

Securing Ideation

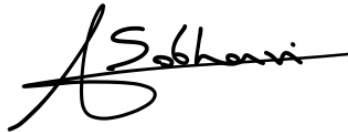
Submitted May 2020, in partial fulfilment of
the conditions for the award of the degree **BSc Computer Science**.

Student's ID: 4301063

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated
in the text:

Signature:



Date: 04/05/2020



The University of
Nottingham

UNITED KINGDOM • CHINA • MALAYSIA
School of Computer Science

4301063

Supervisor: Dr. Tim Muller

Module Code: G53IDS

2020/05

UNIVERSITY OF NOTTINGHAM
SCHOOL OF COMPUTER SCIENCE

DISSERTATION REPORT
Securing Ideation

Student ID:
4301063

Supervisor:
Dr. Tim MULLER

May 4, 2020



Abstract

This paper sets out to simplify the process of ideation whilst making it possible to document and share valuable IP with others securely using a simple android application. Key security concepts such as asymmetric encryption are used to implement a reliable digital signature system, making the process of signing legally binding documents easier than ever before. An elaborate range of research is conducted, analysing similar products, concepts and platforms. We then go onto defining the project and its requirements before making some important structural decisions discussed in the methodology. A mobile app and database are then designed and implemented leaving us with a full stack application, the resulting software is then tested and evaluated to ensure a high level of quality. Finally we reflect on the achievements and limitations considering what needs to be done before it can be deployed. The Ideation project started as a proof of concept but concludes not far from a final product, that is able to overcome the problems we set out to resolve.

Acknowledgments

First and foremost I would like to thank my supervisor, Dr. Tim Muller, for his advice and guidance throughout this project. Next I would like to thank Ariel Makambo for his professional perspective and legal analysis on the matters discussed in this paper. Then I would like to thank Nima Sobhani and Afshin Shahi for supporting me, in both the report and my academic career. Finally I would like to thank my friends, Adam Kulpa and Marek Canavan, for consistently motivating me to achieve more and accompanying me during the highs and lows of this difficult year.

Contents

1	Introduction	1
1.1	Overview	1
1.2	Background and Motivation	1
1.3	This Report	1
2	Related Work	2
2.1	Existing Applications	2
2.2	Existing Protection Measures	2
2.3	Existing E-Signature Platforms	3
3	Description Of The Work	3
3.1	Key Objectives	4
3.2	User Requirements	4
3.3	Software Requirements	5
3.4	User Flow Diagram	6
4	Methodology	6
4.1	Integrated Digital NDA	6
4.1.1	Asymmetric Cryptography	7
4.1.2	Android Frameworks	7
4.2	Business Development	9
5	Design	9
5.1	User Interface	9
5.1.1	HCI Research	9
5.1.2	Initial Designs	10
5.2	Database	11
5.2.1	Infrastructure Research	11
5.2.2	Schema Design	12
5.2.3	Document Field Properties	13
5.2.4	Data Persistence	15
6	Implementation	15
6.1	Launcher	15
6.2	Registration and Authentication	15
6.3	Login and Forgotten Password	16
6.4	Navigation System and Fragment Optimisation	17
6.5	User Profile Section	18
6.6	Projects Section	18
6.6.1	My Projects Tab	19
6.6.2	Shared Projects Tab	22
6.7	Discovery Section	24
6.8	Selected Project Page	25
6.9	Access Request States	26
6.9.1	Possible States	27
6.9.2	Requests State Diagram	27
6.10	Digital Signature and Encryption	28
6.10.1	Generating The Keys	28
6.10.2	Hashing The NDA File	28
6.10.3	Creating The Digital Signature	29
6.10.4	Signature Verification	29

7	Project Evaluation	30
7.1	Application Testing	30
7.2	Digital Signature Testing	30
7.3	Acceptance Testing	31
7.4	A Legal Perspective	32
7.5	Limitations	32
7.5.1	Real ID Verification	32
7.5.2	Multiple Device Key Support	32
7.5.3	UX Testing	33
8	Summary and Reflections	33
8.1	Project Management	33
8.2	Reflections	35
8.3	Conclusion	36
9	Bibliography	37
10	Appendix	39
10.1	Ideation User Manual	39
10.1.1	Application Set Up	39
10.1.2	Getting Started	39
10.1.3	Project Guide	39
10.2	Evaluation Tables	40
10.2.1	Implementation Testing	40
10.2.2	Acceptance Testing	43
10.2.3	Resolved Issues	45
10.3	Project Management	47

List of Tables

1	User Document Fields	13
2	Project Document Fields	13
3	Access Requests Document Fields	14
4	Application Testing	40
5	Digital Signature Testing	42
6	Functional Requirements Testing	43
7	Non-Functional Requirements Testing	44
8	Software Requirements Testing	45
9	Application Resolved Issues	46

List of Figures

1	User Flow Diagram - Current Design	6
2	Asymmetric Cryptography - Visual Representation [2]	8
3	Initial Design - Login Page	10
4	Initial Design - Navigation Menu	10
5	Database Schema - Collections and Documents diagram	12
6	Firestore UI - Authentication Panel	14
7	Application - Splash Screen	16
8	Application - Account Creation	16
9	Sign In Activity	17
10	Forgotten Password Dialog	17

11	Profile Fragment - Not Verified	18
12	Profile Fragment - Verified	18
13	My Projects Tab	19
14	Create New Project Activity	19
15	Select File Activity	20
16	Firestore Project Record	20
17	Project Removal Confirmation Dialog	21
18	Access Requests Dialog	21
19	Shared Projects Tab	22
20	Revoke Access Dialog	22
21	Pending Signatures Dialog	23
22	Signature Activity	23
23	Verification Snack Bar	24
24	Create Request Dialog	24
25	Search Results Example	25
26	Firestore Duplicate Title Field	25
27	View Project Activity	26
28	Changes Confirmation Dialog	26
29	Access Requests - State Diagram	27
30	Firestore Database - User Public Key	28
31	Cloud Verification Function - Logs	30
32	Ideation Verifier Application	31
33	Verification Passed Example	31
34	Project Trello Board - At Sprint 3	34
35	Project Gantt Chart - Final State	35
36	Project Management - Toggl Tracking (April)	47
37	Github Repository	48
38	Project Gantt Chart - Begin State	48
39	Project Gantt Chart - Interim State	49

1 Introduction

1.1 Overview

The path from an innovative idea to a profitable business is long and difficult [24], we aim to create a secure application that simplifies the process and removes restrictions that are known to slow business development down. Once we have created a secure environment for ideation, we hope to implement features that allow users to view each other's projects in such a way that IP infringement cannot occur. This project is a proof of concept for a worldwide start-up hub, like Facebook but for projects, where users can come together to share and develop new ideas without having to worry about intellectual property infringements.

1.2 Background and Motivation

"Powerful ideas are everywhere, and the vast majority will go nowhere" [19], in our opinion this is due to a whole range of reasons that boil down to poor execution. There are many sources that provide guides that try to help the user convert their idea into a real business plan but will not have the impact they promise to. Inexperienced entrepreneurs will often find themselves lost with no real starting point or structure, meaning they will struggle when starting to construct a viable plan (business model, target market, etc). This is something I personally experienced when starting my own venture, undergoing the excruciating process made me realise why so many fantastic ideas never leave the drawing board. Our project aims to improve this system, allowing powerful ideas to come to fruition.

"Success requires moving your passion quickly from the idea to the business implementation" [27]. For the most part this doesn't happen and for those that manage to push through and find themselves with an initial business model are limited when asking for advice, funding and hiring for their start-ups. This is due to security issues that complicate the distribution of well-documented ideas such as Repudiation and Copyright Infringement.

1.3 This Report

In this report we will be covering the research we did with regards to business development to ensure the framework we are providing will actually help the user capture the essence of their idea in a way where others can comprehend it. We then thought it crucial to explore similar work done within this field, in order to highlight strengths and weaknesses. This enabled us to summarise our findings, include a section discussing similar work in the market and shed light on what else we could add to the sector.

Upon doing this, we were able to make decisions with regards to selecting various security measures, methodologies, and design principles. The outcomes resulted in the use of E-Signatures and Asymmetric Encryption which ensure that the application is as secure as possible, both for storing and sharing ideas. Once we acquired the correct tools, we created various designs for both the UI and database, establishing the blueprints of what we planned to build. Next we began the application implementation and started integrating all the concepts and frameworks we had decided to use. The application was tested and documented throughout, but upon finishing we did some final tests and evaluations to assure software quality. Finally, we conclude the report by first discussing the project management techniques that were used, followed by a reflection on the achievements and contributions that were made, before acknowledging the downfalls and keeping them in mind for future projects.

2 Related Work

Ideation is widely used in many different sectors and simply means generating, developing and sharing ideas. A vital part of our research consisted of looking at what already exists and finding out what makes them successful. Doing this would allow us to consider about how we could bring them all together to best execute the project. This was done in three parts.

2.1 Existing Applications

- “Kickstarter PBC is a funding platform for creative projects” [23]. This is an application which allows users to display their ideas and designs in hope that passing users will donate towards their funding goal. This concept has performed exceptionally and has made it possible for hundreds of aspiring projects to come to life.
- LinkedIn is another existing application where users are able to connect, apply and hire. “Its interface caters towards the need for professional self-promotion” [25]. The platform can also be used to create a company page which people can either claim to be working at or alternatively apply to. It also allows the page owner to display branding, slogans and descriptions along with any posts they want to release to the public.

Both Kickstarter and LinkedIn are widely used and enable people to discover new businesses but neither of them has created an environment that allows innovators to freely share the sensitive details of their idea without having to worry about IP infringement. Due to this problems have occurred a notable amount of times “for instance, the success of Matthew and Mark McLachlan in raising over \$6.4 million on Kickstarter for the Fidget Cube was quickly copied and the Stress Cube was launched by an existing organization” [21]. Our application aims to eradicate this issue by ensuring that users wanting to know more about a project will undergo certain security measures to hold them accountable for any infringement they may commit.

2.2 Existing Protection Measures

- “A patent is supposed to protect intellectual property, something truly innovative” [1]. It boils down to an agreement that the owner will disclose a detailed report and description of his/her invention for the world to see and use in exchange for a fair legal ownership for enough time to make a fair deal of money from it. Patents have been used for hundreds of years and work but under the condition that the innovative idea has been completely thought through and finalised.
- Non-Disclosure Agreements (NDA) exist to tackle the issues mentioned and is legally binding, “companies and start-ups use these documents to ensure that their good ideas won’t be stolen by people they are negotiating with” [17]. This can be a great solution but is neglected as they can be quite intimidating to setup and needs to be signed in physical form. This is currently done in person at the start of a meetings but is not always convenient especially if the potential partner is not based in the same location.

As mentioned patents are effective ways of protecting a concept but in the case of our project they can’t be applied to every project due to the fact that we’re providing a platform for ideas that are yet to be fully developed which means that majority of our users will not be able to purchase a patent until they have completed developing their idea. So we will keep patents in mind as a potential stretch goal that allows users to mark a project as finalised, allowing them to attain a patent but this will not suffice as our main form of IP protection.

2.3 Existing E-Signature Platforms

We decided that an NDA is the best legally binding measure for the circumstances of our project but was limiting since it was a physical paper. Further research brought us to the concept of online signatures, which are still accepted by law [26]. We did some further research into companies that have already implemented this feature to see what they offer.

- A company called DocuSign already do this but has not been implemented seamlessly into an application to grant or reject access. To implement this effectively we'll have to build on top of the concept by implementing additional security measures to ensure that repudiation cannot take place, whilst not compromising the simplicity of the process.
- Another company called PandaDoc also offers secure digital signatures but simply as a service that sends signed documents, again it has not been incorporated into any specific application or project. A point worth mentioning is that they have complied with both ESIGN and UETA and also abide by the European Union Regulation which is a stretch goal that may not be within the scope of this project. But again similar to DocuSign and other companies we looked into such as "Contract Book" and "Secured Signing" they don't offer much more than the secure digital signatures for various documents.
- The closest a company comes to compromising our idea is called Hello Sign, this is because they aim to offer an API that can be implemented in company systems which would technically enable other applications in the business sector to create a similar application and just use the API, the drawback however, is that it's not free. This means the company that incorporates it will have to charge its users to maintain sustainability. Our application aims to be available to all users free of charge.

After doing a spread of research on the topic of how other companies are using digital signatures we have found there is no application doing anything exactly the same, especially in the sector, in that the technology exists but has not been implemented to serve the purpose of creating a start-up hub. In addition we still think that the proposed project has advantages over its e-signature competitors as at the very least we provide freely what other companies charge up to \$40 a month for, which makes it worthwhile pursuing and implementing into the proposed application.

3 Description Of The Work

In this project we aim to create a secure business development application which will encapsulate the process of ideation. This will provide the user with structure and guidance, allowing them to effectively document and develop their thoughts. The app will allow users to securely share their ideas with potential clients, advisors, investors, etc without having to worry about known issues such as repudiation and copyright infringement. The user wanting to access will have to read legally binding document and print their signature digitally before accessing any sensitive IP, doing so with a way of verifying the user identity will mean that we have incorporated non-repudiation.

Repudiation, simply meaning that a person denies something that they have done and refuse take any responsibility. this can be a big issue for projects such as this, as users can get away with breaking any agreements if there is no evidence held against them. For this reason we need to make sure we consider non-repudiation.

Definition 1 (Non-Repudiation). *Non-repudiation is the assurance that people cannot deny the validity of something due to measures put in place to hold them accountable.*

In addition to this the owner of the project will be able to select privacy options for their project, from how much of the project is shown without a request to what measures a user has to pass before accessing any sensitive information.

3.1 Key Objectives

- Implementing a secure database.
 1. This will keep account details and information about the user.
 2. It will also store a list of all the projects and their owners respectively.
 3. It will keep a log of all the requests and digital signatures.
- To create a secure business development application.
 1. Allow the user to register, login and create an account.
 2. Allow the user to create a new project.
 3. Facilitate and provide structure allowing effective idea documentation and development, providing a template.
 4. To create a platform where well documented ideas can be shared seamlessly and securely using various security measures.
 5. Allow users to view and sign an NDA through the app.
 6. Create a system that eliminates the possibility of non-repudiation.

