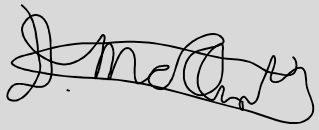
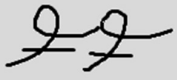
IAB330: Assessment 2

Scruff

Submission Coversheet Declaration

You must sign below. By signing this form you agree to the following:

- We declare that all of the work submitted for this assignment is our own original work except for material that is explicitly referenced and for which we have permission, or which is freely available (and also referenced).
- We agree that QUT may archive this assignment for an indefinite period of time, and use it in the future for educational purposes including, but not limited to: as an example of previous work; as the basis for assignments, lectures or tutorials; for comparison when scanning for plagiarism, etc.
- We agree to indemnify QUT and hold it blameless if copyright infringements are found in this work and the copyright owner takes action against QUT that is not covered by the normal terms of Educational Use.

John Tang	n9713913	
David McClarty	n9939768	
Fredrik Forsell	n10297057	
Ivar Sandvik	n10297154	

Executive Summary of MVP

Scruff is an application to connect potential pet owners with animals digitally. It facilitates the "seeking" process by allowing users to follow animals which require re-homing. By using Scruff, users will be able to experience more of the animal's personality over a long-term duration, before pulling the plug and committing. On the other end, current pet owners, businesses, and shelters can utilise Scruff as another marketing form for their animals. By taking advantage of the social media side of Scruff, they can keep followers up to date on the animal's life, which has a greater chance of adoption by the followers.

Scruff is a hybrid social media platform and marketplace for animals. By researching the current line up of applications already existing, we found that there were no applications which combined the two services. While it is the main function of Scruff, users will also be able to browse as normal, similar to Instagram or Facebook. This allows the platform to remain open as social media, rather than devolving into a marketplace only service.

The first viable release of Scruff will include the major features such as the ability to post and view animals, search and filter through them, and the ability to follow certain animals and owners. These features represent the core of the Scruff platform and will provide the most complete picture of the goals Scruff aims to achieve.

Due to time considerations, budget, and technology requirements, some extraneous features will not be implemented in this release such as the in-app messaging, post commenting, cloud storage integration, and other network required functionality, Wide device compatibility will not be fully reached (especially on Android) as it will require extensive testing and the purchasing of a wide variety of devices.

Discussion of Scoping Criteria

To be able to market Scruff, all the distinguishing features must be developed. Following are the key features that will be developed in the minimum viable product.

- **Accounts functionality**: (registering, logging in, logging out, creating animal listings, creating user profiles). This will allow the user to save posts, follow animals/shelters, and contact owners. For owners, this will allow them to be contacted, manage multiple animals, and post content. This is an essential feature for Scruff.
- **Home feed:** This will give each user a personalised feed based on who they're following. It will contain photos/videos/gifs.This is the core of the social experience.
- **Following/unfollowing**: Users will be able to personalise their feeds by following and unfollowing animals/users/businesses/shelters. Future releases will use the follower patterns to better suggest animals.This creates per-user personalisation.
- **Posting content:** Animal owners will be able to post photos (not videos or gifs) about their animals.
- **Searching and filtering animals:** To facilitate discovery, users will be able to search keywords such as names, locations, and breeds to find animals. This along with extensive filters will allow the user to narrow down searches if needed. A core part of the marketplace experience.

- **Individual animal page:** This page will display the animal's gallery, its details, and the contact information. As users will want to know more about the animal before adoption, this page is of high importance.

The features that will not be implemented in the first release are as follows:

- **Post commenting/sharing/liking**: In future releases, users will be able to like, share, and comment on images posted by an animals profile. This feature has been saved for later releases due to time constraints.
- **Saving posts**: Saving will make posts appear in a special page exclusive for saved posts. This gives users a page where they can find posts dating back long periods of time. Other features were prioritised over this one.
- **In-app messaging:** This feature was considered as it would help relate each message recipient back to their animal. Rather than traditional email or phone, Scruff's messaging will show which animal prompted the user to message this owner.
    - Group messaging may also be implemented.
- **In-app payment features:** This app will also be integrated with payment services such as Visa, Mastercard, and Paypal to facilitate the transfer of the animal. Included in later releases due to network requirements.
- **Animal buying guides / lifestyle blogs:** There will be documentation which will help users when finding an animal. Things such as FAQs, tutorials, blog posts regarding health, lifestyle, and similar commitments will also be present. This would require additional time, and therefore will not be in the MVP.
- **Sponsored posts / animal of the day:** Animals owners can pay to get the animal to the top of the page, increasing the chances of adoption. This would require payment functionality, and therefore will not be in the MVP.
- **Posting advanced content:** Animal owners will be able to post videos and gifs alongside images of their animals.

