APRIL 8, 2018

TESTING DOCUMENTATION

MID – MY IDENTITY

CILLIAN MC NEILL - 14352621
SCHOOL OF COMPUTING
SUPERVISOR: GEOFF HAMILTON

Table of Contents

Introduction

This document outlines the steps taken to test the MiD application and the various platforms that it has been deployed to. Testing methodologies changed depending on the platform as there were different requirements of each. Within each section, I will detail the different steps and reasoning behind each one.

Mobile Testing

Overview

The mobile portion of the MiD application was created for the end user. This would be someone that wants to verify and store their identity. This application allowed for this as well as the requesting of information from other users. The primary function of MiD was to provide endpoints that users and parties could subscribe to. This means that a user interface (UI) wasn't exactly necessary. However, I feel that if the application needed to be demoed then we need some form of UI to show to someone. This also means that testing didn't need to be as rigorous in terms of code/path coverage. I opted for a more functional form of testing for this portion of the project. This came in the form of myself performing tests and putting the potential application out to users to get feedback on and expose bugs that I would not have been able to spot. As the application was developed for android, I was able to avail of Google's "Play Console" and the testing platform that came with it.

Functional Testing

For basic testing of the application I developed a chart of several workflows that can be accomplished. These ranged from big tasks such as creating a profile to something secondary like viewing a pending submission. Each workflow has a set input and result. Below lists out these workflows detailing their criticality to the application and perceived difficulty.

The second charts list out the results of the tests performed on those workflows. I change the input of each and see if I get the expected output or correct error message.

Functional Testing – Workflow Chart

Workflow ID	Description	Application Criticality	Perceived Difficulty	Expected Input	Expected Result
1	Create a profile locally to be sent to the server for remote storage	High	Low	Username & Pin	Profile Created Remotely and returned. Local profile then created. User is then prompted to create an identity card.
2	Create an identity type locally	High	Medium	Fields from the selected identity (eg. name, d.o.b, etc.)	Card is stored locally and displayed to the user.
3	Submit an Identity Type	High	Low	Confirmed fields from that type of identity and a current photo	Change the card status to pending and display it to the user.
4	View submitted identities	Medium	Low	Tap on the respective button.	A listing of all submissions is returned to the user.
5	View submitted identity	Medium	Low	Tap on the desired identity	User is shown the submission and all information for it.
6	Create Request	High	High	Scan user QR code and select fields from a specific identity type	Request is created and displayed to the user
7	View Requests	Medium	Low	Tap on the respective button.	A listing of all requests is returned to the user.
8	View Request	Medium	Low	Tap on the desired request	User is shown the request and all information for it.
9	Create Second Profile	Low	Low	User selects the create button on the profile selection page	User is prompted to create a user in the standard way
10	Delete Identity Type	Medium	Low	User selects options on the desired identity and selects select	The selected identity is removed from the profile
11	Delete User Profile	Low	Low	From the user page they select delete	User profile is deleted

Functional Testing – Results Chart

Workflow ID	Input	Expected Result	Actual Result	Comment
1	“test” for username and 1234 for the pin	Profile is created called “test”	Profile is created called “test”	Within expected parameters
1	“test” for username and no pin	Error	Prompted to enter a pin	Within expected parameters
1	No username and 1234 for pin	Error	Prompted to enter a username	Within expected parameters
1	“test” for username and 1234 for the pin but no internet enabled on the device	Error	“Error creating profile” message displayed	Profile cannot be created without being connected to the server
2	“test” entered for identity type for one field	Identity type created	Identity type created locally	Within expected parameters
2	nothing entered for identity type for one field	Error	Prompted to enter something into every field of the identity	There is nothing to check what is being entered only that something is entered
2	“test” entered for identity type for one field with no internet enabled on the device	Identity type created	Identity type created locally	No need for internet at this stage as the identity is created locally
3	Nothing changed on the created identity type (left as “test”) and a picture is taken	Submission created and set to pending	Submission created and set to pending	Within expected parameters
3	Fields are emptied and picture is taken	Error	Prompted to enter something into the fields	Within expected parameters
3	Nothing changed on the created identity type (left as “test”) and no picture is taken	Error	Prompted to take a picture	Within expected parameters