3.2 User Requirements

Non-functional Requirements

1. The performance of the application must be reasonable (to avoid negatively impacting usability)
2. The application should be responsive to any interrupt (saving the state if the user tabs out)
3. The use-ability should be self-explanatory ensuring users can instinctively know how to navigate through the application without any expert guidelines or manuals
4. A level of reliability should be implemented (Not allowing users to accidentally make mistakes that could destroy a project)
5. The application should be secure with a reasonable level of encryption and other security measures to protect sensitive information within the app

Functional Requirements

1. Account Management
 - (a) The user must be able to create an account (personal details and password)
 - (b) The user must be able to login and access the contents of their account
 - (c) The user should be able view their profile details
2. Project Management
 - (a) The user must be able to create a new project
 - (b) The user must be able to develop the project from within the app (name, description, category)
 - (c) The user should be able to delete a project
 - (d) The user should be able to update sections of the project (using a provided framework)
3. Security Settings

- (a) The user must be able to upload an NDA file to protect their project
- (b) The user must be protected by measures that ensure other users are verified
- (c) The user could be able to sign and validate forms digitally

4. Project Discovery And Sharing

- (a) The user must be able to view other public projects
- (b) The user must be able to request access to other public projects
- (c) The user should be able to revoke access from projects

3.3 Software Requirements

1. Availability

- (a) The application must be developed for android users
- (b) The application should be free for everyone to download
- (c) The application could be available on the google play store

2. Performance

- (a) The application must not crash unexpectedly
- (b) The application should switch between activities swiftly
- (c) The application should allow the user to navigate to their destination within 5 inputs

3. Reliability

- (a) The application must save project progress if user exits
- (b) The application should be intuitive to use minimising mistakes and confusion
- (c) The application should prompt the user to confirm deletion of a project

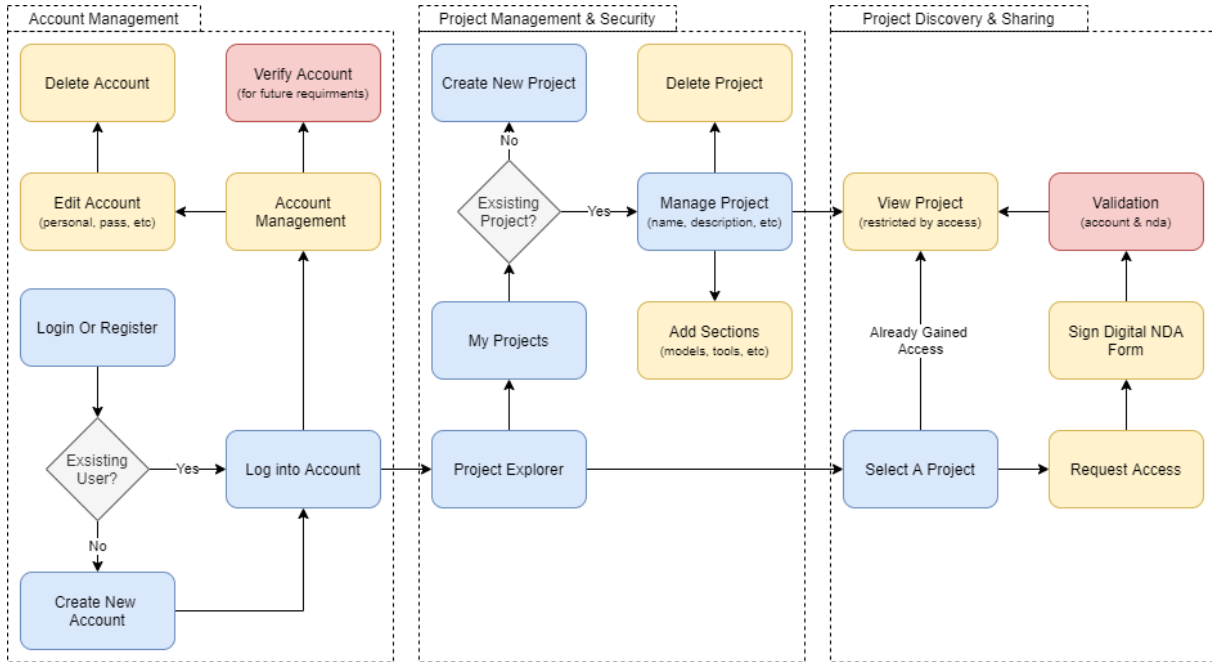


Figure 1: User Flow Diagram - Current Design

3.4 User Flow Diagram

We created a user flow diagram using the requirements defined above to visualise the process a user will undergo when trying to perform various tasks. We have used three colours throughout this diagram to represent the must (blue), should (yellow) and could (red) features in the application. The could goals are project stretch goals but play a big part in the more elaborate security aspects of the application, so if we are unable to implement them we will discuss them in the project limitations and consider how they could be implemented.

The advantage of creating this diagram is that it forced us to lock in some important design and workflow decisions which then help when creating the design prototype, for example where the user would be navigated to once logging into the application.

4 Methodology

The research done on effective business development, security issues and resolutions whilst liaising with the project supervisor has brought us to these project decisions and entail the following methodologies being used.

4.1 Integrated Digital NDA

As we mentioned in the related research, E-signatures already exist and are used by companies such as DocuSign. However by implementing the concept into an application as a security measure, before granting access to sensitive information, is what our application aims to do better. Our solution is as follows.

1. The user trying to access the project would need to login or create an account

2. They would then find the project and request to gain access to it
3. This then prompts the project owner which can then accept the request
4. The user would then be prompted to read and sign an NDA form digitally
5. Upon doing this they would be granted access into the full project

This formal process shows that there has been an interaction between the user and the project owner which is good but needs to be reinforced with some verification. To do this the user will need to create their account with a real email address and verify it before trying to access any projects.

Having this all implemented will mean that the system is pretty secure but is left exposed in one specific case. If the user agrees to everything but repudiates to having signed the document, claiming that it's not legitimate. This may happen in very rare cases but it's important to cover this contingency, as the data stored could be very sensitive, our approach was to consider implementing another methodology which would prove that the user has in fact signed the form, and to ensure unauthorised user cannot sign in other people's names. The method we will be using is known as Asymmetric Cryptography.

4.1.1 Asymmetric Cryptography

This is a method that companies such as DocuSign will use to ultimately prove legitimacy of the signed documents against people that try to repudiate [7].

- The Signing Process
 - It works by taking a mathematical summary of the NDA known as a hash code, it's basically an identifier, for the document (similar to a fingerprint). When a user requests to view a project and sign the NDA document the system will sign the document with the project owner's private key, this signed hash code (signature) will then be stored securely in the request logs with a corresponding time stamp.
- The Verification Process
 - Once the document is received by the owner, they will be able to verify the signature sent by applying the requester public key. To do this a new hash code will be generated from the received NDA document. If the two signatures match then it is evidence that the document has been signed and not been altered, as only the requester will have had the correct private key to encrypt the initial document. This is important as being able to prove that it hasn't been touched since the user signed it leaves them no room to argue against its legitimacy.

Android has a section of implemented encryption methods and many libraries and tutorials exist; these can be used to help implement the method into the application which will minimise the chance of any vulnerabilities that could compromise the system. This method can seem quite complex, but our goal would be to make this process happen quietly in the background, it will then only be referred to in necessary circumstances.

4.1.2 Android Frameworks

After getting an sufficient understanding of asymmetric encryption and how it enables the use of digital signatures, it was important to do further research and find effective ways of implementing the concept into an android application. With the help of a very helpful guide to secure app development [6] we found the following.

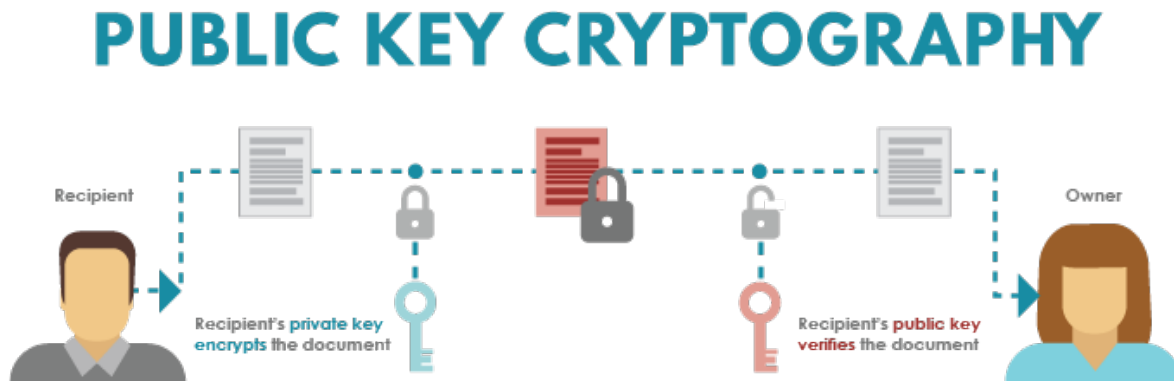


Figure 2: Asymmetric Cryptography - Visual Representation [2]

- Bouncy Castle (Deprecated)
 - Initially we looked into a framework Android had adopted called Bouncy Castle. This library provided many tools to perform all types of encryption, this included the key pair encryption we required for the project. Unfortunately after looking at the Android documents we found that it had been deprecated and replaced [11]. A new library provided by Android offers the same tools but with better algorithms and seem less integration with Android studio.
- Key Pair Generator
 - After doing some simple private key encryption and decryption we looked into Key Pair Generation which is a series of functions that enable the creation of matching public/private key pairs [14]. This tool was almost perfect for the key generation stage of our implementation, but we quickly realised that by acquiring the private key we have presented a legal exploit to any malicious users when fighting the argument of repudiation, as they could argue that we failed to keep their private key secure and it was compromised resulting in signatures being created in their name by other people.
 - To tackle this issue we had to research further and find a solution which ensures that the private key is made and kept solely by them and is stored locally on their device, this would mean that only they hold possession of the key and could sign in their name. We on the other hand would store the public key to verify any signatures made, we would also use it in cases where past signatures are required to be verified as evidence.
- Android Keystore
 - “Android Keystore mitigates unauthorized use of key material outside of the Android device by preventing extraction of the key material from application processes and from the Android device as a whole” [9]. This security measure was ideal for the issue we faced and supported key use for cryptography which meant it incorporated features of the android encryption library that we required, such as the key pair generator.
- Producing Digital Signatures
 - Before creating a signature the document needs to be hashed by an algorithm, for this purpose the “SHA256” algorithm would be sufficient in creating a secure message digest. This will then be used when creating the signature.
 - Once we have the message digest, we will sign it with the “RSA-SHA512” algorithm using the users private key.
 - Finally we will verify the signature using the public key to confirm the integrity of the private key that was used, if it fails the user will not be granted access. This will ensure that the verification will pass at any point in the future if evidence is needed.

4.2 Business Development

- The business model canvas is a popular tool that's used to make the user think about different areas of their project to whilst documenting it down in a clear format [5]. We don't plan on adopting this model in its entirety but will use it as guidance for the projects our users create. Implementing its ideology into an application that new aspiring entrepreneurs will be using introduces them to an effective method that they may not have considered otherwise.
- Another interesting model that we will implement concepts from is the SWOT analysis, it's a tool used to evaluate the strengths, weaknesses, opportunities and threats of an idea or project [16]. This is another angle of thinking that seems quite basic but is overlooked, having it as an optional framework will make the users consider it and if implemented will be appreciated by any prospect investors that are interested.

A con of these models is that at first look it can be quite overwhelming, to tackle this issue we will allow users to add sections to their projects once they are ready. For example once the user wants to start thinking about their target market, they can add that section and start filling it out, this is similar to the process that LinkedIn provides when users are adding information about their career.

5 Design

Before making any decisions about our application design we looked at a range of applications and did some research into design concepts that would help us create something aesthetically pleasing but also systematically functional. In this section we will describe the main design decisions and considerations we made and then go onto explaining how we implemented them.

5.1 User Interface

In terms of our UI approach we have decided to take a minimalist stance. This consists of creating a design that has a limited palette of colours and uses simple geometric shapes [18], this will generally ensure that key components of a given page stand out and are not clustered with anything that may distract the user. We prefer this approach as it helps create a professional look that doesn't compromise the aesthetics.

5.1.1 HCI Research

- A focus of mine was to create an application which is intuitive to use upon opening it up for the first time, this done by considering a concept known as interface metaphors. The concept explains that people are familiar to seeing commonalities between applications, this allows them to learn new but similar applications easier and quicker than they would if it were alien to them. So meeting these expectations in design will in most cases make the user experience more fluid and will reduce the gradient of the learning curve [4].
- Another important concept that we considered is Gestalts principles of proximity, the general idea is that user will expect components to be located near to other similar components. An example of good practice would be to have all navigation options in one section so getting around the application is instinctively quite easy, even for that may not be frequent mobile users.



Figure 3: Initial Design - Login Page



Figure 4: Initial Design - Navigation Menu

5.1.2 Initial Designs

We brainstormed in pursuit of a good logo and decided to go with a simple light bulb as its easily recognisable and relevant to the name and project (Ideation) in that light bulbs are associated with new ideas. We then decided on the font and colour palette. The font (Roboto) we chose follows the minimalist approach meaning it's simple and clear to read, it's also the standard android font which means helps with familiarity. The main colours we chose (White, Blue and Black) make the application feel professional, this makes the user more likely to trust the application with their project idea. This house style complies with the expectations of the target market which consist of business owners or aspiring entrepreneurs.

Consistency is another design concept that we have chosen to follow, by having a congruent layout and an aesthetically pleasing interface the user will become familiar with the application and its features a lot sooner than if the design varied massively from section to section.

The login page we designed incorporates the research and decisions that we ended up making. It consists of a login section which is simple and intuitive as it's similar to other application login screens.

The next thing we designed is the navigation bar and its available options. The navigation bar is consistent throughout the application and is what the user will mostly use to get around the application. This is an initial design and may be implemented slightly differently but the concepts discussed will be kept in mind and applied.

5.2 Database

5.2.1 Infrastructure Research

The initial plan was to design and implement a relational database which was hosted live on AWS (Amazon Web Servers), as we had lots of previous experience with SQL and relational databases. We started out by roughly developing the tables so that we could get an idea for structure.

Once the initial design of the relational database was complete we did some research on connecting an Android app to a live hosted database, we found that it's doesn't have much compatibility with AWS and would require an API that communicated with a server. After doing some extensive research we found an alternative called Firestore, developed by google, which is a "flexible, scalable NoSQL cloud database to store and sync data for client- and server-side development" [15]. We chose to use this database infrastructure as it would provide seamless integration with android studio and the following features:

- **Authentication** - A feature which provides a library that communicates with a back-end service with an interface which allows for secure user authentication. This can be done using an email address and password or even accounts from other platforms such as Gmail, Facebook, Twitter, and more. This feature also ensures that the user's password is safe by taking the credentials and passing them to the Authentication SDK which will run a verification and return a response to the application. The password will be encrypted and stored away, only being used for future authentication. [12]

Uses - This feature is perfect for the account creation section of the application, the ability to safely store credentials and have user authentication is key when identifying users and keeping them accountable to their actions. We can also use this feature to send an email verification to the user ensuring that accounts are not spam.