Feature List

Task priority levels: High (top priority), Medium (average priority), Low (bottom priority)

Task estimates: Measured in points, by ratio of how much time it would take to complete the task compared to the time it would take to complete a task estimated at one point (for example, a task that would take four times as long as a one-point task would be estimated at four points).

Although not specified, **unit testing** is included implicitly in each task.

Release 1 (MVP)

The following features will be completed for the application Scruff by its Release 1. This release will be the minimum viable product (MVP) of Scruff.

T1. Feature that allows the owner to input data about their animal, such as type, breed, age, weight, general information, etc. Pertains to use case 2, 3. Priority: High. Estimated points: 4

T2. Feature that allows the profile to be updated with images. Pertains to use case 2, 3, 4. Priority: High. Estimated points: 2

T3. Feature that allows users to search via a text bar. Pertains to use case 7. Priority: High. Estimated points: 2

T4. Feature that allows users to search via an advanced search tool where they can specify multiple attributes covering a wide range of aspects. Pertains to use case 7, 9. Priority: Medium. Estimated points: 4

T5. Feature that allows users to login. Pertains to use case 11. Priority: High. Estimated points: 2

T6. Feature that allows users to register. Pertains to use case 12. Priority: High. Estimated points: 2

T7. Feature that allows users to logout. Pertains to use case 13. Priority: High. Estimated points: 1

T8. Feature that places links to all the at the bottom of the screen in iOS, or on a left-side hamburger menu that can be opened in the top-left corner in Android. Pertains to use case 14. Priority: Low. Estimated points: 2

T9. Feature that allows the user to change their password if necessary. Pertains to use case 16. Priority: High. Estimated points: 2

T10. Feature that allows users to search via distance from the home of the owner to their home. Pertains to use case 7. Priority: Medium. Estimated points: 4

T11. Feature that allows app notifications. Priority: Medium. Estimated points: 2

T12. Feature that allows access to a user's gallery. Priority: Medium. Estimated points: 4

T13. Feature that allows use of the camera in-app. Priority: Low. Estimated points: 4

T14. Feature that allows users to follow specific animals via a "feed" page which is regularly updated with any media posted on each followed animal's profile. Pertains to use case 8. Priority: Low. Estimated points: 4

Future Releases

The following features will be completed for the application Scruff in future releases:

T15. Feature that will allow animals that belong to an animal shelter or similar organization, to be indicated as such in their profile. Pertains to use case 1, 9. Priority: Medium. Estimated points: 2

T16. Feature that allows users to search via a "shelters only" option. Pertains to use case 7, 9. Priority: Medium. Estimated points: 4

T17. Feature that will allow animals that belong to an animal shelter or similar organization, to be indicated as such in the results page of a search. Pertains to use case 1, 9. Priority: Medium. Estimated points: 2

T18. Feature that allows the profile to be updated with gifs and short videos. Pertains to use case 2, 3, 5, 6. Priority: Low. Estimated points: 2

T19. Feature that allows users to like media from an animal's profile. Priority: Low. Estimated points: 2

T20. Feature that allows users to comment on media from an animal's profile. Priority: Low. Estimated points: 2

T21. Feature that allows users (with animal profiles) to share media from other animal's profile. The post would be shared on the profiles' own feed so anyone following the profile that shared would see the shared post. Priority: Low. Estimated points: 2

T22. Feature that allows users to save media from an animal's profile. The saved posts will be in their own separate page. Priority: Low. Estimated points: 4

T23. Feature that allows users to message other users in-app. Priority: Low. Estimated points: 8

T24. Feature that allows users to pay other users in-app. Priority: Medium. Estimated points: 8

T25. Feature that allows users to message other users in-app. Priority: Low. Estimated points: 8
DUPLICATE^^^^ with T23

T26. Feature that allows users to access a FAQ/blog related to the application's use, looking after pets, and animals as a whole. Priority: Low. Estimated points: 2

T27. Feature that allows users to pay money to prioritize and advertise their for-sale pet. Priority: Low. Estimated points: 4