3	Nothing changed on the created identity type (left as “test”) and a picture is taken but no internet is enabled on the device	Error	“Error creating submission” message displayed	Submission cannot be created without being connected to the server
4	Submission list button is tapped	Submission listing shown	Submission listing shown	Within expected parameters
4	Submission list button is tapped but no internet is enabled on the device	Error	“Error retrieving submissions” message displayed	Internet is required to pull down the latest version of the submission listing
5	Desired submission is selected	Submission shown	Submission shown	Within expected parameters
5	Desired submission is selected but no internet is enabled on the device	Error	“Error retrieving submission” message displayed	Internet is required to pull down the latest version of the selected submission
6	User QR is scanned and fields from an identity type are selected	Request Created	Request is created and user goes back from the request page	Within expected parameters
6	User QR is not scanned and fields from an identity type are selected	Error	“Must scan user QR” message displayed	Need to know what user you’re sending the request to
6	User QR is scanned and no fields from an identity type are selected	Error	“Must select a field” message displayed	Need to know what to request from the user
6	User QR is scanned and fields from an identity type are selected but no internet is enabled on the device	Error	“Error creating request” message displayed	Internet is required to create a request
7	Request list button is tapped	Request listing shown	Request listing shown	Within expected parameters

7	Request list button is tapped but no internet is enabled on the device	Error	“Error retrieving requests” message displayed	Internet is required to pull down the latest version of the request listing
8	Desired request is selected	Request shown	Request shown	Within expected parameters
8	Desired request is selected but no internet is enabled on the device	Error	“Error retrieving request” message displayed	Internet is required to pull down the latest version of the selected request
9	Select new profile. Button and enter test data “test” for username and 1234 for pin	Profile is created and displayed in the profile selection page	Profile is created and displayed in the profile selection page	Within expected parameters
10	Identity type delete button is selected	Identity type is removed	Identity type is removed	Within expected parameters
11	User deletion button is selected	User profile is removed from the device	User profile is removed from the device	Within expected parameters
11	User deletion button is selected but no internet is enabled on the device	Error	“Error deleting profile” message is displayed	Internet is required to delete the profile locally and remotely, it cannot be only half-done

Functional Testing Conclusion

By following the above workflow, I was able to work out any bugs that would occur during big changes or updates as a result of changes to the backend. Whenever a change anywhere was made I would run through the relevant workflows and ensure that the behaviour matched up. These workflows were a heavy influence on what I asked users to do during the user testing portion of development.

User Testing

When creating something like a mobile application it's imperative to ensure that the target audience is able to use it to the best of their ability. This means that a user must always know what they're doing at any point in the workflows of the application.
As this application will be used by any and all age groups, as identity cards are age agnostic, the screens within the mobile application needed to be clear and simple. All steps of each workflow needed to be clearly labelled to ensure the user takes the correct actions.
To cover a wide audience I deployed the application so a publicly accessible server and had friends and family test the mobile portion of the application and fill in a survey. All were encouraged to ask their own acquaintances to test it if they so wished.

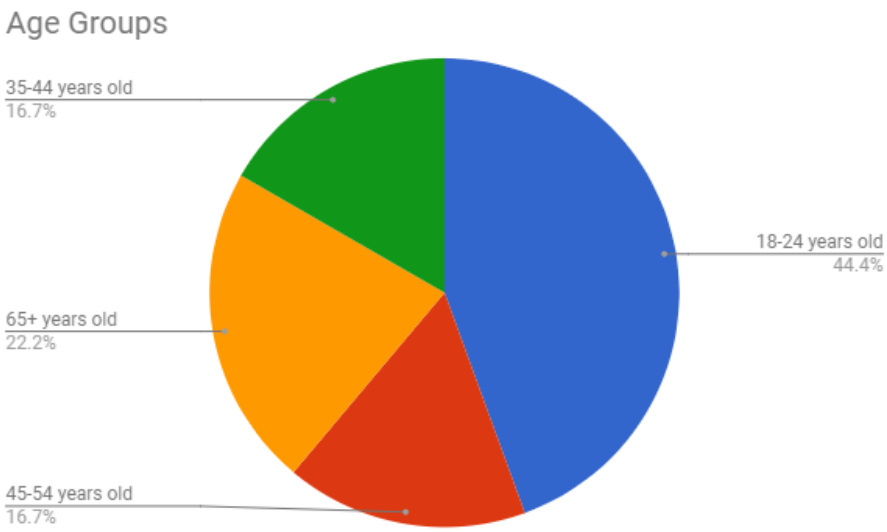
User Testing Breakdown

The participants we asked to install the application to their android phone (as this is currently the only mobile operating system the application supports) and work through several workflows:

- Creating a profile
- Creating a (local) identity card
- Submitting that created identity
- Creating a request to another user (I was used in most examples here)

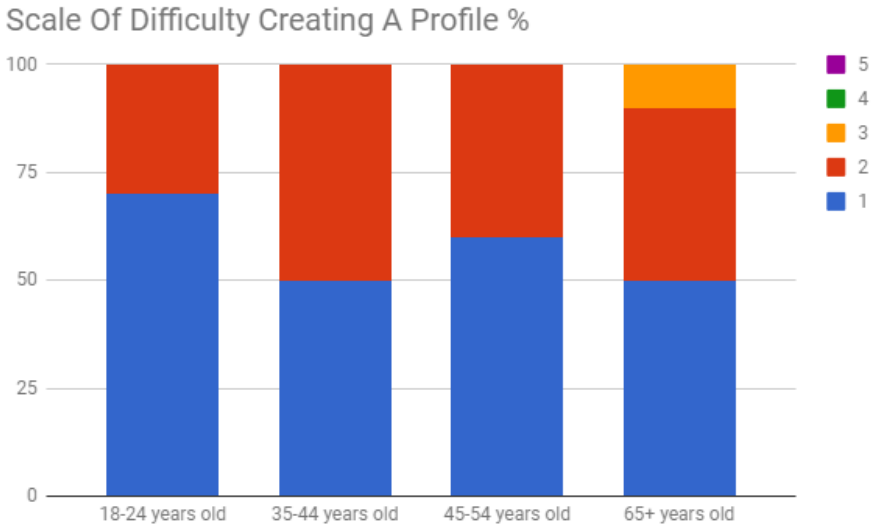
Once this was completed they were asked to complete a short survey and give me their thoughts on these workflows. Answers were anonymised and compiled below into their respective sections.

General User Information



Here is a chart of the distribution of users I asked to test the application. I tried to have as diverse a range as possible so that I could get a variety of feedback on the application. As you can see the primary group of users that tested the application were between the ages of 18-24. I don't really see this as an issue as this would be the age group that would be quicker to pick up a mobile application over conventional means.

Creating A Profile



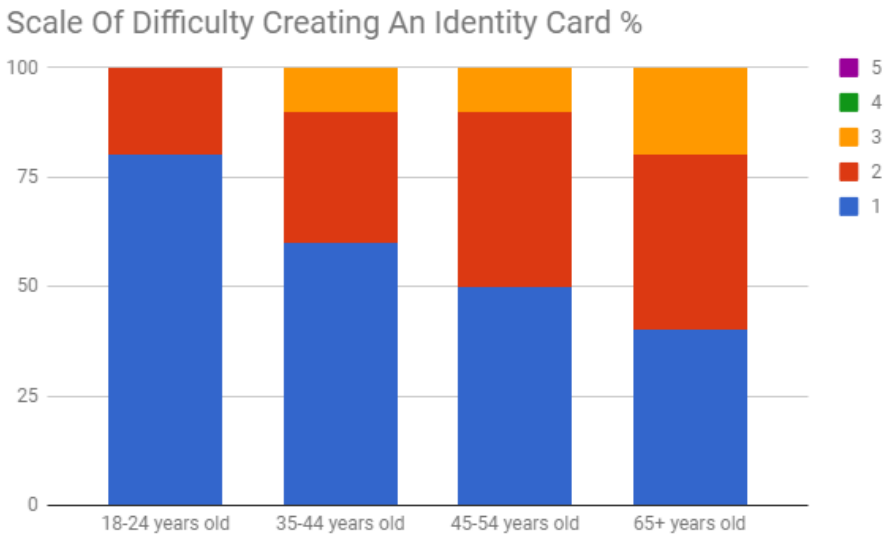
This chart details the feedback received on the difficulty of creating a profile on a scale of 1 to 5 (1 being the easiest and 5 being the hardest). You can see that the majority of users had little to no issue creating a profile. Difficulty remained around 1-2 with several 3's on the higher age bracket. Some of the feedback included:

- *"Unlabelled text fields were confusing" - 45-54 Years Old*
- *"Unsure of pin length required" - 35-44 Years Old*

The common point between these were that the fields were unlabelled and confusing. While there are only 3 text fields for the user to input data into, I want to ensure that they know exactly why they're doing it.