- **Database** - Following the NoSQL data model, data is stored as fields in documents. The documents are then contained in collections which act as a container that allows you to organise, filter and manipulate the data. To get a piece of data you would navigate to the correct collection, find the relevant document and query the field the data is mapped to. [15] This structure is very different to relational databases but allows for easy and effective integration into android studio.

Uses - A live database will allow us to store all the user and project information and correlate the data to the users we have authenticated and verified. "Users" documents will be created for each authenticated account holding information about them and will also store their public key. A collection of project documents will not only store the project details but also information about who owns it, the access logs for that project and also any signatures received.

- **Cloud Storage** - This is a storage bucket that we can securely upload and store files to directly from mobile applications, files can range from simple documents to rich media such as photos and video. Once uploaded we can retrieve them using the application stack to load profile pictures, play videos or present documents [10].

Uses - We can use this feature to store the project NDA files. They will be uploaded to cloud storage when a user creates a project and provides an NDA form, the file will then be retrieved when a user is required to read the document and provide a signature.

- **Cloud Functions** - As the Firestore database is provided as a service, cloud functions are implemented as a feature to allow the developer to add functions to the back-end which will automatically run when called. The event that triggered the function will then receive a result based on its request.

Uses - This feature will be used when something needs to happen outside the user's scope for security purposes, an example of this is the signature verification. If the verification didn't happen on the server a skilled hacker could exploit the source code and skip the verification step. Using this feature it would mean that the code would only continue if the request is sent to the server and the result returns true.

Using Firebase to host the live database had one drawback, it was strictly NoSQL. Having had no experience with non-relational databases, meant that we had to undergo a very big learning curve to understand its theory, structure, and the programming used to operate it. Within Firebase a new, more efficient, branch had been developed called Firestore which is what we planned to implement as the back-end of the application.

5.2.2 Schema Design

Upon studying the Android documentation for Firestore and watching various tutorials on YouTube we started to convert what we had initially designed to be our relation database into the Collection/Document structure that NoSQL implements. In order to do this we had to consider the tables, fields and their data types respectively and construct an efficient NoSQL database design that would function efficiently when being queried. Fortunately Firestore's steep learning curve was worthwhile and enabled us to design the following database schema.

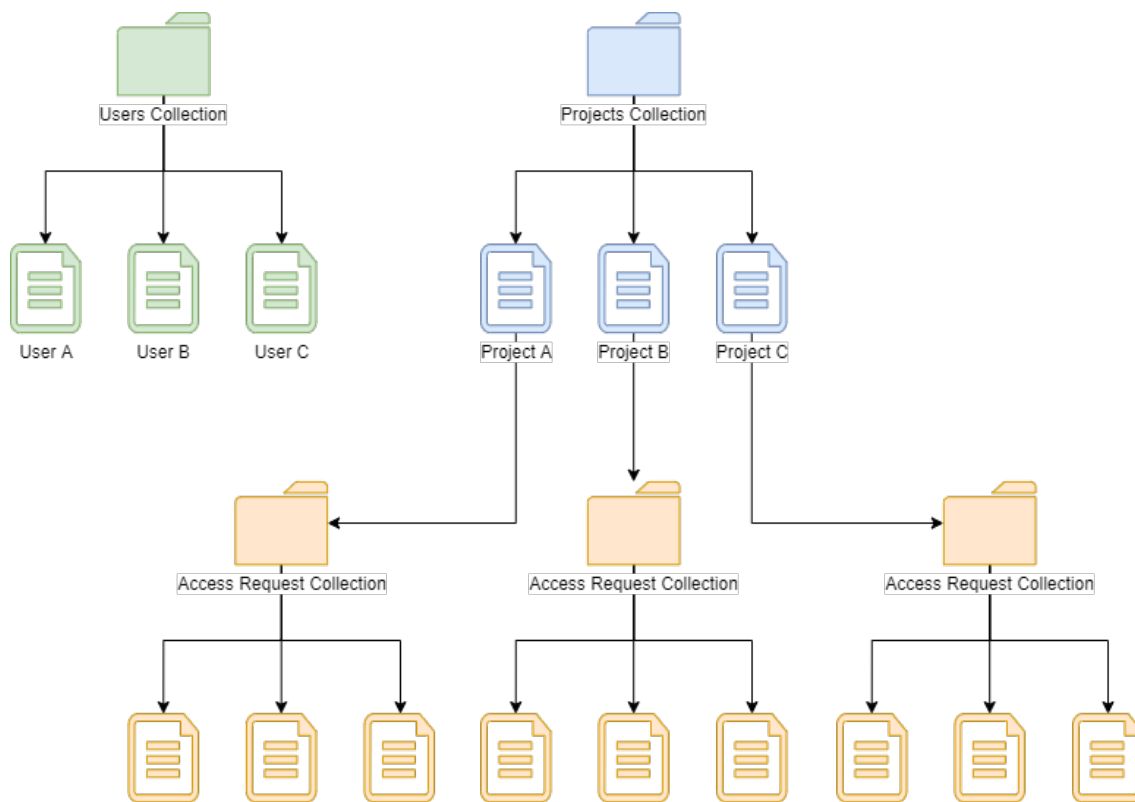


Figure 5: Database Schema - Collections and Documents diagram

This would consist of all the users, that had signed up to the application, unique documents with ID's inherited from their accounts in the Authentication section of the Firestore framework.

As we can see from the diagram the schema consists of three unique collections. The “users” collection will hold a document with a unique ID, inherited from their accounts stored in the Authentication section of the Firestore framework. The fields inside will map to all their account data. The “projects” collection will hold a document with a unique ID for every project created, the fields will map to the projects details and a reference will be held to its respective access request collection. This will hold all the request documents, again with a unique ID, which will contain information about the interest other users have shown to that project and the details of their request.

5.2.3 Document Field Properties

Name	Type	Description	Example
FirstName	String	The users first name	John
LastName	String	The users last name	Doe
Username	String	A username that will be displayed to others	JohnnyDonny64
PublicKey	String	The public key generated upon account creation, used to verify signatures made by them	MIIBIjANBgkqhkiG9...

Table 1: User Document Fields

Name	Type	Description	Example
Title	String	The project name chosen by the user upon creation	Trendy
DateCreated	Timestamp	The project creation date	25.04.20 at 6pm
Category	String	The industry sector the project falls in	Technology
Description	String	A brief description of the project, such as an elevator pitch	Our company does A to solve B in this way
OwnerName	String	The owners username	JohnnyDonny64
OwnerID	String	The owners unique ID	a4589G98efe8Ucidsa
NDAPath	String	A path to the NDA form if cloud storage	/NDAForms/Trendy.pdf
Whitelist	Array	An array containing all of the user ID's that have been granted access	0:a4589G98efe8Ucidsa 1:cE12Dncio4ji23kfw2
TitleSearch	String	A non-capital version of the title for searching	trendy
Archived	Boolean	A flag representing if the project has been archived	true
ProblemSolution	String	The problem at hand and the projects solution for it	Waste ruins the planet, we recycle it into accessories
RevenueModel	String	How they plan to make an income to stay sustainable	The waste materials are free and so the sales will all be profit
TargetMarket	String	Who they aim to sell to	All age ranges, aimed at the adventurous people

Table 2: Project Document Fields

Name	Type	Description	Example
UserUID	String	The requester unique ID	vCk043Ff4f4faSF4vi
UserName	String	The requester username	BobMane12
Project	String	The name of the project that's being targeted	Trendy
OwnerUID	String	The owners unique ID	a4589G98efe8Ucidsa
OwnerName	String	The project owners username	JohnnyDonny64
DateTime	Timestamp	A timestamp of when the request was made	25.04.20 at 2pm
NDAFlag	String	A flag to show if this projects has an NDA to be signed	true
Reason	String	Why they want to access this project	I'm interested in investing
Status	String	States that show the requests current progress	Signature Pending
Signature	String	The NDA signature made using the users private key	C3EYJvpB4ASr7de...
SignDateTime	Timestamp	A timestamp of when the NDA was signed	26.04.20 at 2pm
Applicable	Boolean	A flag representing if user can request access, false if they have applied. Turn to true if they revoke their access	false

Table 3: Access Requests Document Fields

The users table is quite simple but this is due to storing much of the data such as their credentials, password, and verification state in the authentication section corresponding to their unique user ID. Administrating the accounts within the authentication system can be done using the Firestore online interface, this allows us to add, remove, disable, and reset accounts if necessary along with an email system where we can set and customise emails that are sent when we provide a verification or reset password email.



Figure 6: Firestore UI - Authentication Panel

The current schema results in the project and access request documents holding lots of fields, in some cases they even contain duplicate data, but this is done for a couple reasons. One reason is that there are no primary or foreign keys in the noSQL system so the only way to create a link between the user and project is by duplicate data to identify links. Secondly the fire store query system can become quite taxing in terms of time if it needs to fetch a whole document to retrieve one field. So by having some duplicate data where necessary, we minimise request buffer periods, making it worthwhile. After doing some research we found it necessary to have duplicate data in a noSQL schema and is simply one of its drawbacks.

5.2.4 Data Persistence

Finally for reliability we aim to make our database support offline persistence, this can be done via the fire store infrastructure to ensure data is handled correctly when the users network fails unexpectedly. By configuring the application the user will be able to “write, read, listen to, and query the cached data. When the device comes back online, Cloud Firestore synchronizes any local changes made by your app to the Cloud Firestore back-end.” [8] An example user case would be if the user is offline and creates a project holding an NDA form, the request will be stored locally and uploaded when a connection is available.

6 Implementation

The application was broken down into features and implemented using an agile software development technique, this consisted of a series of sprints where features were taken from the backlog, week by week, and completed incrementally. In this section we will discuss the details of how each section of the application was created.

6.1 Launcher

The application icon has been customised to give a professional first impression, on being clicked the first page seen will be the splash screen. This gives the application sometime to load in the background but also is an aesthetic touch which helps the user quickly get familiar with the brand [20]. Once loaded we need to consider what screen the user should use based on previous usage, if they have previously logged into the application it should take them straight into their account. So we have implemented a check to do this using the firestore authentication feature. Given they are a new they can easily navigate to account creation using the “sign up” button, alternatively if they already have an account they can simply login.

6.2 Registration and Authentication

To register the user must provide some personal information for identification purposes. A first name, last name, and username is required along with a valid email address and password. To ensure that the password has been entered correctly the user is required to confirm the password by entering it a second time. Data validation is enforced to make sure the user fills all fields correctly. Examples of this would be empty fields, invalid or already used email address, and not matching or weak password. Once the credentials are provided the user will complete the account creation by clicking the “Create Account” button.

Upon being clicked the application sends a request containing the provided credentials to Firestore using an API, if it passes the user will be navigated back to the Login page and the account will be added. An email will automatically be sent out asking the user to verify the email and complete the account creation process. In the background a key pair will also be generated, and the retrieved public key will be stored in the “users” document along with their personal data, this will be discussed in more depth in later in the report.

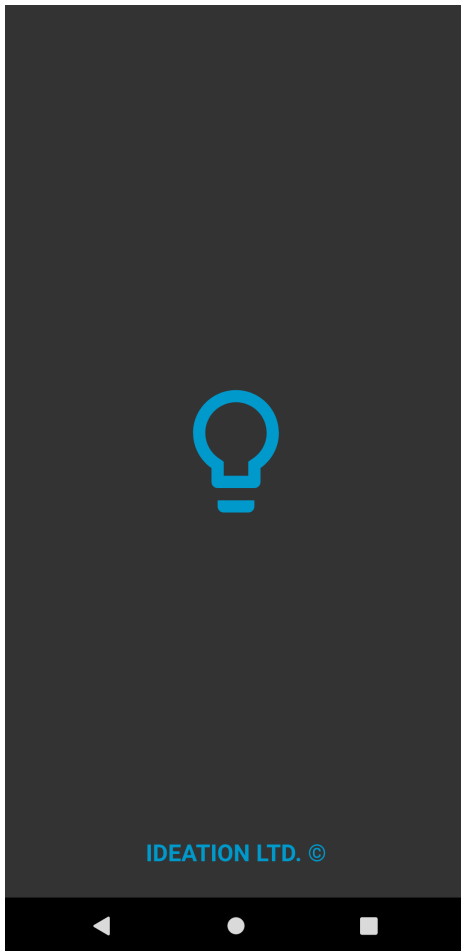


Figure 7: Application - Splash Screen

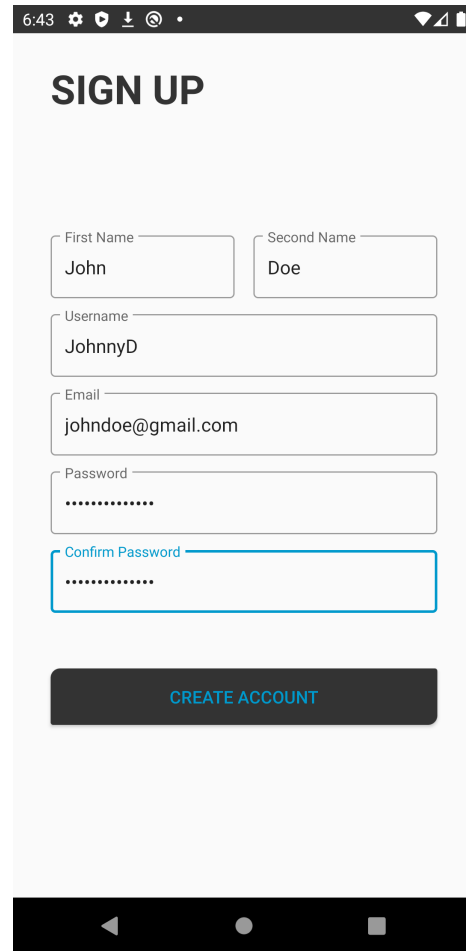


Figure 8: Application - Account Creation

6.3 Login and Forgotten Password

The account login consists of the same input validation and is used in the sign-up section of the application, the fields must be full and the email address provided must be on the server as an account. Failing to provide the correct credentials will simply return an error message. Upon logging in for the first time the loading animation will appear while the request is being processed and once complete the user will be navigated into the application to the discovery screen. In the background the server will check if it's the first time the user is logging in, if so it will create their user document with the information it had received on sign up and store in the the “users” collection within the database.