Project Timeline

- **Week 8**
    - **David:** T3 part 1
    - **Ivar:** T11
    - **Fredrik:** T12 part 1
    - **John:** T8
- **Week 9**
    - **David:** T3 part 2
    - **Ivar:** T1 part 1
    - **Fredrik:** T12 part 2
    - **John:** T6
- **Week 10**
    - **David:** T4
    - **Ivar:** T1 part 2
    - **Fredrik:** T13 part 1
    - **John:** T5, T7
- **Week 11**
    - **David:** T10 part 1
    - **Ivar:** T14 part 1
    - **Fredrik:** T13 part 2
    - **John:** T9
- **Week 12**
    - **David:** T10 part 2
    - **Ivar:** T14 part 2
    - **Fredrik:** Unit testing, any leftover work
    - **John:** T2
- **Week 13**
    - **David:** Unit testing, any leftover work
    - **Ivar:** Unit testing, any leftover work
    - **Fredrik:** Unit testing, any leftover work
    - **John:** Unit testing, any leftover work

Development / support tools

- Bitbucket
- Xamarin
- Gmaps API
- Android/iOS/UWP (Testing)
- MarvelApp (High fidelity Mockups)
- Lucidchart (UML, ERD)
- Nunit (TDD)

Security and Privacy Requirements

When developing Scruff we have a set of security and privacy requirements that we want to implement. We want to make sure that our users will feel comfortable with how their sensitive information is being handled by our system. This includes private information as phone numbers, emails, home addresses and passwords. However, since a lot of this information is stored and handled by the back-end server, the users won't be able to know how secure and private the data really is. For this reason, we must consider the ethical issue about how we should handle the security infrastructure.

In the first version of our app the users will be able to communicate directly to the sellers via email and phone number. To make this possible, the contact details must be available on the advertisement page for the animal. However, this increases the risk of spammers and cybercriminals collecting this information through an Email Address Harvesting process (also known as an email scraper). These emails are then put into a list used to send bulk emails or spam. To prevent this from happening to our uses, we will implement a method that will request contact details from the server only when needed. This way we can monitor users and see the amount of requests that happens within a certain timeframe. There will be 2 control mechanisms for detecting abuse:

1. If the requests happen **faster than humanly possible**, we will blacklist the account from sending more requests. The only way to get removed from the blacklist will be by contacting the support team.
2. If the requests happen **more than what is reasonable**, we will give the user a warning (3 request's left). Reaching the limit will put the user in timeout before being able to send more requests. Multiple violations will increase the timeout. This mechanism will be made to lock out smarter email scrapers that acts like they are human to bypass control mechanisms as the one above.

We will implement a Role-based access control, were privilege and rights will given to users depending on their role. We will have 4 different user roles:

1. **Unregistered**:
    a. Will be able to look/search for ads
2. **Users**: (same rights as Unregistered)
    a. Save/like/follow ads
    b. Request contact details from sellers
    c. Upgrade account to "Seller"
3. **Seller**: (same rights as Users)
    a. Post/edit their own ads
4. **Admin**
    a. Enforcing rules by editing or deleting advertisements.
    b. Able to demote users roles.
    c. Able to blacklist/whitelist users from specific features. (ex: request phone number)

We want our users to have a set of options of what information they want to have associated with their account. For instance, we will be giving them the possibility to choose their preferred communication methods. If they don't want to publicly share their phone number, they can use their email instead. Our database will be based on the Need-to-Know principle, which means that we will only store information that is necessary for our app to function as

intended. For instance, to be able to search for ads in different areas we need a home address. To enforce privacy for our users we will only include the street name in the ads. If a seller and buyer schedule a meetup they will have to give the rest of the information through their choice communication method.

We have an ethical responsibility to secure password in case of a security breach. Since people tend to use the same password multiple times, we have to make sure that potential attackers wont get access to their password. To prevent this from happening we will store all passwords in the database as salted PBKDF2 hash values. Salting the hash values with a unique identifier (such as their userID) will make sure that all passwords stored in the database has a unique value, even if it is the exact same password. This is because a minor change to the text that's being hashed will cause a major change in the output hash value. If it was not salted, a data leak may show that 10+ users were using the same passwords. The attacker could then try to bruteforce the hash value and get access to multiple plain text passwords in one attack.

Another potential threat is network sniffing. If the transmission of data isn't properly secured an attacker can monitor the local network and get access to unencrypted data. This can be enough for a hacker to gain access to the data being transferred. If the password is sent to the server in plain text this can also easily be picked up by network sniffers. We will implement SSL to encrypt the communication in our app.

