# 

# Contents

[**Contents**](#_qpx0hs4ayqw) **2**

[**Member Participation**](#_ncutfve46zq8) **3**

[**Login + Payment Details**](#_jzr5syrvhqll) **4**

[**Student Compliance with Academic Integrity**](#_dzs47d2ymvm0) **4**

[**Introduction**](#_g9t90xqtbamy) **6**

[**Iteration 1 Report**](#_8afffvh2lu05) **7**

[Market research](#_bdwvqzd0hy28) 8

[Test Cases](#_jf70sbircyw2) 9

[**Iteration 2 Report**](#_ihm1cjwadym6) **14**

[Test Cases](#_69npa2ypcjke) 15

[**Iteration 3 Report**](#_qbi7vqlfn1gm) **24**

[Test Cases](#_3bd6ak329szz) 25

[**Summary:**](#_djwb2y4kvtlh) **30**

[**Conclusion**](#_93anbxnhyyhf) **31**

[**References**](#_z5h847ethv35) **33**

# Member Participation

1. Piotr Wojciechowski
   * Sign Up
   * Log In
   * Merchandising
   * Emojis
   * User File uploads
   * CSS/Bootstrap/Javascript
2. Sean O Riordain
   * Mates
   * GroupChats
   * Blocking
   * Privacy
   * Profile
   * Change Password
   * Private Messaging
3. Aaron O Cuileann
   * Posts
   * Comments
   * Likes
   * Reports
   * Adding Product(Admin)
   * Notifications(through emails)

# Login + Payment Details

Log in details for users:

Username = admin password=1234

Username = testuser\* password = ‘Haddy1’

Paypal login:

Email: sb-47ht043685117@personal.example.com

password : IrishSoft123

Stripe login:

Email: test@gmail.com

Card Number: 4242 4242 4242 4242

Expiry date: 01/31

CVC: 111

\*= this covers tesuser1 to tesuser15

# 

# Student Compliance with Academic Integrity

All students are expected to complete their courses in compliance with University regulations and standards. No student shall engage in any activity that involves attempting to receive a grade by means other than honest effort, for example:

1. No student shall complete, in part or in total, any examination or assignment for another person.

2. No student shall knowingly allow any examination or assignment to be completed, in part or in total, for himself or herself by another person.

3. No student shall plagiarize or copy the work of another and submit it as his or her own work.

4. No student shall employ aids excluded by the instructor in undertaking course work.

5. No student shall knowingly procure, provide, or accept any materials that contain questions or answers to any examination or assignment to be given at a subsequent time.

6. No student shall procure or accept assignments from any other student from current or prior classes of this course.

7. No student shall provide their assignments, in part or in total, to any other student in current or future classes of this course.

8. No student shall submit substantially the same material in more than one course without prior authorization.

9. No student shall alter graded assignments or examinations and then resubmit them for regrading.

10. All programming code and documentation submitted for evaluation or existing inside the students computer accounts must be the students original work or material specifically authorized by the instructor.

11. Collaborating with other students to develop, complete or correct course work is limited to activities explicitly authorized by the instructor.

12. For all group assignments, each member of the group is responsible for the academic integrity of the entire submission.

**N.B to be filled out by each member of the team.**

By including my name in the form below, I declare that I understand and will abide by the University Regulations and Policies covering Academic Integrity. I accept that each member of the group is responsible for the academic integrity of the entire submission. I will retain a copy of this agreement for future reference.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name (1):** | **Sean O Riordain** | **Student No.:** | **x00154002** |
| **Student Name (2):** | **Aaron O Cuileann** | **Student No.:** | **x00155628** |
| **Student Name (3):** | **Piotr Wojciechowski** | **Student No.:** | **x00152561** |
| **Module title:** | **Year 2 Project** | | |
| **Programme Title:** | **Computing with Software Development** | | |
| **Date: DD/MM/YYYY** | **27/04/2020** |  | |

# Introduction

For our second year project we decided to do a Social Media Website named Loop. We landed on this idea when we were throwing ideas around for what we could do for our project. We wanted to do something we thought would challenge us but would also be doable in the time frame given.

We found that every iteration threw a new challenge our way and that our project was no easy task, leading to many arguments and long sessions staring at code trying to figure out why it wouldn’t work how we wanted. We took advantage of practices such as paired programming to get our project done. We also took advantage of software like Zoom to get our work done as we spent the last iteration in quarantine.

We used market research and advice from friends to help shape the User Interface of our website so that it would be accessible and pleasing to the eye. We settled on a white website with purple accents. For testing we implemented test cases and selenium IDE for each feature we could.

# Iteration 1 Report

The approach we took this iteration was the Agile methodology mainly the SCRUM process framework. We used this methodology because we found it made sense with the iteration setup the project already had, also a big part of SCRUM is adaptation, we found this very useful as the design we had needed changed when we started coding. We started the semester off spending a couple of hours in room planning, designing the website and mapping out our iterations. We used value-based prioritization to map out our iterations and everything we thought was essential to the website was placed in iteration 1. We used a survey to help design our website and got 49 responses. This helped us decide some key factors like layout [fig 1], which social media was most popular so we could look at them for inspiration [fig 2] and what demographic to gear our website towards [fig 3, 4]. In addition to having a meeting at the start of the semester we also had near daily stand up scrum meetings. We found this was vital to working this iteration as it meant we were constantly striving towards a common goal and no one strayed from the iteration plan. We utilised the agile technique Paired programming as it meant we could catch problems before testing and verify efficiency and streamline problem solving. If we were not using the paired programming technique, we usually tried working together in the lab and not working separately as often as we possibly could to keep maximum collaboration. On the occasions we were working separately at home we sent constant updates to each other as we found communicating was key.