As we are storing information that may be of value to the user within the application it important that we have a “forgotten password” feature that allows them to reset their password using their email address, so that they are not locked out from their account and projects. Upon clicking the forgotten password text located under the “login” and “sign up” buttons a dialog will appear asking for an email address, this will check the records and if that email exists the user will receive an email where they can set a new password.

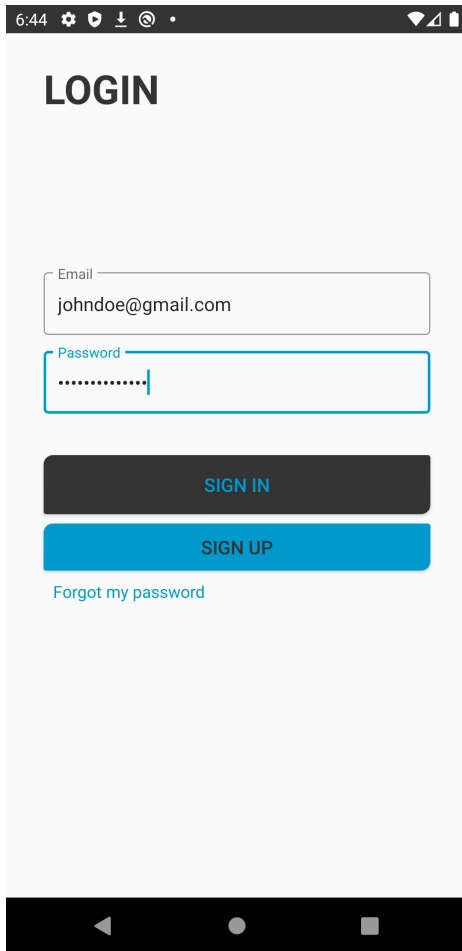


Figure 9: Sign In Activity

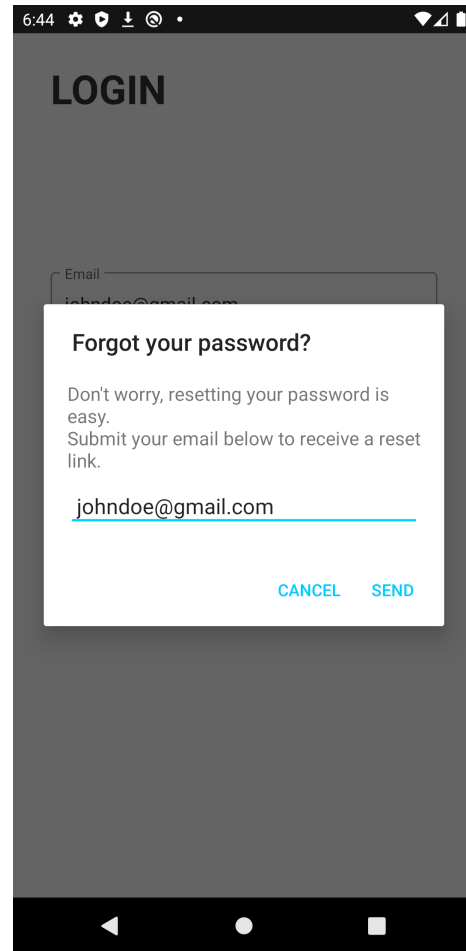


Figure 10: Forgotten Password Dialog

6.4 Navigation System and Fragment Optimisation

Once the user has logged in they will be navigated the discovery page and will have access to the navigation bar, they can use this to explore the different sections of the application (Discovery, Projects, and Profile). It's clear that the navigation bar is different to the one we had initially designed, this decision was made in the implementation stage due to the elaborate libraries provided by android for the bottom navigation bar, and what it enabled us to do.

- **Why** - Naturally the bottom navigation bar uses a fragment system which is far more efficient than what we had originally designed, using activities (pages) which would be created and destroyed every time the user switches between sections. A taxing process which requires the application to send requests more often to the server which is not ideal from a security standpoint as it creates a larger window for a malicious user to exploit. Alternatively Fragments live as a part of one activity and can be optimised to create quick and effective navigation.

How - To implement and optimise this system we first created the 3 different fragments, one for each section, and stored them in a fragment manager. By doing this the fragments are initialised and kept open, when the user navigates from one fragment to the next the application simply hides all the fragment views and shows the one that has been selected. Using the android shared preferences feature we can save the navigation location so if the user minimises the app the navigation location will be saved and restored from memory when the user brings the app back to the foreground.

6.5 User Profile Section

The profile section of the application sends a request using the firestore API which returns all the user's information from the database and presents it to the user. An active listener is implemented to check for the profile's verification status, if the server returns false the verified text field will read "Not Verified". Until the account is verified a button will be visible allowing the user to resend the verification email in case the one sent on account creation has expired or just failed to reach the user for whatever reason.

The logout button is also located in this tab as we wanted to follow the interface metaphors concept discussed in our design research, users will instinctively navigate to the profile section when looking to logout of their account. We will test to see this works and is intuitive when we perform our UX test later in the evaluation section of the report.

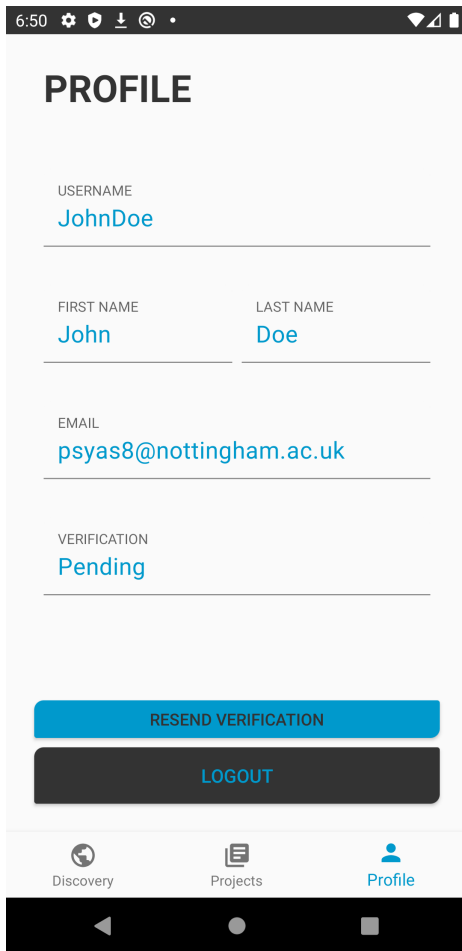


Figure 11: Profile Fragment - Not Verified

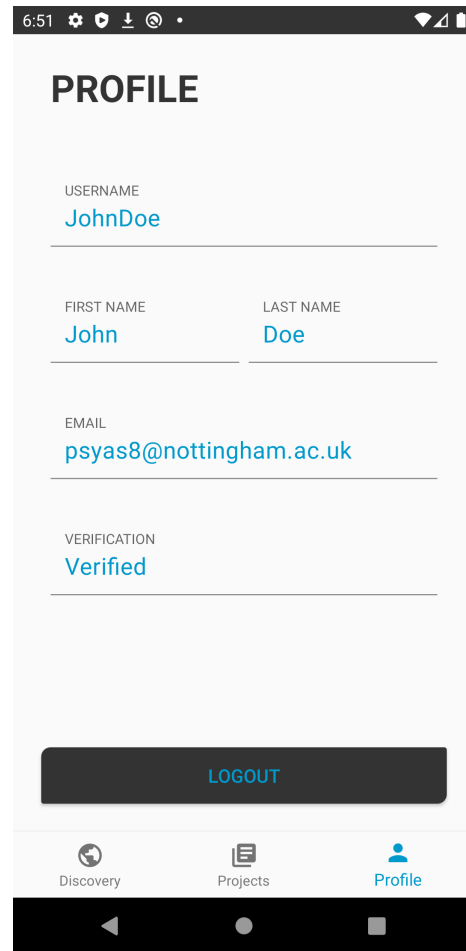


Figure 12: Profile Fragment - Verified

6.6 Projects Section

This tab is empty with a view pager and a tab selector located at the top of the screen, this allows us to show multiple views and swipe between them. It encapsulates the "My Projects" and "Shared Projects" tabs. This was implemented to separated projects that the user had created and owns from the projects that they'd gained access to and a few other features, the content in these tabs load and restore similarly to the navigation tabs, meaning there are optimised and quick to respond. They operate as follows:

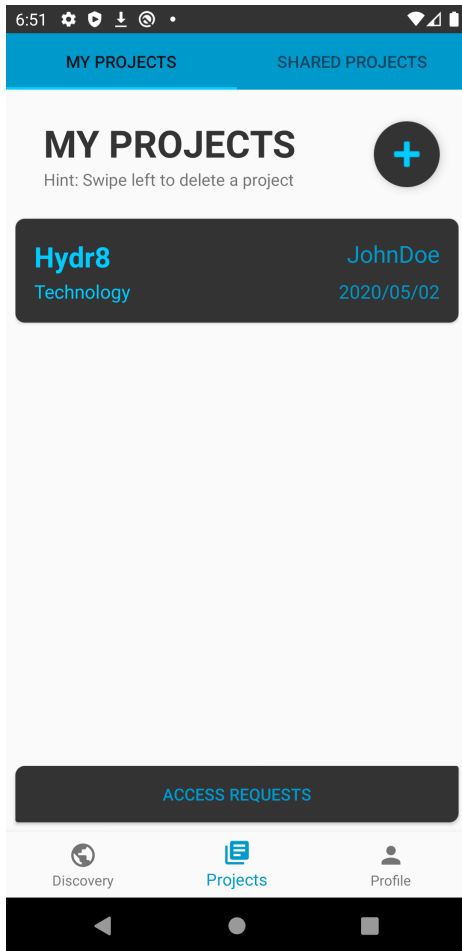


Figure 13: My Projects Tab

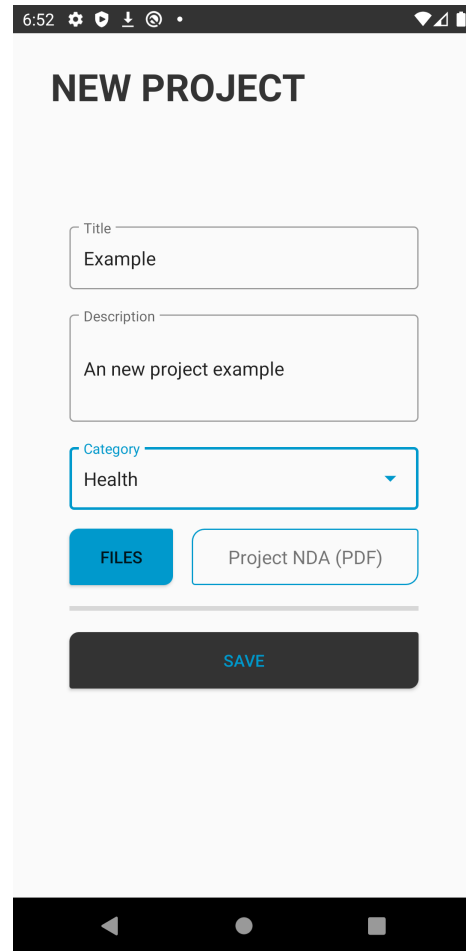


Figure 14: Create New Project Activity

6.6.1 My Projects Tab

This tab is responsible for displaying new projects, allowing the user to create new projects, and also allowing the user to view and interact with their access requests.

- Creating New Projects

- **Project Creation:** Located just right of the title we have placed the create new project button, if the user clicks this a “new project” activity will be made. This page consists of the several fields (e.g title, description, category) that the user must fill out to create a new project, this is to ensure the projects uploaded to the database have some content and are not just empty skeletons. To assist the user we have added a drop-down feature to the category feature allowing to either select some of the preset categories or just write their own.
- **NDA Upload:** Finally before the project is created the user has the option to provide an NDA form as a an extra layer of security ensuring that those that are granted access are also liable for the terms stated in the document, this will be done due to our digital signature feature which will be discussed in great detail later in the report. To provide a file the user can click the files button which will open their device files, by selecting a PDF file the upload request will be made and the selected file field will display the file name to reassure the user that the correct file has been selected.

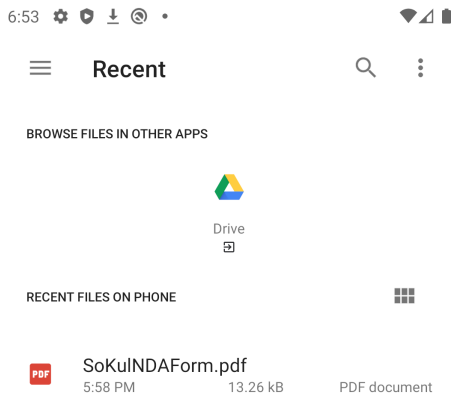


Figure 15: Select File Activity

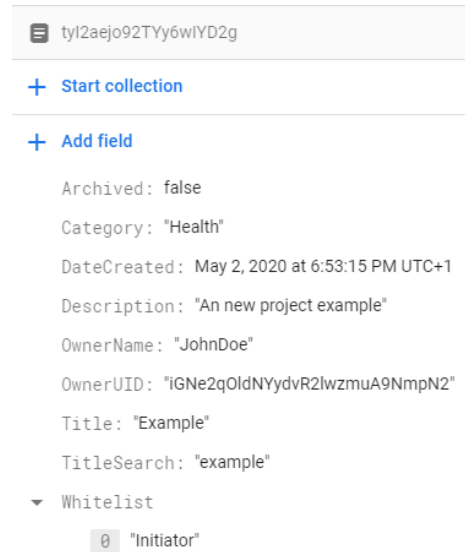


Figure 16: Firestore Project Record

- **Updating Database:** Once the said steps are complete the user can progress and create the project. a quick validation will be done ensuring the fields have all been filled, and then the information will be extracted from the views and uploaded to the database. An NDA check will occur to identify whether or not an NDA file path needs to be made. Given a file has been selected it will be uploaded (progress bar will show completion) into the firestore storage bucket, once complete the project will be added to the “projects” collection in the form of a document along with all its data and a path to the NDA file in storage. Once complete the application will navigate the user back to the “my projects” activity where the new project will appear. If anything fails an error message will appear notifying the user what needs to be fixed.
- **Existing Projects**
 - **My Projects List (Recyclerview):** In order to retrieve the projects them we had to query the database for all the projects that had an owner UID that matches the UID of the our user and retrieve them. Then to display the on the screen we used a “recyclerview” which loads and shows items on the screen in a scroll view, a clever feature it has is detecting the screen size and only loading in as many items that can fit. Each document in the collection that’s being queried will have its own item and how the data is utilised will be defined using an adapter class. So we created a card view that displays some important information about the project, such as title and owner, and populated its fields using the adapter we defined. A notable feature is that the recycle view is constantly listening for any changes made to the database and will update to show new items when new documents are added.
 - **Accessing Projects:** Once we had the documents (projects) appearing in the recycle view we set an on click listener which opens a new “View Project” activity which would retrieve the project information and display it all on the page, this will be discussed further in the view and edit projects section. But note that opening project in this way is only possible as the item clicked will pass through the project UID, we then use this to retrieve the specified projects information from database.
 - **Removing Projects:** Finally we implemented the remove projects feature by allowing the user to swipe the project box to the left, to confirm that it was an intentional action a dialog appears asking them if they are sure, as the data in these projects could potentially be of value and so measures needed to be implemented to ensure they’re not deleted accidentally. Once confirmed a