Actions Taken: Headings and input fields are given clear labels as to what to put in

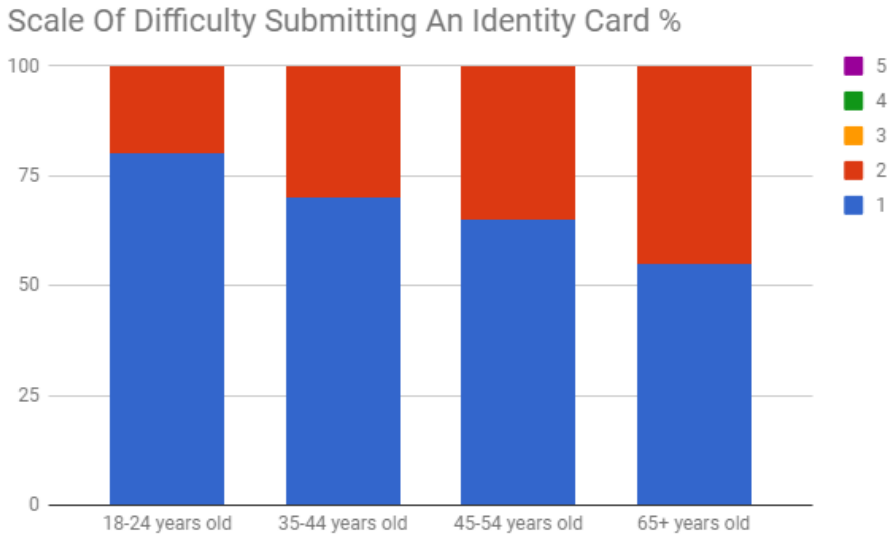
Creating an Identity Card



From this we see the difficulty users had with creating an identity was slightly higher than before. I hypothesize that this is due to the increased complexity of the data being asked for. It's also interesting to note the increase in scale with the increase in age. While this is not true for all it's certainly interesting to see it occur here at a small scale.
Again most of the feedback I received was around the headings and information that was being asked of the user. One user expressed concerns of security of their data. I needed to describe that it wouldn't leave the phone at any point unless they gave the application permission (during the submission of request workflow for example).

Actions Taken: While I can't address the security question of that user in an in depth manner within a simple form, I can subtly let the user know the form is being stored locally only at this stage. Along with this I need to once again improve the labels of the form.

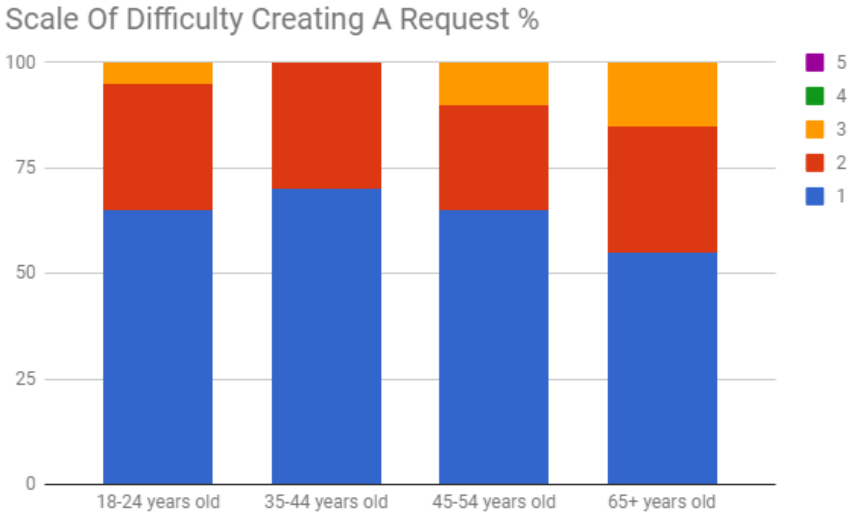
Submitting A Created Identity



The scale of difficulty encountered here seemed to be a lot lower. Perhaps this is due to the fact that you're not asked for more information beyond taking a picture. This however was a point of contention as the users often didn't know what to take a picture of, themselves or the identity card. Users questioned where the information would go once they submitted it (some thought that the application was actually tied to the passport authority, the owners of the example identity card being used) and how long a submission would take.