From our survey we gathered that the majority of people prefer a more clean and easy to navigate design in the form of Instagram or Snapchat [fig 5]. We gave links to different parts of our website icons so it would be easily identifiable and users would understand where the link would take them. It also allows for more veteran users to quickly move around the website in a short amount of time. Another design aspect we decided on was to make sure that a user could see the most important features on their dashboard like their friends, user options, search bar etc. The last major design choice in our first iteration was to try and remove as many html pages as possible, a good example of this is the post button on the main page. Instead of having a user click the button which would redirect them to a form post where they input whatever they want to share and then redirecting them back to their dashboard where they could see their newly created posts we opted to remove the middleman and have the form on the same page. The purpose of these design choices is to ensure that users won’t get quickly frustrated with a bad UI and have to spend too much time trying to achieve a simple goal.

Instead of using Django’s default auth.user, we opted to make our own custom user for the benefit of being able to add in the functionality of the user being above the age of 13 years old. Doing so would stop the user registering a user under the age of 13 and the world redirect them back to the registration form to change their date of birth to a higher age. Now having the custom user implemented we could now implement posts, profiles, the search functionality, likes and in the end mates. We found that profiles were the hardest part of the iteration to get through in the end. We had trouble trying to figure out how to link the user to their own profiles because there were multiple variables to take into consideration when we got to this point.

We have used Selenium as the frame for the testing of our social media website. We have done tests on Sign-in, Sign-up, Log-out, Likes, Make Posts, Delete Posts, Edit Posts,Create Profile, Update Profile, Lose Mates and Add Mates. We did these tests multiple times for each function to make sure they all functioned properly as we had expected them too.