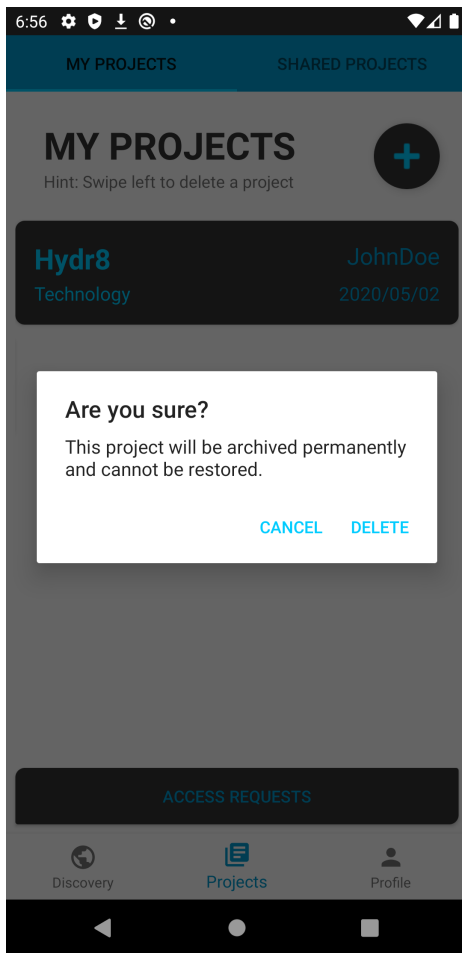


Figure 17: Project Removal Confirmation Dialog

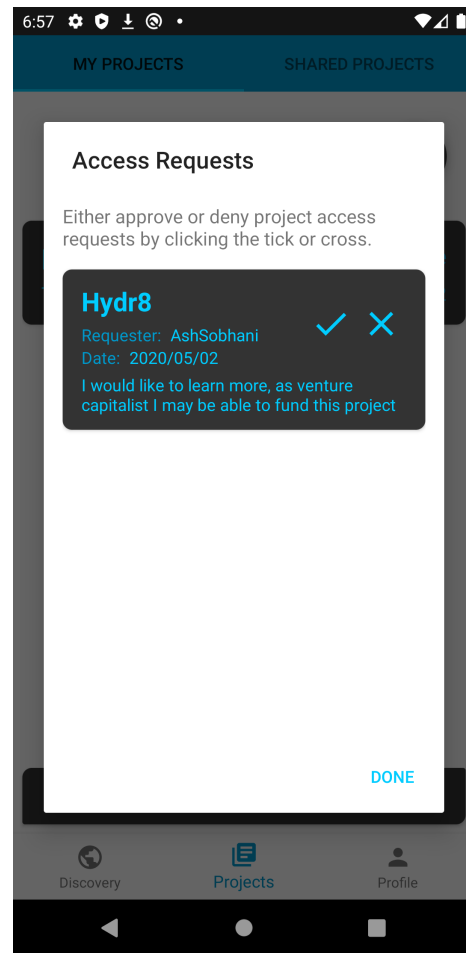


Figure 18: Access Requests Dialog

request will be made to set the project status to archived, this will remove the project from all views across the project meaning that everyone will lose access to it.

- Managing Access Requests

- **Retrieving Requests:** The final feature in tab is the project access requests which can be accessed using the button at the bottom of the page, once clicked a dialog appears that contains a recyclerview (explained in section 6.6.1). This is populated by a database query which looks across all the owners project documents and retrieves the requests, in the access requests collection, that are awaiting a reply. The items display the target project, the requester name, the request date and a reason if one was provided.
- **Owner Response:** To respond the user is given two options in the user request card using a tick and cross to, either decline the request, or accept the request to either grant them access by adding their user UID to the white list and updating the request status to “access granted” or, if the project is protected by an NDA form, require them to read and sign it before granting them access. Once a decision is made the database will be updated accordingly and the request card will disappear.

6.6.2 Shared Projects Tab

This tab is responsible for retrieving and displaying the projects the user has gained access to and also provide them with a list of signatures they need provide to complete any requests that are still processing.

- Retrieving Shared Projects
 - **Shared Projects List:** Retrieving, displaying and accessing projects is again done with a recyclerview system (section 6.6.1), the difference being in the query. For this case we query the database for all the projects that have the users UID in their white list. Once the data is retrieved, the items will populate and allow the user to open the projects by clicking them.

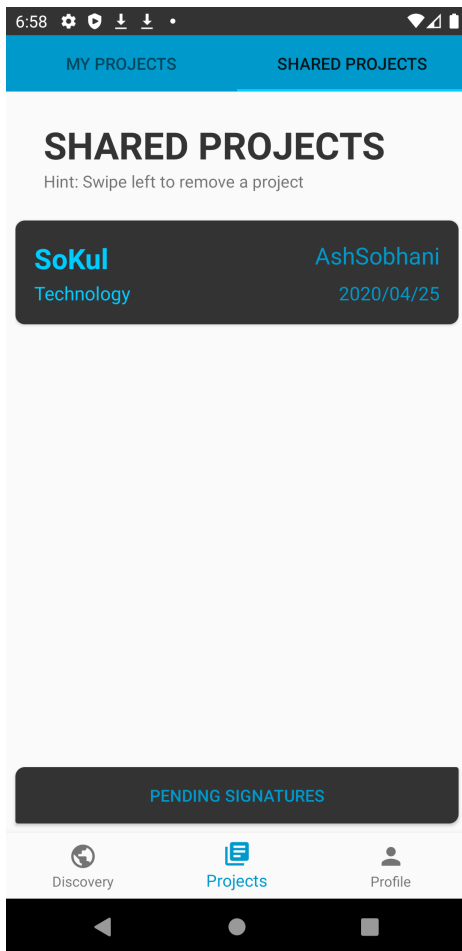


Figure 19: Shared Projects Tab

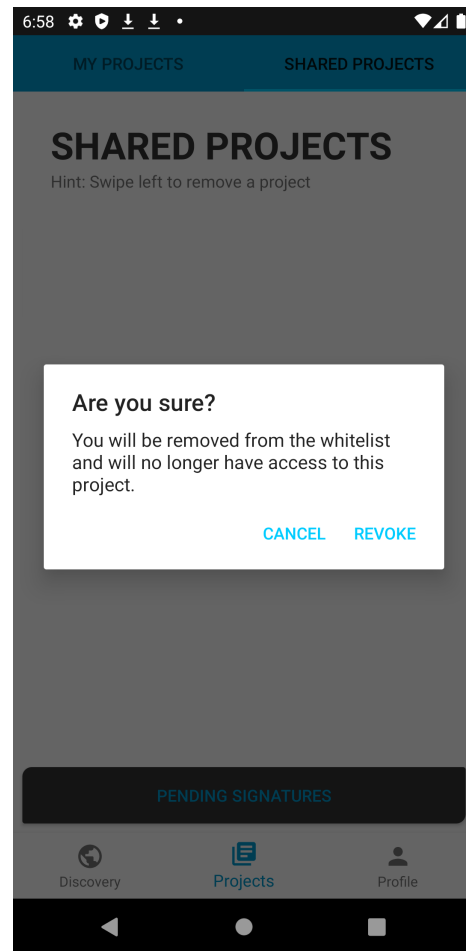


Figure 20: Revoke Access Dialog

- **Revoking Access:** Similar to the my projects section, projects can be removed from this list swiping the left but in this case an API will update the database telling it to remove the user from that projects white list and require them to reapply given they want access again, this will also double check if the user is sure before revoking their access to ensure it was an intentional action.
- Managing Pending Signatures
 - **Retrieving Pending Signatures:** This feature is very similar to the access request dialog in the my projects page (section 6.6.1), but instead queries the database for the requests made by the user that are in the “pending signature” state (states will be explained further in the Request

State System). Once the data is retrieved we load it into the signatures adapter which has its own style as can be seen in the figure. It creates an item for each document and displays the project title, owner name, and the requests creation date.

- **Open Signature Activity:** To interact with signature the request items have an on click listener which navigate the user to the projects NDA digital signature activity. This page explains that the project is protected by an NDA and requires the user to download, read and sign the document before being granted access. At this stage the user has 3 choices:
 1. Decline the conditions and close their access request
 2. Return to the previous page and respond later
 3. Read the NDA and agree to its terms

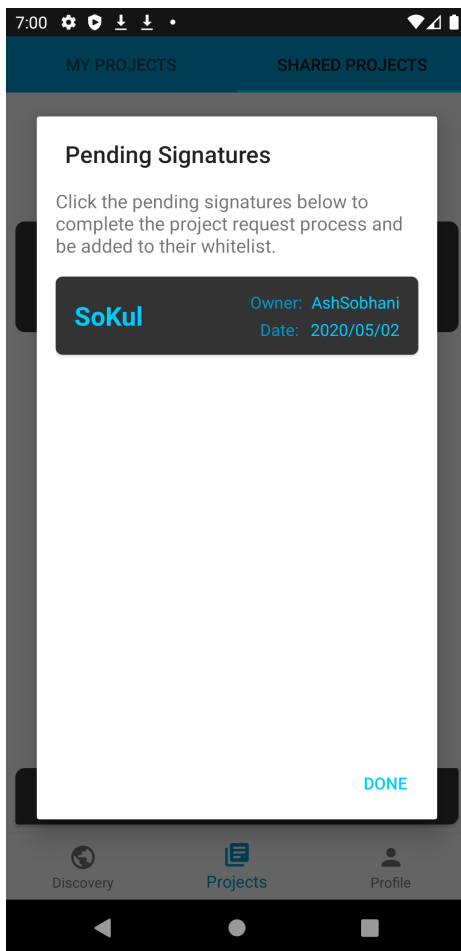


Figure 21: Pending Signatures Dialog

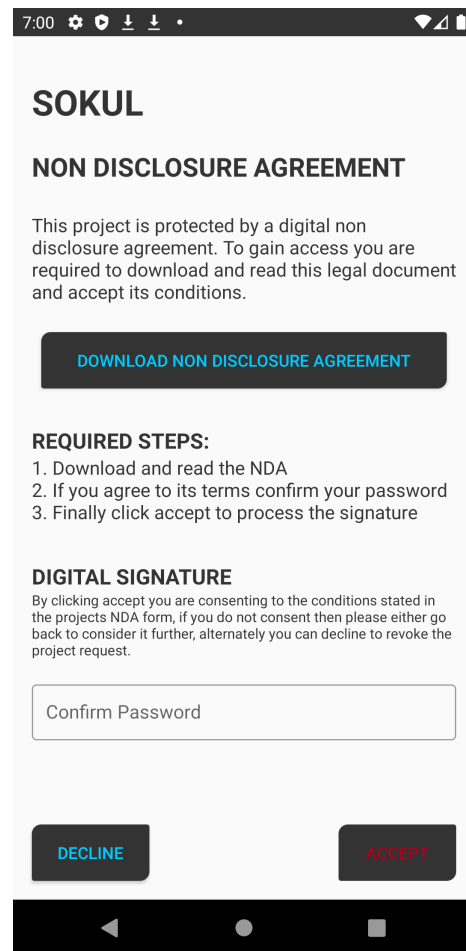


Figure 22: Signature Activity

- **Pending Signature Response:** If the user declines, their access request is closed and logged and the status is changed to show they declined the signature. Alternatively if they want to accept and be granted access they can follow the steps displayed. Once the NDA has been downloaded the agree button will become active and will only process if the user correctly confirms their password credential. This will then send an API to have the signature verified and if that passes the signature and a corresponding timestamp will be stored within the request record. Once complete the user will be added to the project white list and granted access. (The signature and verification will be expanded on in the Digital Signature Section)

6.7 Discovery Section

- Projects List
 - **Retrieve Projects:** First we query the database for all the projects in the database, excluding the ones that have been archived. This query is then executed and its data is represented as project items like the ones user in the projects implementation (section 6.6.1). It also adopts the on click function that creates a view project activity, but before doing so performs a couple checks to ensure the user is both verified and has been granted access.
 - **Verification Prompt:** The verification check occurs first ensuring that the users account is not just spam and is associated with a real email address and identity, if the user hasn't verified their account a snack bar (notification bar) will pop up from the bottom of their screen asking them to verify their email before trying to use this feature.