Actions Taken: While I don't want to break that illusion of something actually being submitted somewhere real as that's ideally what I'm aiming for, I let the users know that submission times are dependent on the parties taking care of their respective identity types.

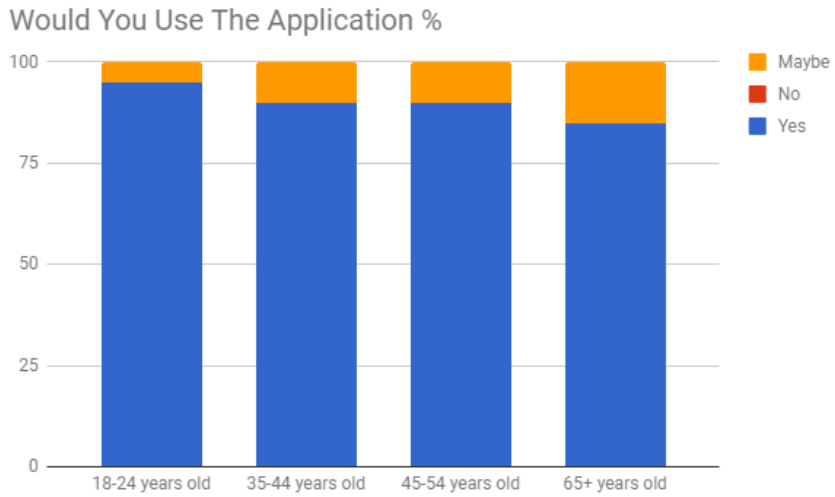
Creating A Request



This would be one of the more difficult tasks to accomplish within the application due to the complexity of what you need to do. My worries were unfounded as the majority of users had no problems scanning another users QR code (in a lot of cases it was my own) and requesting parts of information from that user.
As there are only 3 tabs within the application I assume finding it is more down to a process of elimination rather than naturally being able to find it. I don't really think this is an issue due to the fact that there are so few screens within the main portion of the application. There were no specifics remarks made on this section beyond what has been said before about labelling.

Actions Taken: Labelling was improved and the profile page of the user was consolidated into one page with popups. This means that the user can gain access to their own QR code within the same location that one would begin to make a request of another user.

User Testing Conclusion



At the end of the questionnaire I asked the users to give overall feedback on the application. This included what they thought of the layout, any overall difficulties and if they thought they would use the application in a real world scenario.
There was very positive feedback from the users on whether they'd use it or not. Nobody said that they wouldn't and a lot mentioned that if it was widely accepted then they would prefer to use it over a traditional form of identity. Some of the feedback included:

- *"App is the answer to a gap we are having in infrastructure at present"*
- *"It needs to replace all forms of identification NOW!"*
- *"To be able to manage multiple cards /forms of identity from a single app would be very convenient"*

This feedback is very promising and leads me to believe that if an application like this was implemented in multiple places then it would be widely adopted.

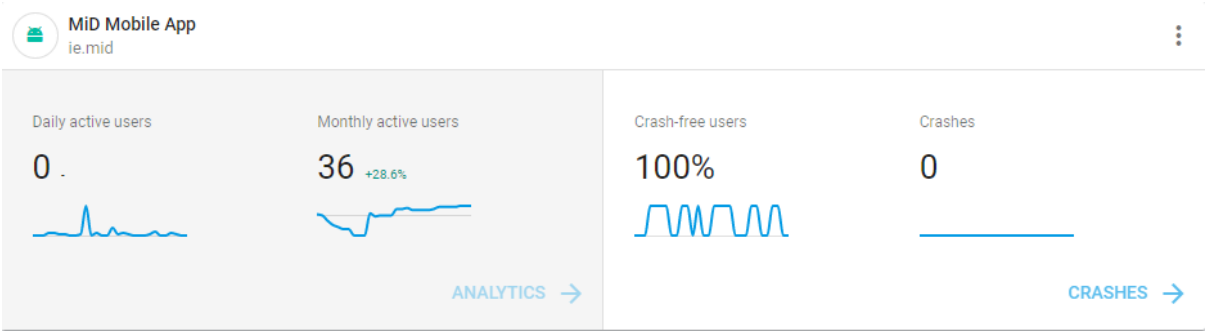
- *"I'm uncomfortable putting things like this online"*

This is a very reasonable response and I received other similar remarks about identity online and its security. I appreciate users are thinking about security of their personal information, especially their identity online. If this was to go into full production I feel it would be necessary to stress the security that goes into keeping this information safe but above all else, it's important to let users know that they have full control over who has access to their identity within the application.