## Market research

Fig 1

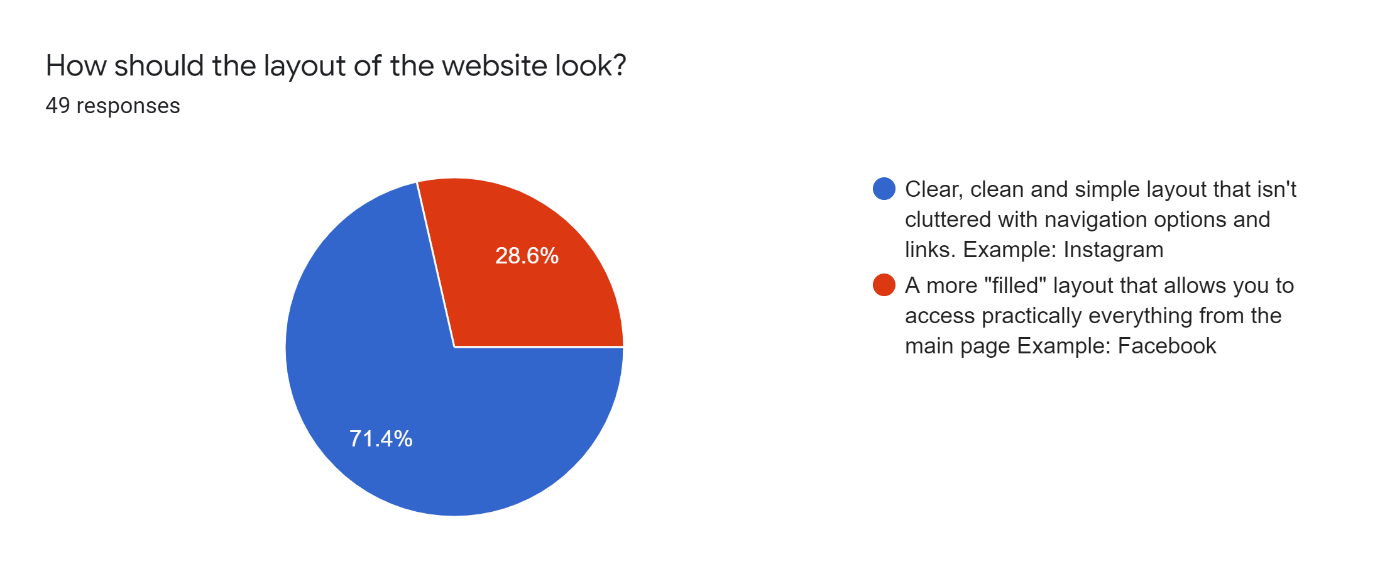


Fig 2

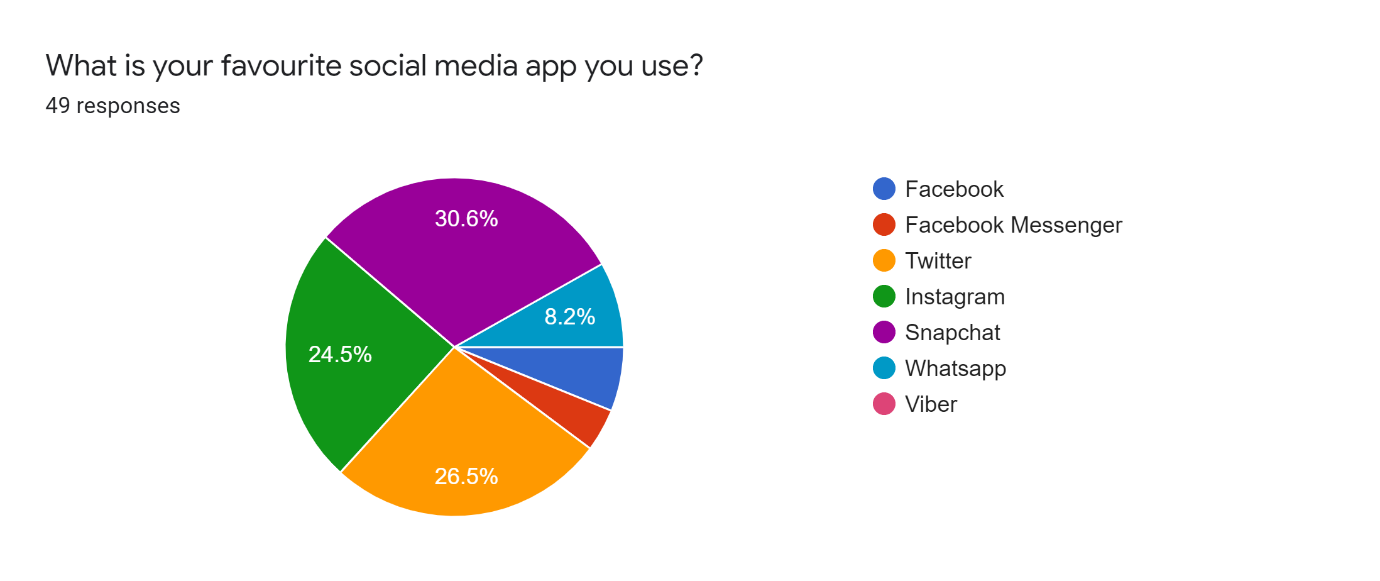


Fig 3

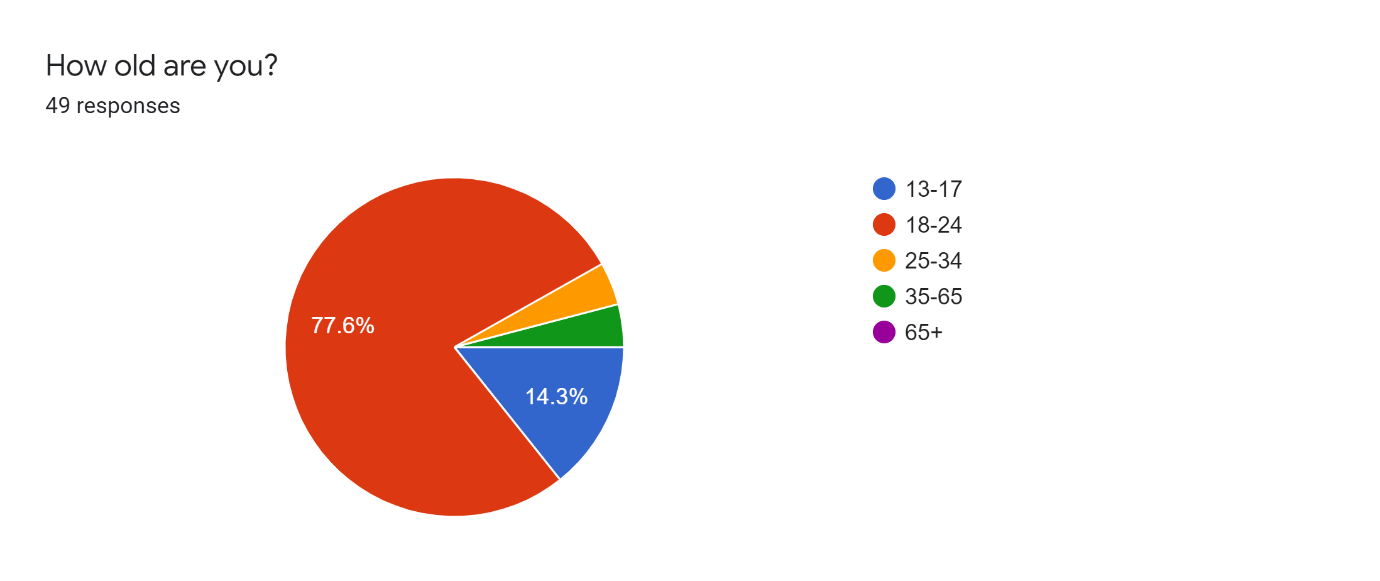