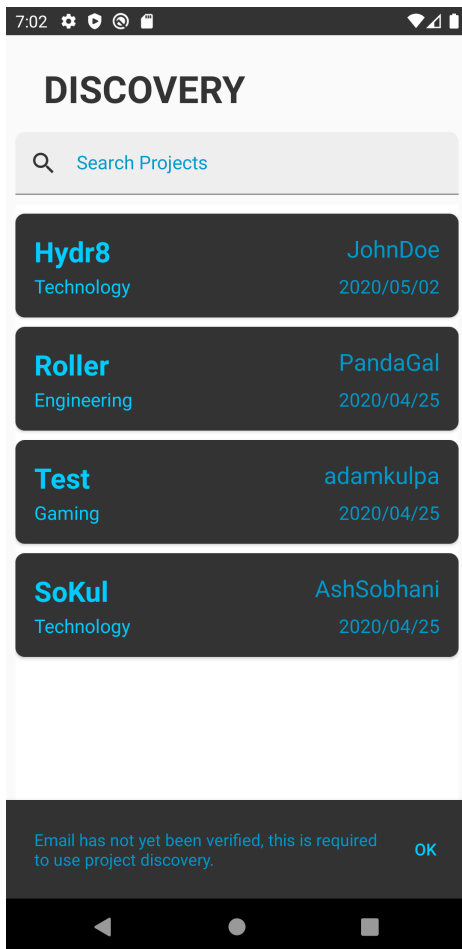


Figure 23: Verification Snack Bar

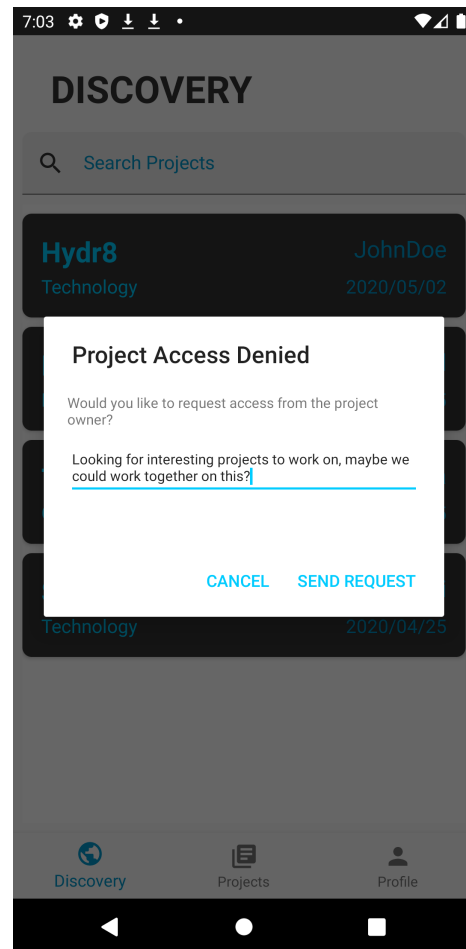


Figure 24: Create Request Dialog

- **Creating Access Requests:** Once verified it will check if the user has access to the project by checking the whitelist, if not a dialog will appear allowing the user to create a request. For a higher chance of getting their request approved they can provide a reason to explain their intention, this will then be uploaded to the database and logged as a new request with all of the necessary data. A timestamp will also be made to log when the request was made. This request will then only be visible by the project owner in their access requests dialog (section 6.6.1). The user will then have to wait for a response and take it from there.

- Search Bar

- **Recyclerview Integration:** Having a search bar was vital for this section of the application as the user would struggle to find a project if there were more than 50 projects documents. The firestore query system made it more difficult than expected to implement this feature as the query results are loaded directly into the recyclerview and the only way to filter the results was to re-query the database. So the current implementation takes a search and on enter returns all the results that contain the text entered.

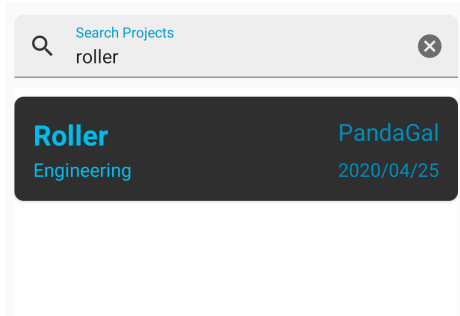


Figure 25: Search Results Example

Title: "Roller"

TitleSearch: "roller"

Figure 26: Firestore Duplicate Title Field

- **Field Formatting:** A minor issue that we experienced is that the firestore library doesn't offer a function to search for the fields of a document without case sensitivity. After some research we found the currently the best resolution for this problem is to make a second field on project creation which simply holds the lowercase version of title, and then convert the search text to lowercase to match it. With that fix the search bar works as intended and small features such as the "clear text" button make the search field more user friendly.

6.8 Selected Project Page

Once a project has been selected we do two checks, if the user is the owner a manage mode button will appear next to the title allowing the user to change and develop the project. Another check will occur to see if the project has an NDA form, if true it will show the download project NDA button allowing users to download the project NDA form at any time.

- Breakdown and Display

- **Information:** This page consists of all the detailed information a project holds, this consists of its title, owner, creation date, category, description, and development sections. On creation the project development section will be empty and exist to make the project owner think when looking to develop their idea, this is something we planned in our business development methodology (section 4.2).
- **Styling:** This activity is styled slightly differently to the rest of the app whilst maintaining consistency. The dark title background and scroll view are used to display the projects information both effectively and aesthetically. This is because project viewers may be investors or potential partners that are looking to be impressed.

- Management and Development

- **Manage Project Mode:** If the user owns the project, they can click the edit project button mentioned in the section intro to make changes and fill out the project development structure. When the button is pressed it changes to a tick icon and all of the text fields become editable.

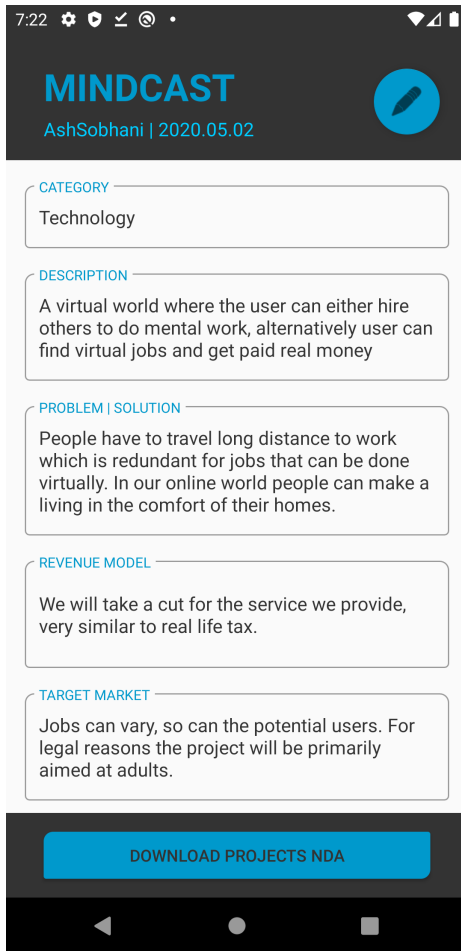


Figure 27: View Project Activity

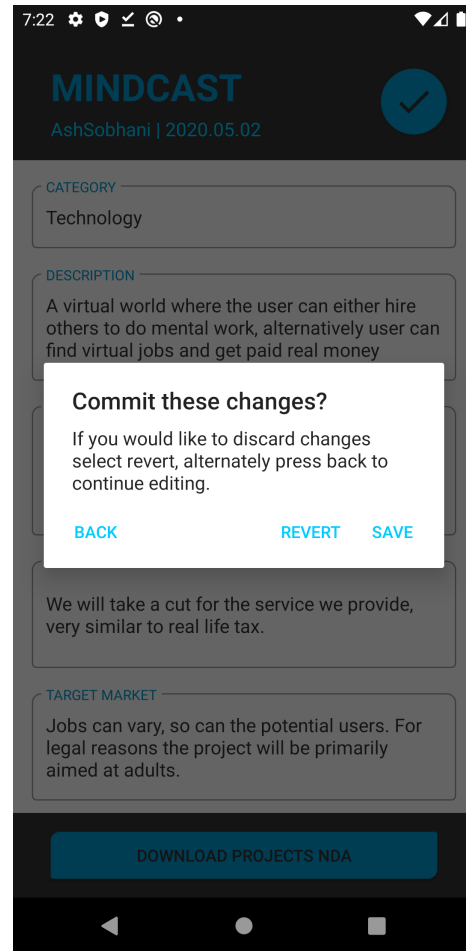


Figure 28: Changes Confirmation Dialog

- **Development:** Whilst in the management the user can take time to fill out the provided development boxes, the sections are an adaption of the business model canvas [5]. They are intended to make the user think, develop and effectively document their progress. The sections are as follows:
 1. **Problem/Solution:** What they want to solve and how
 2. **Revenue Model:** How they plan to make money and be sustainable
 3. **Target Market:** Who their product is aimed at
 4. **Current Progress:** How far through the project is so far
- **Confirming Changes:** Once a set of changes have been made the user can click the tick icon to exit management mode, doing so will trigger a dialog that asks the user if they would like to save the changes, revert back or simply to continue editing. This feature is important as they may make changes the regret and should have a way to restore the projects previous state.

6.9 Access Request States

In order for users to request and gain access to projects they have discovered, it's important that a secure system is put in place. A user that wants to gain access to a project must first have their request accepted by the project owner and secondly be prompted to sign the NDA form if the project has one. This will ensure that the owner has full control over his whitelist and everyone in that list will be legally liable for any infringements stated in the non-disclosure agreement.

6.9.1 Possible States

In order to implement this process we have implemented various request states that the user can switch between. The different states are as follows:

- A. **Access Requested:** A user has requested access from a project
- B. **Signature Pending:** The request has been accepted with the condition that the NDA is signed off
- C. **Request Accepted:** The request is successful, the user has been added to the project whitelist
- D. **Request Declined:** The project owner has declined the users request
- E. **Access Revoked:** The user has revoked their access privilege after gaining access
- F. **Signature Denied:** The user has disagreed to the NDA form and revoked their request
- G. **Project Archived:** The project has been archived, no one can access it and all requests are nullified

6.9.2 Requests State Diagram

To represent this system and show the relationship between different we created this state diagram. Note this does not include the “project archived” state, as it simply nullifies the request regardless of its state.

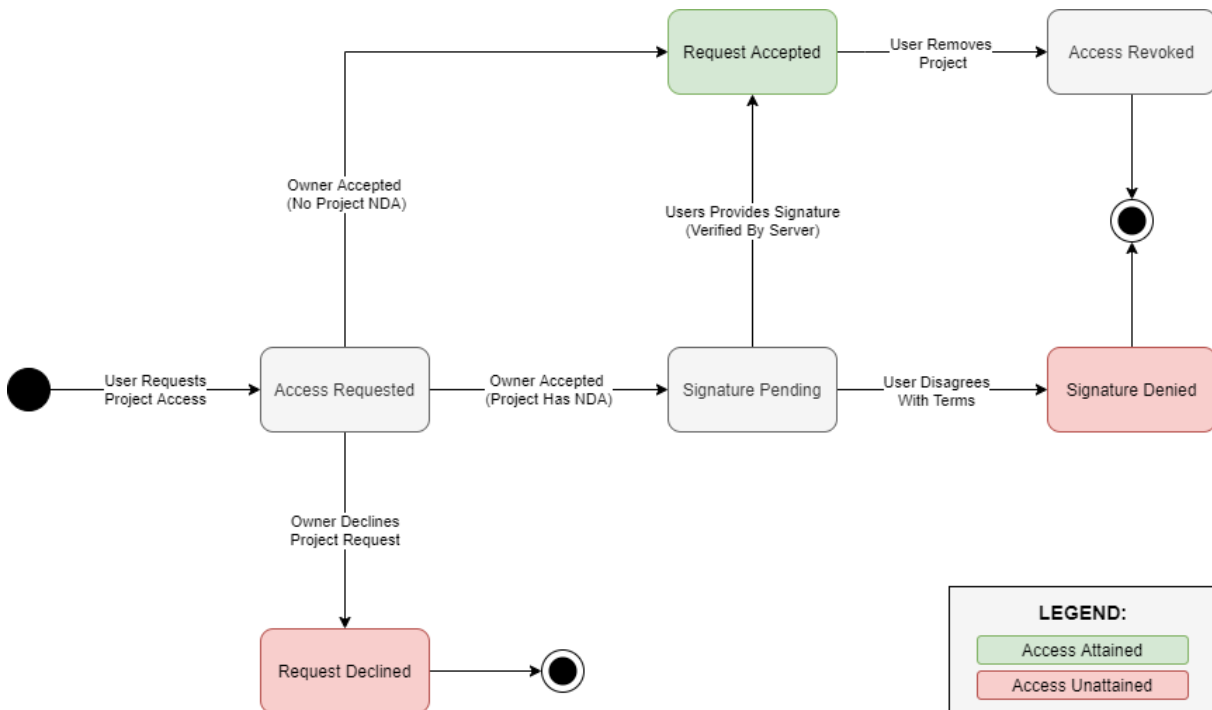


Figure 29: Access Requests - State Diagram

When implementing the states we realised that it cannot happen locally on the phone, this is because a skilled hacker could exploit the states and skip to gaining access. For this reason we ensured that reaching key states required action that was external to the user, such as the project owner accepting the initial request and by having the signature verification request being sent to the server to be validated. Based on the result the user will either be granted access by being added to the whitelist or will receive an error message notifying them that the verification failed. This can be seen in the provided state diagram.

6.10 Digital Signature and Encryption

Once the application infrastructure was complete, we started to implement the digital signature using asynchronous encryption using the Android frameworks discussed in the methodology (section 4.1.2). In order to do this we had to generate and store the keys, retrieve and hash the NDA document, allow the user to perform the signature, and finally verify and store the signature. In this section we will break down each step and explain how it was done.

6.10.1 Generating The Keys

First we created an instance of the key pair generator (section 4.1.2), setting the algorithm property as “RSA” and the provider as android key store. We then initialise the generator with the following properties:

- **Alias:** The keys are locked away with an identifier that we set to be the users UID.
- **Digest:** We then set the digest to be created using the “SHA512” algorithm.
- **Key Size:** We set the key size to a secure 2048 bits.
- **Padding:** This was set to PKCS1 ensuring the data is an exact multiple of the block size.

Once the key pair generator has initialised, we use it to generate the key pair, the keys are automatically stored in the device due to the generator’s provider being set to key store (section 4.1.2). To retrieve the public key we call the “getPublic()” function, this returns a public key object that is just a reference to key store. To get the actual value we can then apply “getEncoded()” method which returns the actual public key bytes. Note this cannot be done with the private key as key store is designed to keep it internal to the phone and cannot be retrieved or manipulated by anyone.

Finally we convert the public key bytes to a string so we can upload it to the firestore database in the “users” document along with the other personal information which is uploaded upon account creation.

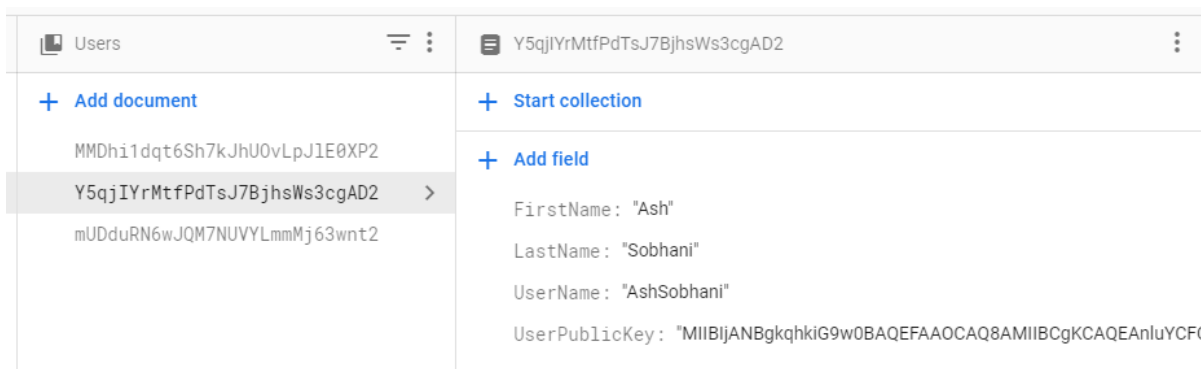


Figure 30: Firestore Database - User Public Key

6.10.2 Hashing The NDA File

Before creating the digital signature the NDA file needed to be retrieved, converted into bytes and hashed to create a message digest. We also need to notify the activity that the NDA files has been hashed and is ready for signing.