One of the main points I took away from the user testing was the difficulty surrounding the actions users are supposed to take at any point. In several of the workflows I found some users were confused over what they had to do next.
To alleviate this, I implemented more direction and substance to the screens being displayed. This came in the form of extra headers and sending users to specific pages when an action is taken (e.g. Going to the submission page of the submission they have just created rather than dumping them back on the main screen). Some of these changes were implemented and retested against users, primarily the addition of further headings. I saw much more positive reactions to each screen which led me to add even more where I thought they were necessary.

Automation Testing

Overview

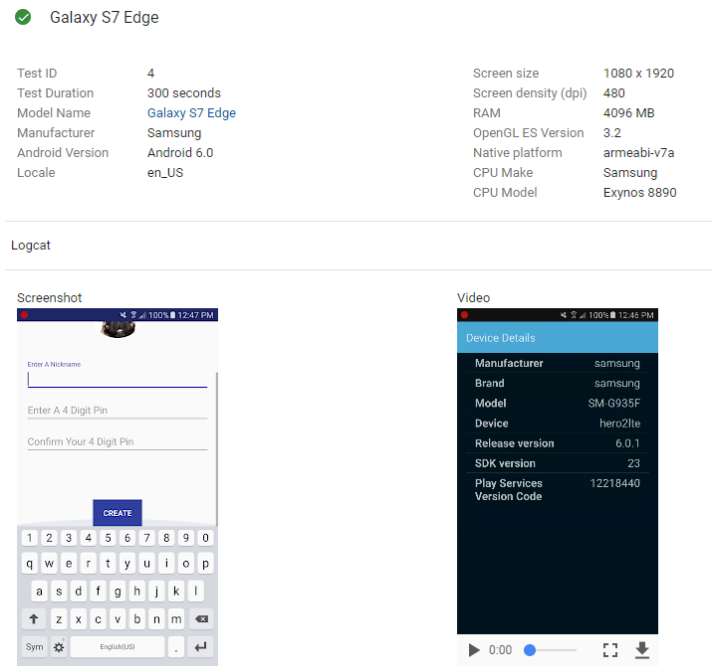


For easy distribution of the mobile application I uploaded it to the Google Play Store. This has several benefits such as the ability to create various release branches for segmented user-base testing and the ability to view statistics of the live application.
It also has the added benefit of integrating Google Firebase into the live application. This allows for deeper analytics of usage and crash data and the inclusion of test scripts to be made. These test scripts can be run against the application using a cluster of different devices. Each device will have its own version of android and screen resolution to allow for quick verification of the apps functionality across devices. Each test device returned a success report with screenshots, video clips of the devices screen and the logcat files from the test.

Automation Testing Results

Model Name	Android Version	Test Result
Xperia XZ Premium	Android 7.1	PASS
Mate 9	Android 7.0	PASS
P8 Lite	Android 5.0	PASS
Galaxy S7 Edge	Android 6.0	PASS
LG G6	Android 7.0	PASS
Google Pixel	Android 7.1	PASS
Google Pixel	Android 8.0	PASS
Moto G4 Play	Android 6.0	PASS
Galaxy J7(2016)	Android 6.0	PASS

The above table lists the devices that were used to test the application. The application was designed for devices running Android 5.0+ so no devices running lower than this version were used.
The testing structure followed the same flow as the functional testing with one exception. Due to the way requests work I didn't include that workflow and as such it wasn't tested here. I feel the coverage in both functional and user testing has allowed me to properly test this portion of the application regardless.
The primary goal of these tests was to see how the UI functioned on different screen resolutions and versions of Android. There were no issues at all and has given me confidence that the application would work across a multitude of devices.