Fig 4

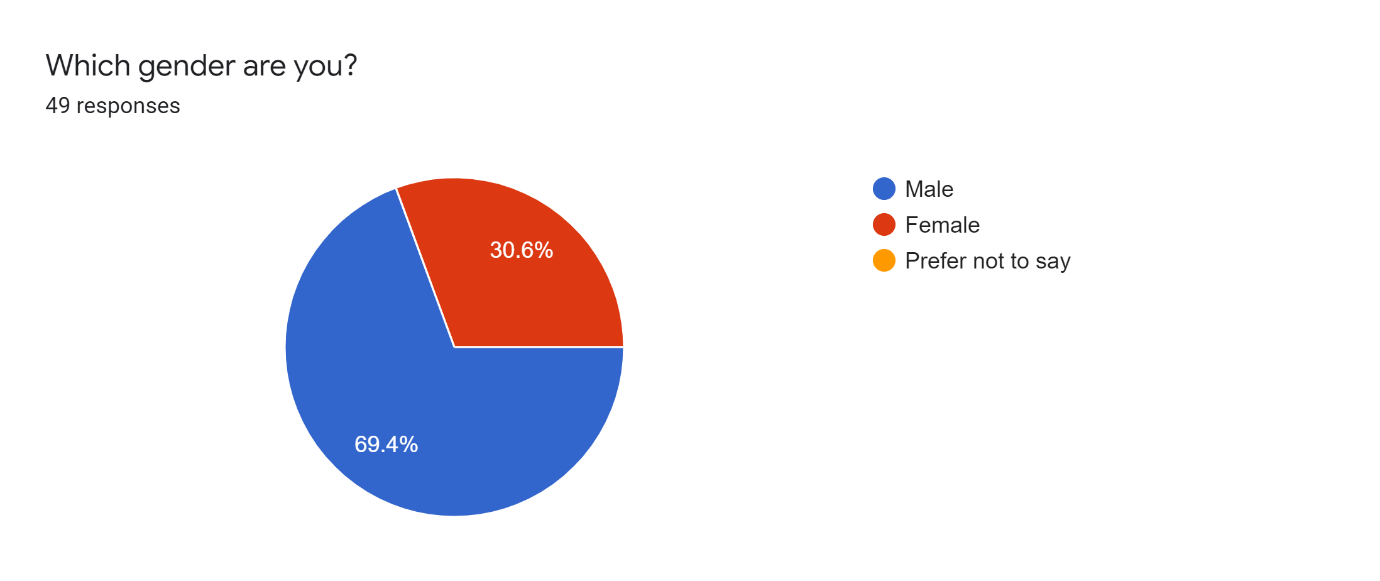
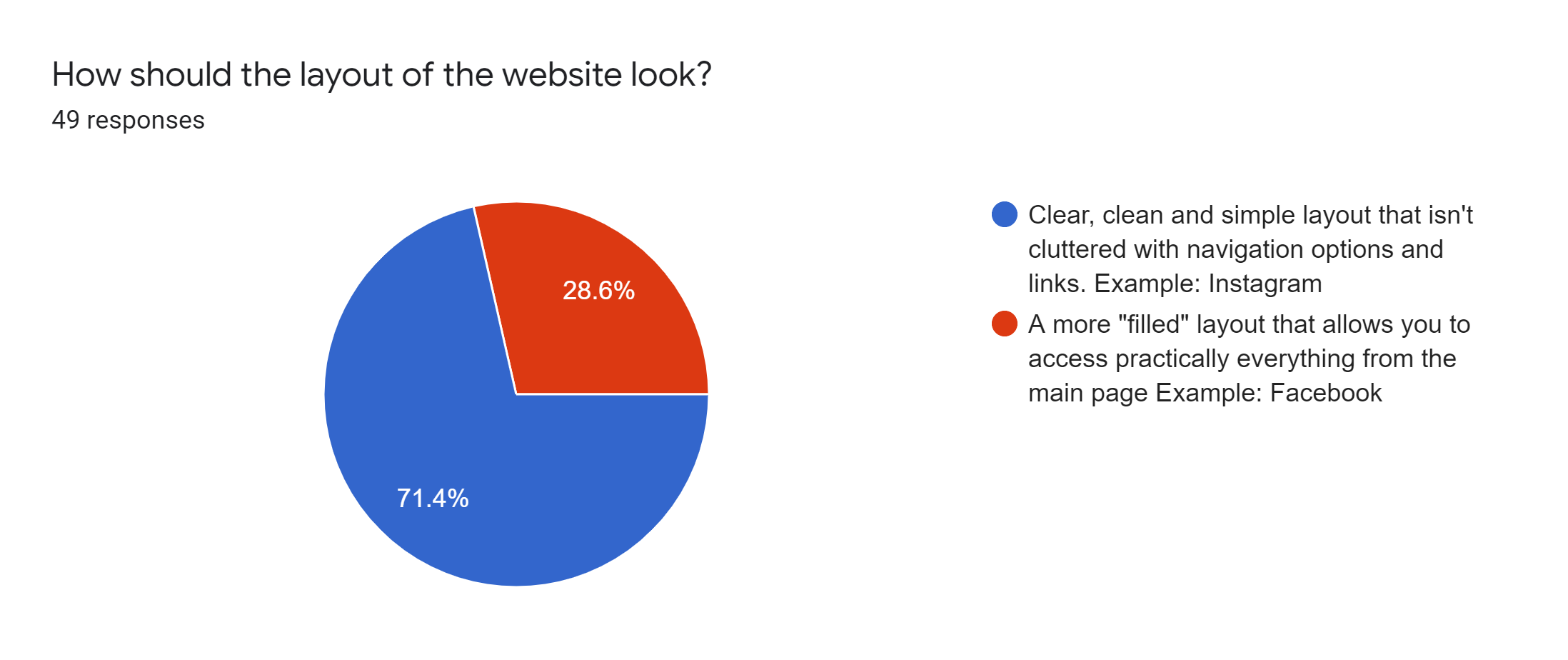
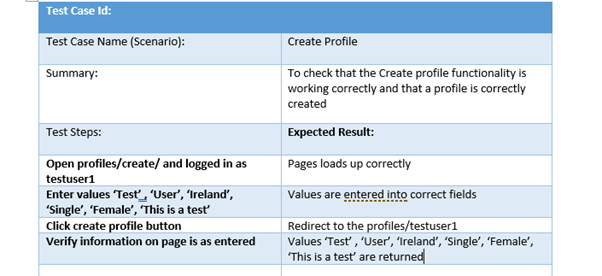
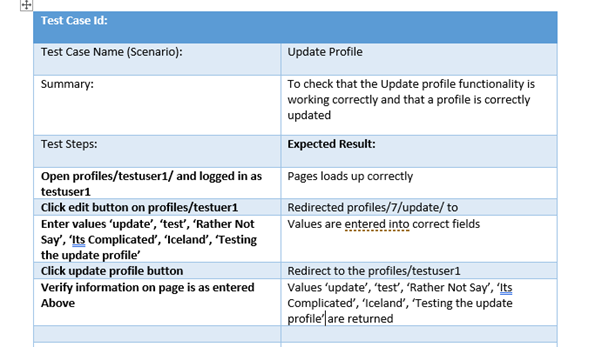