- **Retrieving The File:** Once the user downloads the NDA file within the signature activity (section 6.6.2) we retrieve the file name and use it to create a directory string, this is then parsed into a URI and converted into a file path.
- **Converting To Bytes:** The function “Files.readAllBytes()” is then given the file path as a parameter and returns, if successful, the file in bytes.
- **File Ready Check:** Once done we enable the agree button, which performs the signature, this is because if the button is clickable before the file is converted into bytes the signature will fail. This system also ensures that the user downloads the NDA correctly before being able to continue to the next step.
- **Creating The Digest:** If the user agrees to the terms and clicks the accept button, we get an instance of the message digest and set the algorithm to SHA-512. To create the digest we simply use our digest and update it using the file bytes we made in the previous step. This returns the hashed NDA file which is our message digest and is ready to be used.

6.10.3 Creating The Digital Signature

To create the digital signature we first had to retrieve the user private key from their local key store, and then use that and the file digest to perform a signature. This is done when the user presses the agree to terms button just after the digest is made.

- **Retrieving The Private Key:** This consisted of making a key store instance and getting the relevant entry using the alias that we had previously set to the users UID, we then retrieved the private key and stored it in a variable.
- **Creating and Performing Signature:** In order to make a signature we first had to get a signature instance of type “SHA512withRSA”. We then set the signature key and message to the private key and NDA file digest we had prepared previously. Following which, the “sign()” function is used to perform the signature; this returns a signature in bytes. If successful we return the signature to be verified.

6.10.4 Signature Verification

Once the signature is made, we must both verify that it matches the users public key but also store it in the database for any future cases that require signature evidence. In which case we would retrieve it and prove that the verification passes given the right parameters. A verification will pass if the public key decrypts the signature and returns a hash identical to that of the hashed NDA file, if this is true we can confirm that the private key used to sign the document and the users public key match proving that the user signed it.

- **Runtime Verification**
 - **Explanation:** When the user makes a signature it’s important that the server validates it and returns a result, this is so a malicious user is unable to bypass or exploit the code to gain access with a broken or missing signature. In order to do this we used the firestore cloud function feature we discussed in the database research (section 5.2.1). Once validated we upload the signature along with a timestamp to the access request document, showing when the user agreed to the terms of the NDA.
 - **Server Verification Function:** This required learning some “node.js” in order to program a verify signature function external to the app and then upload it to firestore [13]. A request can then be made that would run the function using the firestore API, containing the hashed file, public key, and signature. It was programmed to only return true if all parameters were provided

and the verification passed, so we applied an on complete listener to the request and based on the result we either notify the user that it'd failed or grant them access by adding them to the project whitelist. The firestore UI provides function logs where can see if they have been triggered along with the result output.



Figure 31: Cloud Verification Function - Logs

- Future Validation
 - **Explanation:** In the rare case that a user tries to repudiate and claim that they did not provide the digital signature we must have system in place to prove that they did. To achieve this an external verification system that takes the parameters and reliably returns the result would be ideal.
 - **Verification App:** We made a very simple but effective verification app. It contains two text fields that take a public key and signature (in string format) respectively. It also requires the user to select a file from the device storage and displays the chosen file name in a text field. Once the required data is provided we can select the verify button, this converts the file into bytes and hashes it into a message digest, it then creates a new signature object taking the public key, hashed file (digest), and signature as parameters. Finally the “verify()” function (from the android encryption library) is called which will return whether or not the verification has passed using an android toast message. An example can be seen in the provided figure.

7 Project Evaluation

After the implementation it was important to test the project from various angles to ensure that the app was working correctly, the digital signature was reliable and that the requirements set had been met. In this section we will show and explain our approach and show the results we attained, using this information we can accurately evaluate the project and explore the limitations we experienced in throughout the project.

7.1 Application Testing

To make sure the application runs bug free and that the different features work as intended we did an elaborate series of tests that cover the majority of possible user actions. To do this we labelled out all of the different actions, wrote what we expected would happen and then tested it and documented the result (table 4). Issues that were found were then documented and fixed (table 9).

7.2 Digital Signature Testing

In this section we test aspects of the digital signature system we implemented and the encryption that was done, this will consist of key generation, file hashing, signing, and verifying (table 5). This is to ensure that giving various inputs will return the expected results. This is important as having bugs occur that comprises

the security at a later stage, it could cost users their valuable IP if their ideas were stolen and we were not able to protect them and fight their case due to a system fault. This was mostly done using the verifier app we made (section 6.10.4). We were happy to find that the tests all passed reassuring us that the system is secure and the measures have been implemented correctly.

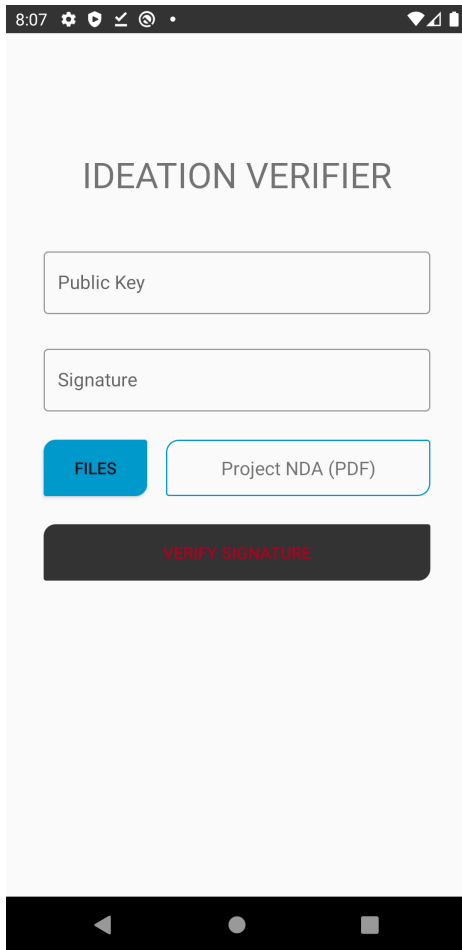


Figure 32: Ideation Verifier Application

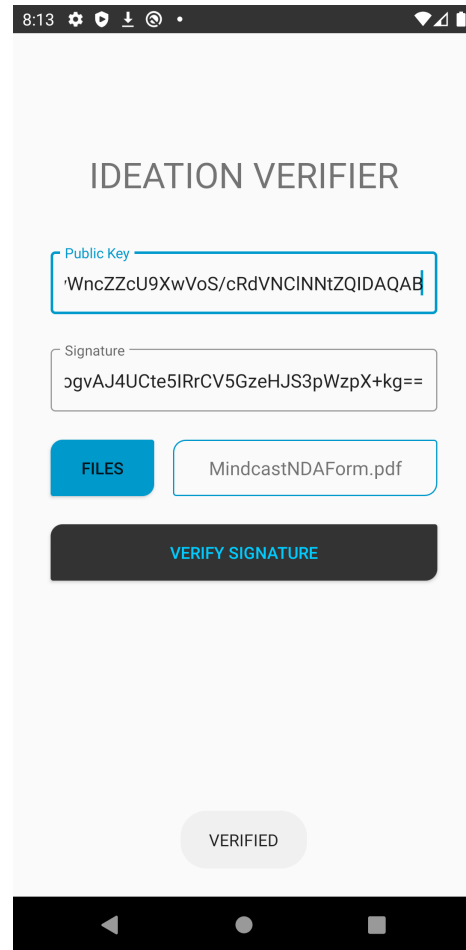


Figure 33: Verification Passed Example

7.3 Acceptance Testing

Once we had covered the more intricate parts of the project it was important to step back and ensure that we had created a product that met the requirements we had defined in our plan and that it has all the deliverables that we expected to provide. In order to do this we composed a test for all three requirement sections, this consisted of the following; functional requirements testing (table 6), non-functional requirements testing (table 7), and finally software requirements testing (table 8).

The tested we did mostly passed reassuring us that we stuck to the original requirements well. The couple that failed were discussed and reasoned in the limitations section (section 7.5).

7.4 A Legal Perspective

To find out if the project has been conducted well enough to satisfy legal requirements we spoke to Ariel Makambo (LLB (HONS)), currently working as a legal secretary, to get an external perspective and a professional opinion on the project and its implementation. He then provided the following summary with references to the EU electronic identification regulations [22].

“Pursuant to the scope of the proposed application, it is paramount that the application ensures that users can have confidence that Electronic Signatures (ESs) are legally binding and that Courts are able to identify and distinguish signatories.

It can be shown by Regulation (EU) No 910/2014 that the implementation of ESs could be legally binding. Should the proposed application satisfy the definitions set out in Article 3 regarding part 10 – 14 and 22 – 23, which it can be argued that the proposed application does, then the Asymmetric Encryption (AE) implemented should allow electronic NDAs signed with ESs to have the equivalent legal effect and admissibility as handwritten signatures in accordance with Section 4, Article 25 and/or Article 26.

Article 26 can be interpreted as a higher standard than Article 25. It seems that the proposed application satisfies this standard even in its infancy as the AE implemented creates ESs that are uniquely linked to and are capable of identifying signatories. Also, the signatory is able to, with a high level of confidence, use their private key under their sole control with potentially unauthorised changes in the data capable of being detected.”

7.5 Limitations

Whilst doing the project various factors meant that some of the goals were not reached and some work still needs to be done before this app is ready for launch. This has been acknowledged and in this section we are going to highlight some notable limitations, why they were not achievable and any plans to overcome them in the future.

7.5.1 Real ID Verification

To add an extra level of security passports or licenses could be used along with a video clip to verify a user’s real identity, this would make it even harder for malicious users to repudiate after signing an NDA form. We tried to implement this but found that a 3rd party would be required to enable this feature, as they have an extensive image detecting AI algorithm which can scan the provided media (image and video), cross reference them and verify that they match.

Due to both costs and time restraints we decided that this would not be appropriate in the scope of this project. However, it was still worth considering as it would have given an extra layer of ID verification, that would be required, if the app were to be deployed.

7.5.2 Multiple Device Key Support

The application can be loaded and logged into from any device, but digital signatures can currently only be signed on the device that made the account. This is because the system we have developed stores the users private key local to their device. We’ve done this to ensure that only they have the tools required to create a signature, using another device will return an error as the private key will not be found. This was not in the scope of the project or its requirements but would have to be done in the second round of development before the app is deployed.

In order to overcome this we'd need to implement a multiple device system which recognises new devices and create a key pair for each one. Once a device is removed from the account the public key will remain to verify any older signatures signed by the user, with a reference to private keys alias that was set.

7.5.3 UX Testing

After working quite extensively on the applications design and workflow we had planned to run some user experience tests to receive some real feedback from friends and family, unfortunately this was not possible due to the coronavirus pandemic and the social distancing measures that were put in place. This meant that we couldn't really test the HCI aspects and concepts that were implemented, instead we performed so in-house tests such as application testing to make sure any unseen bugs would be found and fixed.

8 Summary and Reflections

This section starts with a summary of how the project was conducted by discussing the process and the tools used to structure the progress made. We then go onto discuss some of our reflections, highlighting both the positive points and the areas that have room for refinement and improvement.

8.1 Project Management

In order to manage the project 3 main tools were used, these consisted of a: Gantt Chart, Trello (Kanban Board), Dairy (Personal Log). All of these contributed to a system that helped up organise the aspects of the project effectively.

- Kanban Implementation
 - As mentioned in the project proposal we made use of a tool called Trello which is an online Kanban board to track both our tasks and their deadlines. We set the board up to have 4 sections: Backlog, Doing, QA and completed. The “backlog” was where we put all of the tasks that need to be done, these were generated by looking at the Gantt Chart initially, other topics were then added as we started looking into the main areas. Once we picked up a task it would be moved it to the “doing” section, this is so we knew day by day, what we were working on. Upon finishing a task we'd move it into QA which is where we could analyse and check the quality of the work (done by checking references and by ensuring we have covered the subsections properly). Once this was done, we'd move it to the “completed” section showing that the task is complete. Above the title of the cards there are colours, these are labels that show various things such as priority (red and orange), type of task (research is purple) etc.

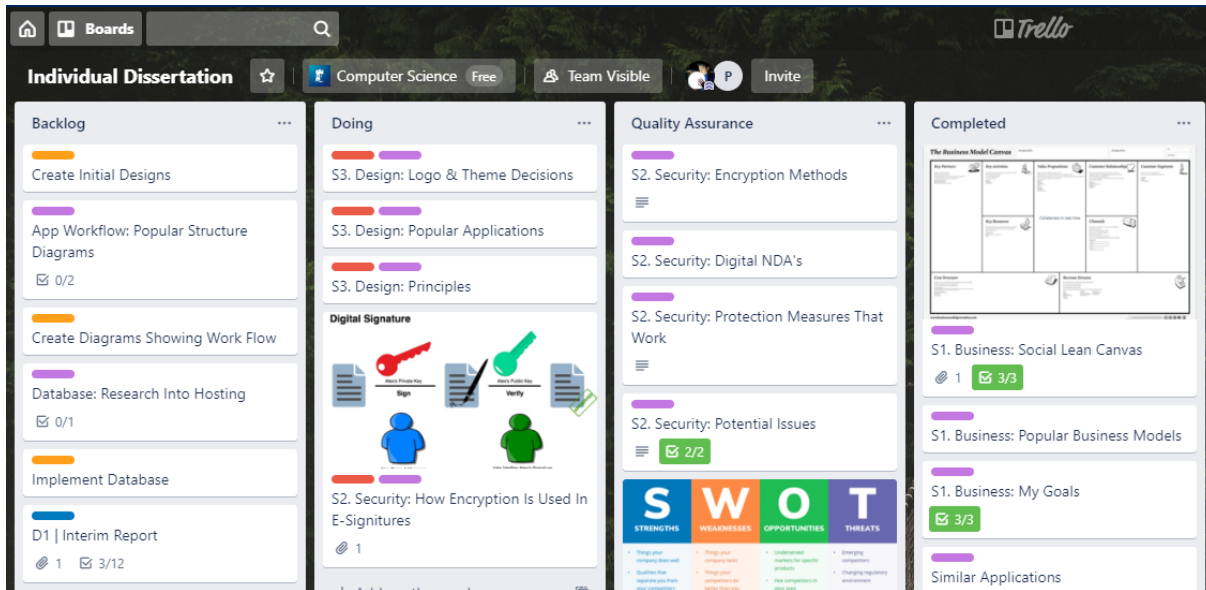


Figure 34: Project Trello Board - At Sprint 3

- Micro and Macro Management

- The advantages of doing this is that we always had a visual representation of the completed and outstanding tasks which was used to help us keep track of progress. Another advantage is that Trello cards allow you to add subsections, links and notes. This meant that each card would be a neat package consisting of everything we needed to get that task done (micro tasks). It also enabled us to quickly add more tasks as they emerged, this is something that a Gantt chart was not very good for so we used it more as an overview to represent main sections of the project (macro tasks).
- Incorporating sprints into this system was quite easy as we could use the Gantt chart to see what macro task needs to be done in the upcoming week respectively to the current plans and then we could label the micro task cards on the backlog relevant to that sprint as the priority cards. In terms of keeping a log, we used a diary to note what tasks from the Trello board we wanted to complete the next day. Doing this meant that once the tasks we planned on doing were done we could spend the rest of the day focusing on other work that needed doing.