Automation Testing Conclusion

The primary goal of these tests was to ensure that the application could function at low and high resolutions without any error. The tests were easily repeatable and meant I could expose any errors that would occur from running different devices. As the project scales up I'd be able to add more and more devices to support different versions and screen resolutions.

This testing also allowed me to catch errors that had come from "quick fix changes". It allowed me to fully ensure the workflows could be carried out correctly with no issue.

Backend Testing

Overview

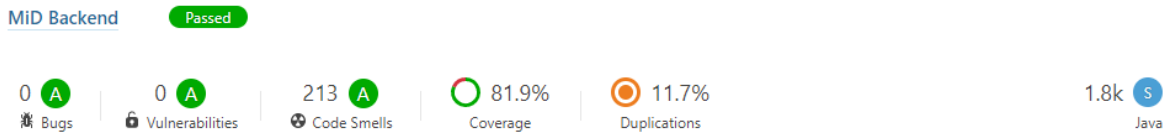
The backend of MiD is the core of the application. It's the hub through which all other applications communicate through. As such it required in depth testing to ensure it worked from a functionality perspective and method-by-method perspective.
I took a continuous testing approach to the application. This means that for every new piece of functionality created there needed to be tests to ensure it worked as I wanted it to. This initially took the form of unit tests and were built out into full integration tests as it scaled up. The testing takes a white box approach so that I could eliminate any issues that popped up along the way.

Unit Testing

Overview

The backend application has an object orientated design. This means that both the classes and methods within the programme are separated out into what function they offer. This allows for easy expansion of the application to meet new requirements.
For unit testing, I made use of the Junit and Mockito libraries. These allowed me to easily construct tests for each method and mock out any dependencies that each class had. For example, If I had to test class A but it depended on method b from class B then I could easily mock out what would get returned from that method whenever I called it. Classes can be easily tested in isolation and the whole suite can be easily run with any maven build.

SonarQube



To allow for quick, continuous, test-driven development I integrated SonarQube into the application. This allowed for continuous reports of code quality in the form of code smells, bugs and vulnerabilities. It is a tool that's used highly within the industry today and it has been extremely beneficial to the project.
When unit tests on the project are run, a report is generated from them containing all results of the tests (successful tests, code coverage, etc.). This report is parsed by sonar and an ongoing record of each report is updated. I can see what has been changed and where new problems arise. It also keeps close track of code coverage within the application. This has been integral to the ongoing development of the project and has sped up things significantly in terms of refactoring and testing.

Integration Testing

Another integral part of testing the backend was the integration tests. These are tests that were developed to ensure that all components of the backend part of the application worked. As this is a crucial part of MiD overall, I had to ensure that tests covered as many possibilities as I could think of. To allow for easy code-readability and reuse I developed these tests using Cucumber. It runs acceptance tests in a behaviour driven development (BDD) style. Each test is written in plain English which is then translated over to actual methods that I have written. This means that developers external to the system can get a clear idea of what the tests are supposed to accomplish without having to read through lines of Java code.

```
Scenario: client makes a successful call to POST /party
  When the client calls /party with party data
  Then the client receives status code of 200
  And the client receives back a reference to the created party

Scenario: client makes a failed call to POST /party
  When the client calls /party with no party data
  Then the client receives status code of 400
```

Above we can see an example of two scenarios used within the backend integration tests. Each scenario defines independent events that can occur within the application. These scenarios are broken down into logical steps that will happen as a result of the previous steps ie. “When I turn the key in my car the engine starts”.

```
@When("^the client calls /party with party data$")
public void theClientCallsPartyWithPartyData() throws Throwable {
    HttpUtil.endpointExtention = "/party";
    StoredParty party = new StoredParty();
    KeyPair keyPair = KeyUtil.generateKeyPair();
    if (keyPair == null) throw new NullPointerException();
    party.setPrivateKey(KeyUtil.byteToBase64(keyPair.getPrivate().getEncoded()).replace( target: "\n",  replacement: ""));
    party.setPublicKey(KeyUtil.byteToBase64(keyPair.getPublic().getEncoded()).replace( target: "\n",  replacement: ""));
    party.setName(NAME);
    ObjectMapper mapper = new ObjectMapper();
    PartyDTO partyToCreate = new PartyDTO();
    partyToCreate.setName(party.getName());
    partyToCreate.setPublicKey(party.getPublicKey());
    StorageUtil.storedParty = party;
    HttpUtil.sendPost(new HttpCall(mapper.writeValueAsString(partyToCreate)));
}
```

Above is the implementation of the line “When the client calls /party with party data”. You can see that a single English sentence expands into multiple lines of Java code. This is true for all 5 lines of English seen above.