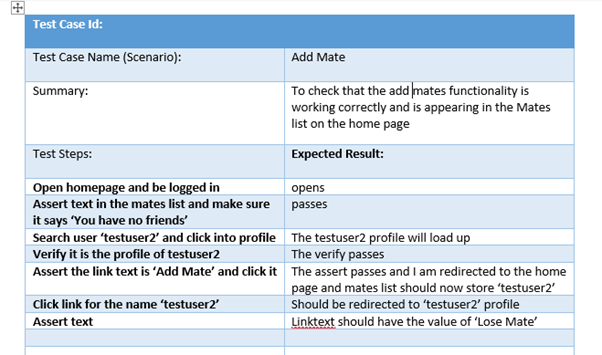
Fig 5

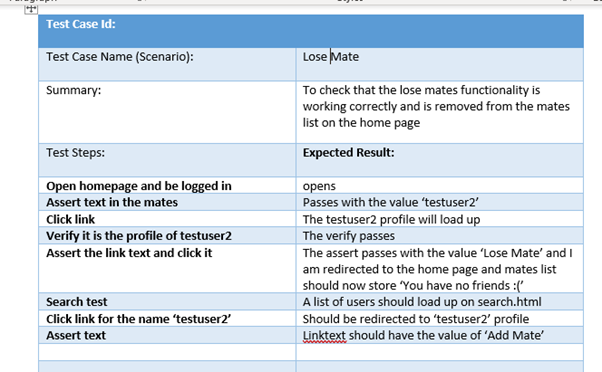


## Test Cases









|  |  |
| --- | --- |
| **Test Case Id: 0005** |  |
| Test Case Name (Scenario): | Sign Up |
| Summary: | To check that the user is successfully registered and redirected to the home page |
| Test Steps: | **Expected Result:** |
| **Open** [**http://127.0.0.1:8000/signup/**](http://127.0.0.1:8000/signup/) | Page launches successfully |
| **Select username field and enter the username of the new user (testuser99)** | Box should highlight blue to indicate it is selected and text should appear as typed |
| **Select date of birth and make sure it is above the age of 13(02/02/1999)** | Box should highlight blue to indicate it is selected and text should appear as typed |
| **Select the email field, enter the email**  **address of the new user(testUser**  **@gmail.com** | Box should highlight blue to indicate it is selected and text should appear as typed |
| **Select the password field, enter the password of the new user(test1)** | Box should highlight blue to indicate it is selected and text should appear as typed |
| **Select the confirm password field, re-enter the same password of the new user(test1)** | Box should highlight blue to indicate it is selected and text should appear as typed |
| **Select and click the register account button.** | Page redirects to users dashboard on  [http://127.0.0.1:8000](http://127.0.0.1:8000/signup/) |

|  |  |
| --- | --- |
| **Test Case Id: 0006** |  |
| Test Case Name (Scenario): | Sign in |
| Summary: | A registered user should be able to successfully login at<http://127.0.0.1:8000/signin/>  Precondition:  The user must already be registered with an email address & password. |
| Test Steps:  **Open** [**http://127.0.0.1:8000/signin/**](http://127.0.0.1:8000/signin/) | **Expected Result:**  **Page launches successfully** |
| **Select the ‘username’ field, enter the username of the registered user (testuser1)** |  |
| **Enter the password of the registered user (test1)** |  |
| **Select and click Sign in button** | Page redirects to users dashboard on [http://127.0.0.1:8000](http://127.0.0.1:8000/signup/) |

# 

|  |  |
| --- | --- |
| **Test Case Id: 0007** |  |
| Test Case Name (Scenario): | Log Out |
| Summary: | A signed in user should be able to click the logout button inside menu bars icon and successfully logout |
| Test Steps: | **Expected Result:** |
| **Open**[http://127.0.0.1:8000](http://127.0.0.1:8000/signup/) **and be signed in** | Page launches successfully |
| **User navigates to the menu bars icon in the top right and clicks on the button** | A list of options for the user should dropdown |
| **User clicks logout** | User is redirect to the sign in page and is now logged out successfully |

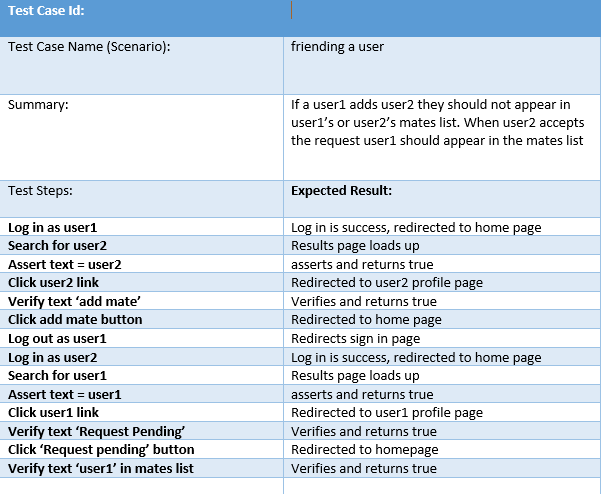
# Iteration 2 Report

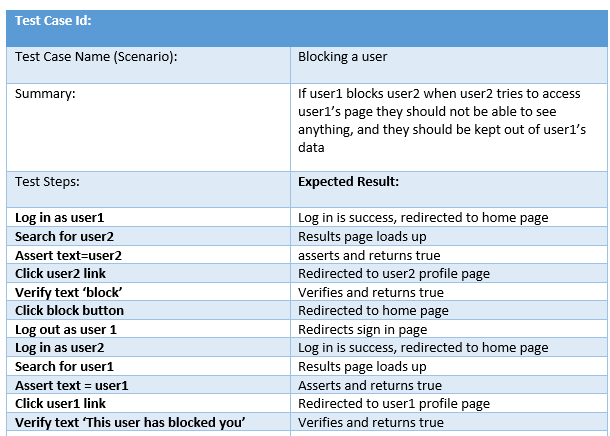
For this second iteration of our social media website, we have had multiple new functions added. The functions that have been added are Commenting, Messaging, being able to block other users, and merchandising. In this iteration a lot of the new functions were expansions of functions from the previous iteration. We have used the custom user model to allow us to message between yourself (the sender) and another user (the recipient) which are private to the both users. This works in both ways as the other user can be the sender and yourself as the recipient. We have also added in image sharing, which allows you to send images as well as text to another user. We found this iteration to be a bit more difficult than the last iteration due to the circumstances of not being in the college and being able to meet up in person for SCRUM meetings weekly. It was challenging at first but as the weeks went by we managed to get more relaxed with the situation, had SCRUM meetings weekly online, and most importantly, we kept each other up to date on what we were doing throughout the project iteration. In all, we believed this iteration to be successful in what we wanted to achieve even with the restrictions we faced.

