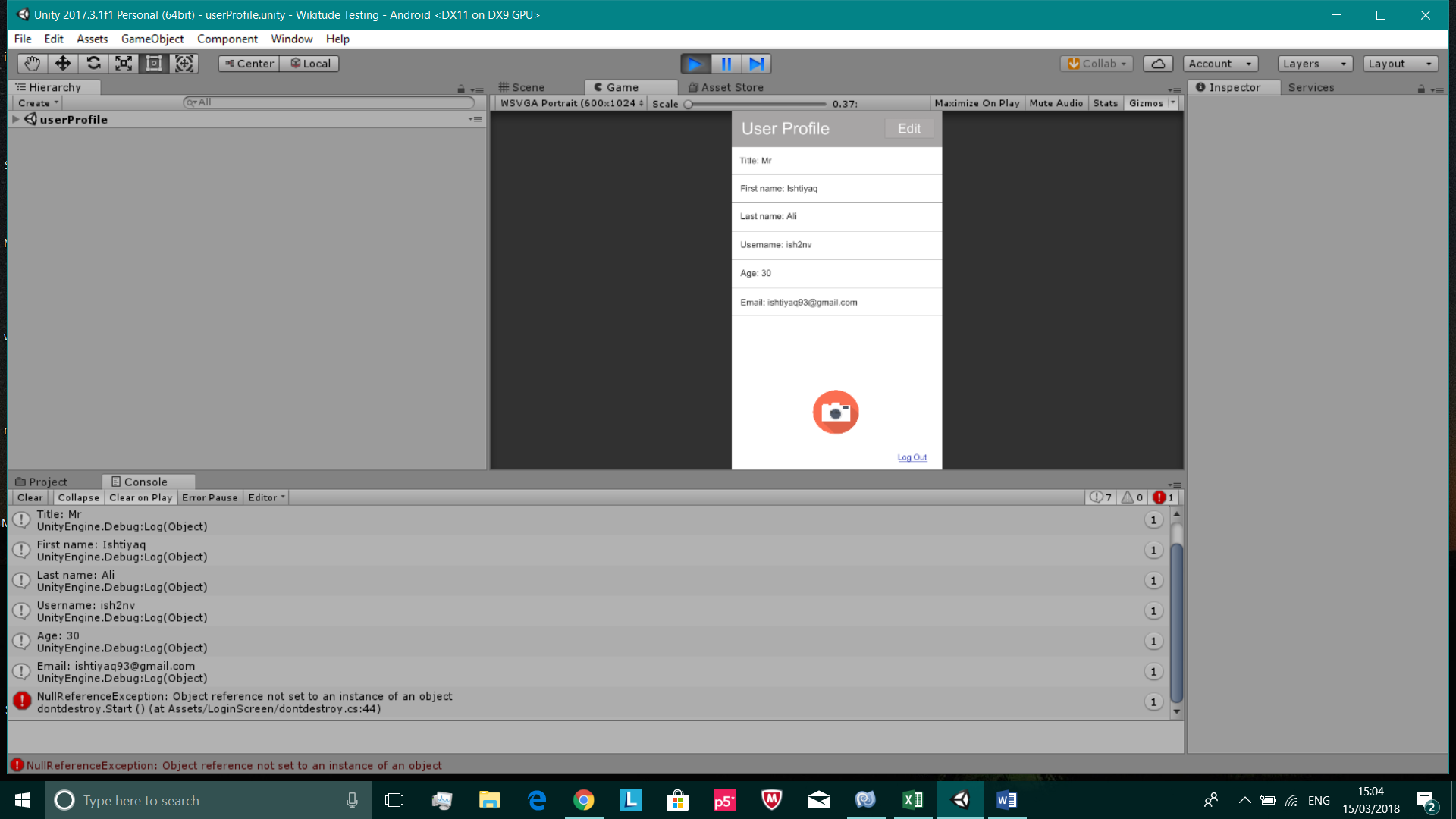
**Test 6**

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

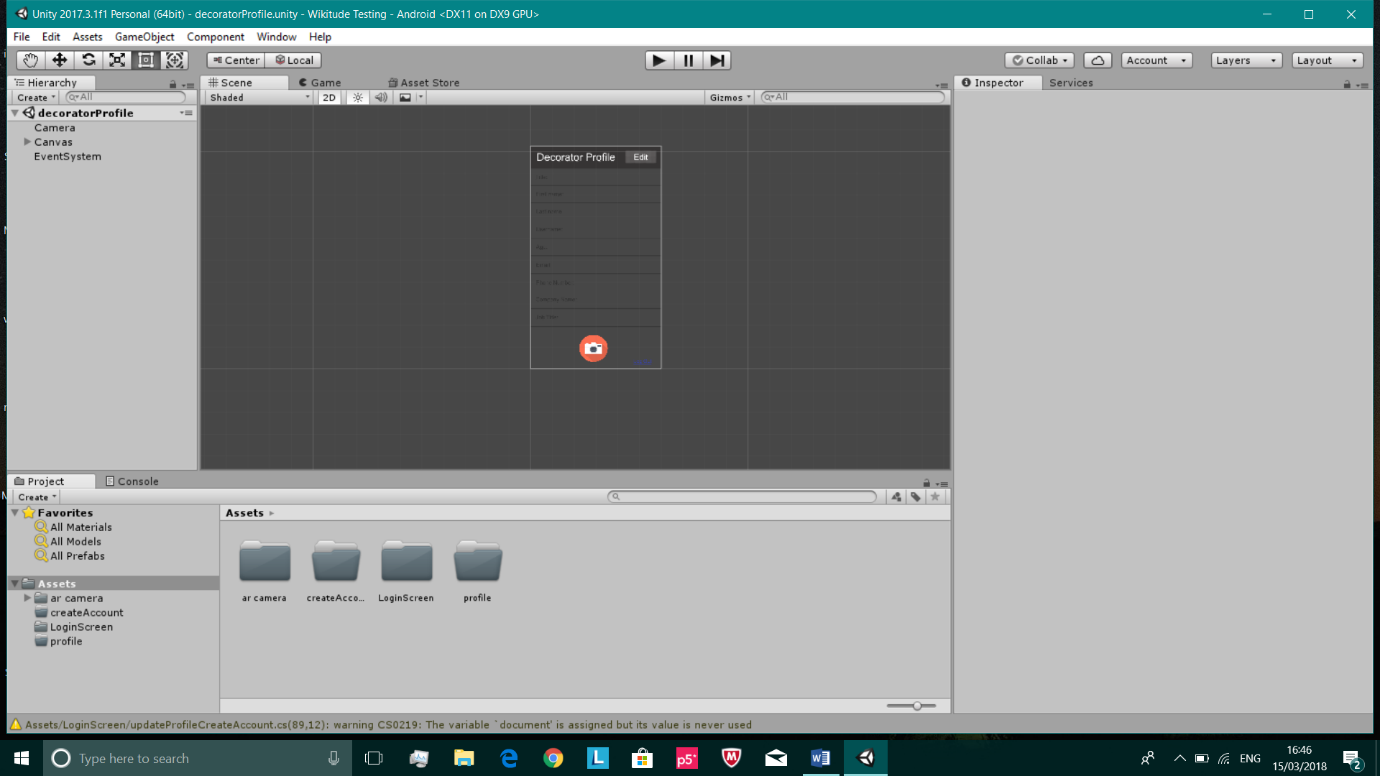


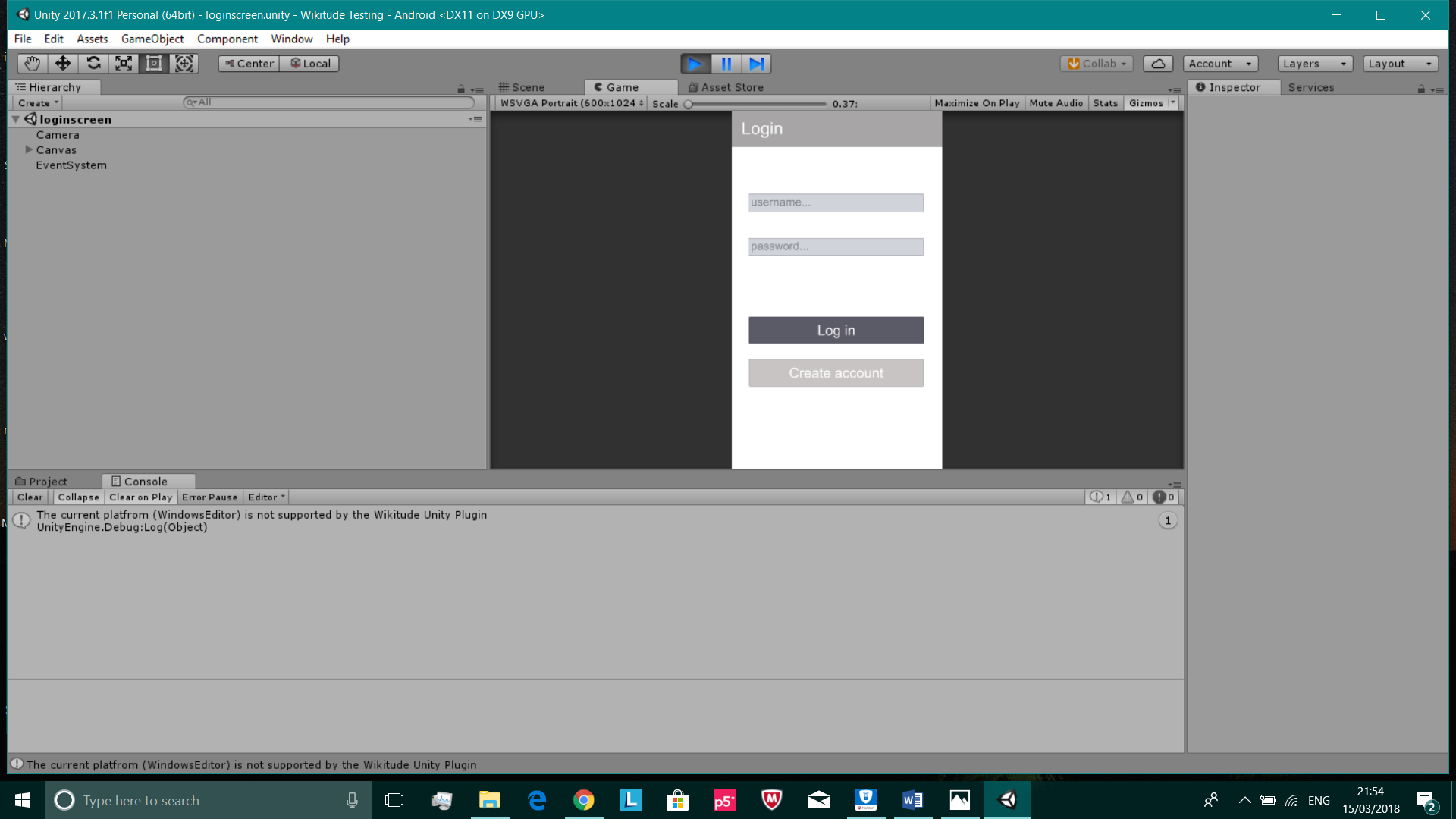
Unity console printing out null, since an invalid username & password was inputted in the login screen

**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