The Java code used in the integration tests primarily makes use of two static classes, “HttpUtil” and “StorageUtil”. HttpUtil makes all HTTP requests and stores the last request body and code of the last GET,POST,PUT,DELETE call made. StorageUtil works as local storage for every scenario. I can use it to store current users that have been created of pending submissions that I am doing a full end to end test of. These static methods allow scenarios to be easily constructed without worrying about the effect it will have on previous ones.

Backend Testing Conclusion

Both unit testing and functional testing don’t require a conclusion on their own. The idea of each is to build up rich code/path coverage of the entire backend application to ensure that all use cases are accounted for and handled appropriately.

Admin Testing

Overview

The admin interface itself was designed as a proof of concept for what an identifying party can accomplish with the application. Its not meant to be deployed as an actual part of the full application. As such, the tests are just used to verify that a user can use the endpoints from a simple Angular interface. There is no access control to these pages and anything created using them can be viewed by anybody else using the interface. Its meant to be for testing of the other portions of the application.
If MiD were to be deployed somewhere then they would just get access to the endpoints that the Admin interface makes use of. They would have to take care of implementing it into their own system's UI.

Functional Testing

The only form of testing that was performed on the Admin Interface was a simple for of functional testing to ensure that any user that develops a similar web interface will have no problem using the application. Its important to note that access control is implemented in the form of 256bit encrypted tokens. These keys need to be created and stored by the end user. The server will hold onto the public key to allow for secure communication with the intended user, but it is up to them to look after the private key and the token.

Functional Testing – Workflow Chart

Workflow ID	Description	Expected Input	Expected Result
1	Create a party	Username	Party created and returned
2	Create an identity type	Title, image url, desired fields	Identity is created and linked to party
3	Update Identity type	Updated Title, image url, desired fields	Identity is updated, version number updated if necessary
4	Delete identity type	Click on the respective button.	Identity type is removed
5	View Submission	Click on the desired submission	User is shown the data of that submission
6	Accept Submission	Click on the respective button.	Submission is marked as accepted and returned
7	Reject Submission	Click on the respective button.	Submission is marked as rejected and returned

Functional Testing – Results Chart

Workflow ID	Input	Expected Result	Actual Result	Comment
1	“test” for username	Party is created called “test”	Profile is created called “test”	Within expected parameters
1	no username	Error	400 error	UI doesn’t handle errors, Expected to know legal/illegal types
2	‘test’ for title and image url with one field called test	Identity created called “test”	Identity created called “test”	Within expected parameters
2	No title or image url with one field called test	Error	400 error	UI doesn’t handle errors, Expected to know legal/illegal types
2	‘test’ for title and image url with no entry fields	Error	400 error	UI doesn’t handle errors, Expected to know legal/illegal types
3	Title changed to “new test”	Identity updated to “new test” and version updated	Identity updated to “new test” and version updated to 2	Within expected parameters
3	Image url changed	Identity updated with new url and version not updated	Identity updated to “new test” and version still set to 2	Within expected parameters
3	Only entry field changed to “new test”	Identity updated to with field called “new test” and version updated	Identity updated to with field called “new test” and version updated to version 3	Within expected parameters
4	Delete button clicked and confirmed	Identity type deleted	Identity type deleted	Within expected parameters
4	Delete button clicked and not confirmed	Identity type not deleted	Identity type not deleted	Within expected parameters
5	Submission for party clicked	Submission data displayed	Submission data displayed	Within expected parameters

6	Accept button clicked for submission	Submission marked as accepted	Submission marked as accepted and page is refreshed	Within expected parameters
7	Reject button clicked for submission	Submission marked as rejected	Submission marked as rejected and page is refreshed	Within expected parameters

Functional Testing Conclusion

From above we see that the testing performed on the admin interface was very minimal. This interface is only meant to be used as a demo and as such, it shouldn't be tested as rigorously as the rest of the platform. If an authority were to approach me and request a version of the interface, then it could certainly be built up to function as a full-fledged application but right now it's not necessary.