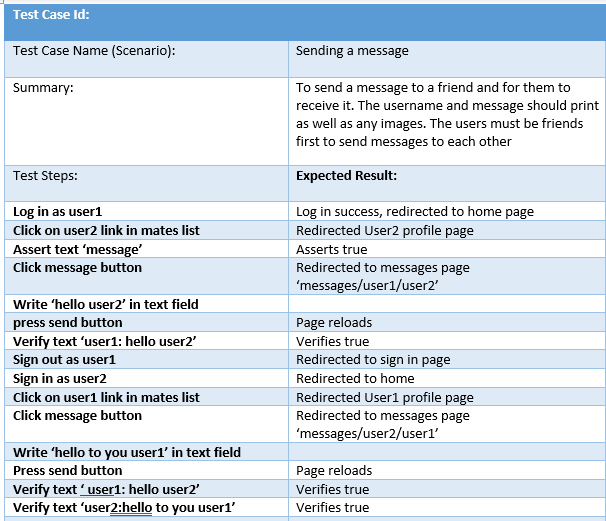
We had the task of fixing friends this iteration so that a user who is in your friends list has to accept your request for both of you to be friends instead of them appearing the instant you add them. This proved to be a challenge as trying to write a way to accept friend requests would involve too much database fiddling and could break what's already been established. In the end we went with the easiest solution but also the least robust and that was just comparing the two lists of friends that a user had. If a user appeared in someone else's list and that person was in their list it would be counted as them being friends.This does lead to a bug where a users post appears on another persons homepage after sending them a friend request and before they are both friends Another issue we had was getting messages working as django's models gave back issues about primary keys not being initialised. While messaging does work we feel there needs to be more added in terms of security. At the moment an admin can see all the messages and can even send messages from another user's account. We do not want this to continue and hope we can fix it in the next iteration.

Before our implementation of shop functionality we discussed how we could tie in such a feature with our main goal of creating a social media website. We couldn’t just add some e-commerce storefront that had nothing to do with Loop. At first we thought about designing a marketplace similar to what Facebook does where the users aren’t just buyers but sellers as well. It seemed as if this idea would just be an extension of our posting functionality but the more time we spent discussing it the more it became difficult to see it realised in our project time constraint. In the end we decided on a simpler implementation which was merchandising. We agreed that if the project was put into a live scenario, merchandising would allow the company to gain income and continue to grow. In merchandising we made it possible to filter our products, produce a different “add to cart form” depending on the product selected for example if a user selected “beanie” then they would be presented with the option of choosing quantity and colour. On top of that we implemented order creation, payment using either Stripe or Paypal and we added Order History so users could see past purchases. A few bugs and problems exist which we hope to fix in the next iteration. For example a user can decide not to pay for the item, the order item will remain in a state of limbo. Another bug identified is when looking through product images if an image is selected and then the arrow is pressed to get the next image it removes the image. Overall our main iteration goals have been successfully implemented and will be refined in the final iteration.

## Test Cases







|  |  |
| --- | --- |
| **Test Case Id: 004** |  |
| Test Case Name (Scenario): | Adding product to Cart |
| Summary: | A user should be able to click a product, choose the options presented in the form in the product detail page and press the add to cart button to successfully add a product to the users cart. |
| Test Steps: | **Expected Result:** |
| **Open** [**http://127.0.0.1:8000/products**](http://127.0.0.1:8000/products/) | Page launches successfully |
| **Hover over the “Flip Flops” product and clicks the link** | Page redirects to <http://127.0.0.1:8000/shop/7/flip-flops/> |
| **Select the ‘quantity’ field and select (1)** |  |
| **Select and click the Add to Cart button** | User is redirected to their shopping cart and can see their product they added to cart, quantity selected, and the cost of the product. |

|  |  |
| --- | --- |
| **Test Case Id: 005** |  |
| Test Case Name (Scenario): | Using the filter menu on the products page |
| Summary: | A user should be able to fill out any filter options on the product list page and be presented with a list of products with the query matched |
| Test Steps: | **Expected Result:** |
| **Open** [**http://127.0.0.1:8000/products**](http://127.0.0.1:8000/products/) | Page launches successfully |
| **Select the “Category” field and selects Clothes option** |  |
| **Select the “Price is greater than” field and inputs “9”** |  |
| **Select the “Price if lower than” field and inputs “20”** |  |
| **Select and click the Submit button** | User is redirected to the products page with the query options entered. |

|  |  |
| --- | --- |
| **Test Case Id: 006** |  |
| Test Case Name (Scenario): | Purchase product using stripe |
| Summary: | Click checkout after adding an item to cart, filling out your order information and then using stripe to purchase the product  Assumption: An item is in your cart |
| Test Steps: | **Expected Result:** |
| **Open** [**http://127.0.0.1:8000/cart/products/**](http://127.0.0.1:8000/cart/products/) | Page launches successfully |
| **Select and click the Add product button** | User is redirected to the admin add product page |
| **Select the “Category” field, choose which category to use(Pillow)** |  |
| **Select the “Name” field, enter the name of the product (Loop Pillow)** |  |
| **Select the “slug” field, enter the sluo (piotr87654@gmail.com)** |  |
| **Select the “Address Line 1” field, enter the address of the user (228 Balrothery Estate)** |  |
| **Select the “Postal Code” field, enter the postal of the user (D24 CKP4)** |  |
| **Select the “City” field, enter the city name of the user (Tallaght)** |  |
| **Select the “County” field, enter the county name of the user (Co. Dublin)** |  |
| **Select the “Country” field, select the country name of the user (Ireland)** |  |
| **Select and click the “Pay now” button** | User is redirected to the order payment page |
| **Select and click the “Pay with Card” button** | A pop up window is displayed |
| **Select the “Email” field, enter the email address of the user (piotr87654@gmail.com)** |  |
| **Select the “Card Number” field, enter the card number (4242 4242 4242 4242)** |  |
| **Select the “MM/YY” field and enter the expiration date of card (01/2030)** |  |
| **Select the “CVC” field and enter the cvc number of card (111)** |  |
| **Select and click the “Pay” button** | User is redirected to the payment made page saying that that the payment was successful |