Testing and Quality Assurance Strategy

We are planning to use a small variety of development and support tools. We have taken in consideration many tools we could potentially use to assist us during development such as third party tools. We need tools for the following tasks:

- Creating the structure of the application, including files and managing versions when working in team.

- A development environment for the application, which supports the tool above.
- Tools for efficient debugging when there are errors in the code (usually logical errors, since we will detect syntax errors during compilation).
- Tools for efficient testing of the application on mobile devices (emulated and physical), to detect errors.
- Tools for testing back end logic/code.
- Tools for UI testing and testing on multiple device brands and models.

Creating and maintaining the structure of the application

In a open source development situation we would be using Github for managing our codebase and collaboration. In order to not have to spend money on a Github premium subscription for private codebase we opted for a similar product called Bitbucket. This is a software project development platform where we can store our files and repositories. It connects right into our development platform mentioned underneath. It supports forks, branches and access control which in turn makes our project structure and files nice and tidy.

Development environment for the application

We choose to use Visual studio xamarin edition for development of our application, since we think this IDE suits our needs best. We will be using the tools that Microsoft has provided in this IDE for testing. The tool includes access to online emulators, local emulators, UI testing, server code emulation, and many other features. In addition this tool is recommended to use in our course.

Tools for debugging the application (after compilation)

When we are developing the application we can use the emulators available in the IDE. These are actually the stock android emulator and iOS xcode emulator. With multiple versions of the operating system available. We can use the online emulators as well provided so we don't have to download all the different versions of the OS to our computers. This will save us some time on the android platform, since the android versions that users use is pretty segmented.

Tools for testing the backend code and logic

We are planning to use "App service helpers" which is a tool provided by Microsoft connecting your application to the backend with very little effort. We will most likely use this as our backend. We have considered other measures for connecting our application to the backend such as java, php. But since Microsoft has made it easy for us to set up the communication channel with Xamarin we opted for that option.

Tools for UI testing on multiple device models/brands

We will be using xamarin automated UI testing on our application. It's pretty easy to set up and we feel like it will handle some of our more simple use cases. We also plan to use xamarin test cloud which tests our application on real devices in a physical location. This will accelerate our testing process and we will find problems faster. For example errors related to other OS versions and devices. We will also do some manual testing for specific use cases that are faster to manually than to record.

We have concluded that we will need a solid testing strategy including the following testing progress:

- We will start by creating test cases. This defines the actual operation we are going to test. In simpler words it means what we are going to do with the app. We can use our User stories here for common sequences that the actual users will preform.
- Control of the result of the test case. This will be our control of the result from the operation performed above. In simple terms control that the operation achieved the desired output/result. In example that a login test with correct credentials resulted in a successful login.
- Testing with different kind of user inputs. For example a space in a text box intended for **int** data type can result in a application exception. So we have to test to ensure that exceptions do not happen often and make tests within our logic. So we have to test with **both** correct and incorrect inputs and make sure the expected result is correct for the context.
- We will try to imagine and find all possible indirect results from incorrect and correct operations, then verify that the results (what happens after/during operations) does not hurt our application: crashes, exceptions, incorrect data sent to backend etc.

As well as our testing strategy we need a quality assurance strategy. We have discussed and made directions for ourself to ensure that our software has a good quality level. Our strategy is as follows:

Functional quality strategy

- When we planned how the application should operate, we first thought about use cases and user stories. The user stories represented the most common user actions when using the application. After that we designed and settled on a final UI design that compliments the actual goals the end user might have.
- In order to respond to user needs, we have taken in consideration what potential dog buyers and sellers want when they are selling/giving away their dog. This includes a clear story in the application aimed towards the action of selling/buying/adopting a dog.
- The functionality in the application is clearly aimed for animal sellers/and buyers. The application is made just for this purpose. There is a social media aspect to the application as well that can compliment user interaction and returns (returning to use app).

Structural quality

- Our software should be reliable and behave in ways the user expects. We want to achieve this with extensive testing before we publish the application for users. This includes UI testing and looking for errors within the application (see testing strategy above for details).
- The software should be easy to learn, we have used as few activities (screens in application) as possible without cramping the screen with too much information. We might also add a small user guide when opening the app the first time).
- When we planned the design (technical) of our application, including backend, database and front end we will focus on security. See our security and privacy requirements.
- Efficiency is also important in our app development progress. We will try to minimize unnecessary processor usage and often more important the cellular data.
- As mentioned in our use of development tools, we will be using Bitbucket when we develop the application and maintain our structure there by using functions like forks and branches. This will help us when we have deployed the application because our structure will be maintained.