|  |  |
| --- | --- |
| **Test Case Id:007** |  |
| Test Case Name (Scenario): | Creating a Comment |
| Summary: | Allowing a user to be able to create a comment on a post that has been created. When created the comment should display in the post. |
| Test Steps: | **Expected Result:** |
| **Log in as user1** | Log in successful, redirect to home page |
| **Click on comment text field** | Awaits input |
| **Input comment** | Comment now in text field |
| **Click comment button** | Redirected to homepage with comment in post |
| **Verify text ‘user1’ on comment** | Verifies and returns true |

|  |  |
| --- | --- |
| **Test Case Id:008** |  |
| Test Case Name (Scenario): | Deleting Comment |
| Summary: | Allowing the user that created their comment to be able to delete the comment they have created. |
| Test Steps: | **Expected Result:** |
| **Log in as user1** | Log in successful, redirect to home page |
| **Click on settings option button** | The buttons Edit and Delete pop up in the comment |
| **Click on the delete button** | Redirected to delete comment page ‘delete\_comment/(comment number)’ |
| **Click the delete button for comment** | Redirected to the home page, comment is now deleted |

|  |  |
| --- | --- |
| **Test Case Id:009** |  |
| Test Case Name (Scenario): | Edit Comment |
| Summary: | Allowing the user that created their comment to be able to delete the comment they have created. |
| Test Steps: | **Expected Result:** |
| **Log in as user1** | Log in successful, redirect to home page |
| **Click on settings option button** | The buttons Edit and Delete pop up in the comment |
| **Click on the edit button** | Redirected to edit comment page ‘edit\_comment/(comment number)’ |
| **Click on the edit text field** | User can alter the comment |
| **Click the edit button** | You are redirected to the homepage with the comment now altered. |

# Iteration 3 Report

For the third and last iteration of our social media website, we wanted to focus on getting small features that we didn’t get done in the last development cycle or expanding on features that already existed. Apart from that we wanted to make sure all major bugs were fixed and making the site look as professional as possible with the help of CSS, Bootstrap and JavaScript. Due to the pandemic over the past few months, we weren’t allowed to meet up in person like we did before which ruined our motivation throughout the iteration but realising the opportunity of having more time on our hands we decided to persevere and refine our website as much as possible and continue our paired programming technique adopted at the start of project by using video/voice calls. We focused on getting multiple files to be uploaded by the user such as images or videos when making a post. Another thing we focused on is making the user experience as smooth as possible since that’s what our participants in the questioner wanted before we started development. We made things like the buttons more visible, the amount of likes a post more cleaner, comments to slide in slowly when a user clicked on the amount of comments. Apart from that we made the page have less white by adding gray sides to the site so it wouldn’t feel as empty and boring. We moved the navigation buttons to the header instead of the sides since we got feedback from friends that it looked weird and almost out of place. All in all we believe we achieved a more friendly user experience that any user could use.

We focused on expanding the messaging system by implementing group chat functionality.

We made sure that you can only create a group chat when you have mates.

We are also adding in little features such as changing password and delete profile to create a more user friendly experience. We now send emails to users throughout the project to ensure they are being notified about what is happening with their profile while they are away from their account.

In this iteration admin functionality was added in. We decided that being able to add a product from the website rather than in the admin site would make it look a lot smoother as we went through the project and it's easier to access. We add an add product button in the product list page, but this button is only visible to users that are admins. If you aren't an admin you won't see the button. Because the products use inlines in the product detail page for scrolling through the images, we had to add in inline formsets to the view which was one of the difficult parts of this iteration as it was coming close to the end of the project, but in the end we managed to get it to work. Reports were also added in this iteration which allows users to report posts that they feel aren't friendly to the website and shouldnt be seen. The admin can then go through the reports and then then decide if the post should be deleted or not. It wasn’t implemented exactly as we would have liked as we wanted the admin to be redirected to the delete post html but due to having CustomUsers and it getting closer to the deadline we had to redirect to the adminsite.

In terms of testing we decided to stick with Selenium IDE tests as we find them intuitive and easy to operate.

## Test Cases

|  |  |
| --- | --- |
| **Test Case Id: 001** |  |
| Test Case Name (Scenario): | Uploading an image |
| Summary: | A user should be able to upload an image when making a post, which will display on his dashboard. |
| Test Steps: | **Expected Result:** |
| **Start Papercut server** | Papercut application opens |
| **Open** [**http://127.0.0.1:8000/**](http://127.0.0.1:8000/products/) | Page launches successfully |
| **Select the text area field with “Write a post…” in it and enter “test”** |  |
| **Select the “Choose files” on the page and select (default.jpg) from your files folder.** | Text next to choose files should say “default.jpg” |
| **Click the “Post” button on the right** | Page redirects to users dashboard on  [http://127.0.0.1:8000](http://127.0.0.1:8000/signup/) |
| **User should see their new photo uploaded** |  |

|  |  |
| --- | --- |
| **Test Case Id: 002** |  |
| Test Case Name (Scenario): | Uploading a video |
| Summary: | A user should be able to upload a video when making a post, which will display on his dashboard. |
| Test Steps: | **Expected Result:** |
| **Start Papercut server** | Papercut application opens |
| **Open** [**http://127.0.0.1:8000/**](http://127.0.0.1:8000/products/) | Page launches successfully |
| **Select the text area field with “Write a post…” in it and enter “test”** |  |
| **Select the “Choose files” on the page and select (default.mp4) from your files folder.** | Text next to choose files should say “default.jpg” |
| **Click the “Post” button on the right** | Page redirects to users dashboard on  [http://127.0.0.1:8000](http://127.0.0.1:8000/signup/) |
| **User should see their new video uploaded and a notification is sent out** |  |

|  |  |
| --- | --- |
| **Test Case Id:005** |  |
| Test Case Name (Scenario): | Create GroupChat |
| Summary: | The user should be able to click the create group button on the home page and be brought to the create group page. Here they should pick a name, image and members for the group and then be redirected to the groupchat page.  The user must have mates before creating a group. |
| Test Steps: | **Expected Result:** |
| **Open home Page** |  |
| **Click Groups button** | Groups tab opens |
| **Click the create group button** | The Create Group page opens |
| **Write name in groupchat name field** |  |
| **Select image for groupchat image** |  |
| **Add members from members list** |  |
| **Click create group button** | Group chat is created and You are redirected to the groupchat page |
| **Assert the text for name** | Returns true |
| **Assert the text for member names** | Returns true |
| **Assert the group image is correct** | Returns true |
|  |  |

# 

|  |  |
| --- | --- |
| Test Case Id:006 |  |
| Test Case Name (Scenario): | Delete Profile |
| Summary: | To remove the user and profile from the database and everything that goes along with it.  The user must be logged in to an account before they can do this |
| Test Steps: | Expected Result: |
| **Open home page** | Home page loads |
| **Click on the settings drop down menu** | The settings drop down |
| **Click the delete profile** | Redirected to the delete profile page |
| **Click yes to confirm deletion** | Redirected to the sign in page |
| **Enter username ‘testuser1’** |  |
| **Enter password ‘Haddy1’** |  |
| **Click login button** | Fails with message user does not exist |
| **Assert text user does not exist** | Returns true |

# 

|  |  |
| --- | --- |
| Test Case Id: 007 |  |
| Test Case Name (Scenario): | Change Password |
| Summary: | To change password of the user, the user must be logged in |
| **Test Steps:** | **Expected Result:** |
| Load up home page | Page loads up |
| Click settings drop down menu |  |
| Click change password | Change password page loads up |
| Enter ‘Haddy1’ in old password field |  |
| Enter ‘Haddy123’ in new password field |  |
| Enter ‘Haddy123’ in confirm password field |  |
| Click change password button | Redirected to home page |
| Log out | Redirected to log in page |
| log in using username = ‘testuser1’ and password=‘Haddy1’ | Fails and returns message ‘invalid password’ |
| log in using username = ‘testuser1’ and password=‘Haddy123’ | Passes and redirects to home page |

# Conclusion

Achievements:

One of the biggest achievements we believe we have accomplished in this project is doing something different than the e-commerce site on its own. From the beginning we knew that we wanted to challenge ourselves with something we hadn't been shown how to do, and we decided to go with a social media website in the end. We had trouble throughout the project, a lot of research went into finding the best solutions for what we need and a lot of hard work went into the project, but this all added to the feeling of achieving a project that we are proud of. We are particularly proud of the CustomUser model that was created in the beginning of the project. This was the starting point of the project and from here we built around it adding more and more features. We definitely had some problems with CustomUser along the way which definitely drove us against each other at times, but overall it brought us to work together more than we had in the beginning of the project.

The Homepage became a big achievement throughout the project. In the beginning the homepage was only a logo and posts but the further into each iteration we got, the more filled in it became with buttons to other functions, comments, likes and our mates list, it ended up bringing everything together in one page.

There were some shortcomings while working on our project. There are two sides to these shortcomings, on one side there are the features we didn't fully implement or add in at all, there are also the shortcomings of us as a group.

With the features we didn’t fully implement or add in there are two notable examples, notifications and friend recommendations. We found that notifications would have taken too much time to implement and would have left us with little time to get the testing and bug fixing finished. We settled on using email notifications instead of notifications on the website as we already had an understanding of how emails worked, as we used them before. We used papercut as the server for this.

We also left out the feature of friend recommendations, this was again due to time constraints as we felt our time was better spent on features we knew we could finish and not spent adding in features that potentially wouldn't work 100% by the end of iteration 3.

There was also a couple of things we never quite got working like we wished like delete reports as the django mixin we were using wasn’t working, also things like a password reset and adding and deleting members to a group chat are all things we wished we had time to implement but couldn’t get done in the time frame.

On our shortcomings as a team we feel like while we worked well as a team, with each of us providing equal work, we believe we could have done more and been more cooperative with each other. We also think we didn't utilize our git repository as much as we should have been merging and pushing our code more often. The quarantine which happened during the last week of iteration 2 and lasted all of iteration 3 really brought our motivation down but luckily we found the light in more free time meaning we could get some extra work in. Our time management was also lacking as we found balancing part time jobs, other modules and hobbies with our work on the project. We found our busy schedules led to many days spent back after college working and many a time we stayed up into the early morning trying to get certain features to work.

We have definitely set up the project to do a lot of enhancements in the future. A Lot of them are actually ideas that we thought about implementing during our development cycle but unfortunately lacked the time and experience to add in successfully. For example we would love to refactor our messaging and group chats code and in websockets to allow users to message each other in real time since websockets are fully asynchronous.We would also allow users to see previews of the images before posting them, see link previews when a user would post a link, allow users to post images when making a comment, make users see who liked their posts. Apart from that we would do more research on file handling using validators so we could offer better quality images to users and explore solutions to a potential file storage bottleneck by using a cloud service like Django Google Cloud Storage. Adding friend recommendations would also be something to look at since we failed to implement it in the last iteration. There are thousands of features we could come up with by looking at mainstream social media websites to gain inspiration.

There were definitely a few shortcomings that came while developing Loop but also a lot of achievements that wouldn’t be possible without the dedication of each member. All in all the project was definitely a success and we showed ourselves that we are capable of developing a functioning site in a short time frame without having any previous experience in developing a social media website.

# References

Max Goodbridge, Django-Tutorials (2017), Github repository <https://github.com/maxg203/Django-Tutorials>

RooneyRulz, dev-social-network (2019), “Full stack social network app built with django, bootstrap including user authentications, permissions and more…”, Github repository <https://github.com/rooneyrulz/dev_social_network>

W3Schools, JavaScript Tutorial (2015)

<https://www.w3schools.com/js/default.asp>

W3Schools, JQuery Tutorial (2015)

<https://www.w3schools.com/jquery/default.asp>

Products used in Merchandise using

<https://www.printful.com/mockup-generator>