- Time Tracking

- We used an application known as Toggl to create time entries and track the amount of time spent of various areas of the project. We decided to use this as it would become useful when attempting to gauge how long things may take, for example after having spent 5 hours on implementing the discovery activity we could estimate that the other activities would take a similar amount of time.
- This helped a great deal both when we were doing the projects but also when looking back in hindsight to analyse how we distributed our time, this information could be used to consider the areas that should've been given more or less time in the project for future reference. An example breakdown of April is provided in the appendix (section 10.3) to give an idea of what this software provides.

- GitHub Integration

- **Version Control:** We used a developer tool called GitHub for various reasons, a major one being “a version control system (VCS) allows you to track the iterative changes you make to your code” [3]. This is really important as being able to revert back to older versions of code can

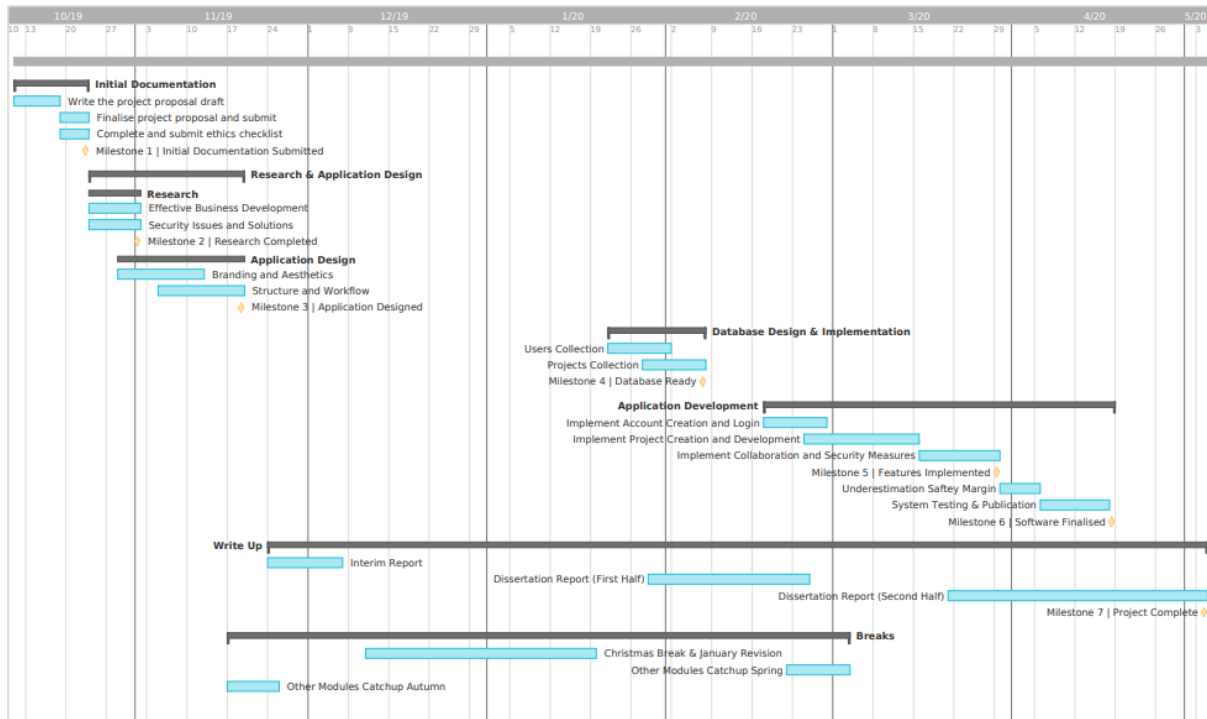


Figure 35: Project Gantt Chart - Final State

sometimes save lots of time, this is because generally code is very interlinked and so a mistake in one place can cause a chain of issues. Using version control we can pull a previous state from the repository.

- **Other Advantages:** Pushing incremental changes to git also means that the project is saved securely on another platform meaning if the project is lost it can easily be restored, For this reason we aligned pushes with the kanban cards (section 8.1) meaning that every feature would be documented and saved. Over the course of the project we pushed almost 90 times, this was later used to help document the project implementation as it acted as a development skeleton (figure 37).

8.2 Reflections

- Successful Contributions and Positive Remarks

- **Project Overview:** We have established and proven that there is real potential in this application and the concept of a start-up hub reinforced by effective security measures. The firestore infrastructure was integrated effectively, our code base is of a high standard, the security concepts were implemented carefully, and testing was conducted to ensure that any issues would be found and resolved. Any limitations have been outlined and a plan of action has been considered where possible for improvements. Everything considered we are happy to have taken what initially was meant to be a proof of concept and instead delivered an application which is not far from being deployable.
- **Project Management:** we have come to an end having completed all of the things that we set out to do and more. The project timeline was not followed religiously but was in place as a guideline, allowing us to constantly re-evaluate and tend to things that were of priority in order to hit milestones on time. The time tracking application we used was a very good way documenting

the workload and seeing how many hours were being spent in different areas, it also meant that we could gauge how long things would take based on previous times.

- **Critical Review and Potential Improvements**
 - **Application Practices:** The application is well built but is far from perfect, there are so many more android libraries and security measures that can be taken to optimise the product. We tried to abide by good development practices but were overwhelmed by how extensive the android development community is. Libraries such as android jet pack can be used to optimise both the applications performance and the way that the code is structured. In future projects we would try to implement such libraries to keep up with the android development community.
 - **Database Infrastructure:** Firestore was great in the scope of this project but does have its limitations and drawback, for example flexibility and scalability. Using a database as a server does provide all the necessary tools but offers little freedom when looking to customise the servers properties. The noSQL structure database that we used was also very powerful but meant that in cases data needed to be exist as a duplicate to make querying possible, this stunts the scalability as the extra storage can add up to cost a lot if the user grows large. In such a case we would have to consider switching to a different database infrastructure to maintain a sustainable platform.

8.3 Conclusion

We think that the project was conducted well throughout, consequently the deliverables are of a high standard and offer effective solutions to the issues posed at the start of this paper. If given a chance to improve this project further we'd invest more time researching the latest android components to create a more to-date application. In addition to that we would look into programming the software in Swift, allowing us to deploy the application into the Apple Store for IOS devices, meaning that the app would be available to a much larger user base.

9 Bibliography

References

- [1] Margo A Bagley. Internet business model patents: Obvious by analogy. *Mich. Telecomm. & Tech. L. Rev.*, 7:253, 2000.
- [2] Artiom Baloian. How to generate public and private keys for the blockchain, 2018. <https://medium.com/@baloian/how-to-generate-public-and-private-keys-for-the-blockchain-db6d057432fb>, Accessed (2020-03-29).
- [3] John D Blischak, Emily R Davenport, and Greg Wilson. A quick introduction to version control with git and github. *PLoS computational biology*, 12(1), 2016.
- [4] John M Carroll, Robert L Mack, and Wendy A Kellogg. Interface metaphors and user interface design. In *Handbook of human-computer interaction*, pages 67–85. Elsevier, 1988.
- [5] DH Coes. Critically assessing the strengths and limitations of the business model canvas. Master’s thesis, University of Twente, 2014.
- [6] Tim Cooijmans, Joeri de Ruiter, and Erik Poll. Analysis of secure key storage solutions on android. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 11–20, 2014.
- [7] Ian Curry. An introduction to cryptography and digital signatures. *Entrust Securing Digital Identities and Information*, 2001.
- [8] Android Developers. Access data offline, 2020. <https://firebase.google.com/docs/firestore/manage-data/enable-offline>, Accessed (2020-03-21).
- [9] Android Developers. Android keystore system, 2020. <https://developer.android.com/training/articles/keystore#SecurityFeatures>, Accessed (2020-03-18).
- [10] Android Developers. Cloud storage, 2020. <https://firebase.google.com/docs/storage>, Accessed (2020-03-19).
- [11] Android Developers. Cryptography, 2020. <https://developer.android.com/guide/topics/security/cryptography#deprecated-functionality>, Accessed (2020-03-15).
- [12] Android Developers. Firebase authentication, 2020. https://firebase.google.com/docs/auth#key_capabilities, Accessed (2020-03-19).
- [13] Android Developers. Get started: write and deploy your first functions, 2020. <https://firebase.google.com/docs/functions/get-started>, Accessed (2020-03-27).
- [14] Android Developers. Key pair generator, 2020. <https://developer.android.com/reference/java/security/KeyPairGenerator>, Accessed (2020-03-17).
- [15] Cloud Firestore. Android keystore system, 2020. <https://firebase.google.com/docs/firestore>, Accessed (2020-03-19).
- [16] Emet Gürel and Merba Tat. Swot analysis: A theoretical review. *Journal of International Social Research*, 10(51), 2017.
- [17] Adam Hayes. How nda’s work and why they’re important, 2019. <https://www.investopedia.com/articles/investing/041315/how-ndas-work-and-why-theyre-important.asp#>, Accessed (2020-03-23).

- [18] Ohad Inbar, Noam Tractinsky, and Joachim Meyer. Minimalism in information visualization: attitudes towards maximizing the data-ink ratio. In *ECCE*, volume 7, pages 185–188, 2007.
- [19] Ron Leshem. Why execution (not ideas) will bring you success, 2012. <https://www.inc.com/ron-leshem/why-execution-not-ideas-will-bring-you-success.html>, Accessed (2020-03-22).
- [20] Google Material. Launch screen, 2020. <https://material.io/design/communication/launch-screen.html#usage>, Accessed (2020-03-23).
- [21] Aaron McKenny, Jeremy Collin Short, and Thomas Houston Allison. Errant signals: How crowdfunding performance elicits competition for de novo entrepreneurs. In *Academy of Management Proceedings*, volume 2018, page 18694. Academy of Management Briarcliff Manor, NY 10510, 2018.
- [22] Official Journal of the European Union. Regulation (eu) no 910/2014 of the european parliament and of the council, 2014. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG, Accessed (2020-05-02).
- [23] Kickstarter PBC. What is kickstarter?, 2019. <https://help.kickstarter.com/hc/en-us/articles/115004996453-What-is-Kickstarter->, Accessed (2019-11-20).
- [24] Jörg Schepers, Ralf Schnell, and Pat Vroom. From idea to business—how siemens bridges the innovation gap. *Research-Technology Management*, 42(3):26–31, 1999.
- [25] José Van Dijck. ‘you have one identity’: Performing the self on facebook and linkedin. *Media, culture & society*, 35(2):199–215, 2013.
- [26] Anthony E Zank and David R Stevens. Electronic signature management system, October 23 2001. US Patent 6,307,955.
- [27] Martin Zwilling. 7 ways to keep your focus on execution after the idea, 2016. <https://www.entrepreneur.com/article/271086>, Accessed (2020-03-23).

10 Appendix

10.1 Ideation User Manual

10.1.1 Application Set Up

In order to get the application up and running please follow these steps:

1. Download the Ideation Project.
2. Download and open Android Studio.
3. Select file, open project, locate the Ideation project folder and load it in.
4. Next we need to get an android emulator running, to do this open AVD manager.
(found at the right side of the tool bar as a phone icon)
5. Create a new virtual device
6. Then select a device (recommended: Pixel 3a) and press next.
7. Select a system image (recommended: Q), select next and finish.
8. Click the green play button located on the toolbar to load and run the application.

10.1.2 Getting Started

Once the application is open you will want to create an account and login to, do this:

1. Select “signup” to navigate to registration.
2. Fill in all the fields, and use a secure password.
3. Once complete select “create account” and to complete registration.
4. After being navigated back to the login page, enter your credentials and select “login”.
5. Welcome to Ideation!

10.1.3 Project Guide

First lets **create a project**: navigate over to the projects section and click the add project button. Then give it a title description and category before creating it (optional: upload a NDA file).

Next lets **develop the project**: Select your new project to open it, now click the edit button to customise it and fill out the development sections. When ready click the complete button and decide if you want to save the changes.

Finally lets **request access**: To request access, navigate to the discovery section and select a project that you find interesting (note: this feature requires email verification). A dialog should pop up allowing you to make a request and give a reason, if you have one. Once it's processed you just have to wait patiently for a response from the owner.

10.2 Evaluation Tables

10.2.1 Implementation Testing

Table 4: Application Testing

No.	Test	Expected Outcome	Result
1.	Pressing application icon	Splash screen should appear and then navigate to login screen	Passed
2.	Pressing application icon (previously logged in)	Splash screen should appear and then navigate to discovery fragment	Passed
3.	Clicking “sign up” button	Navigate user to the registration activity	Passed
4.	Creating account (Incorrect or missing fields)	Display the relevant error message	Passed
5.	Creating account (Fields good)	The account should be made, and verification email should be sent to users email and user navigates to sign in activity	Passed
6.	Signing In (Incorrect Credentials)	Log in fails and error message appears notifying the user	Passed
7.	Signing In (Good Credentials)	Authorisation passes, the user is navigated to the discovery fragment	Passed
8.	Project Clicked (Not Verified)	Snack bar appears informing the user verification is required	Failed (1)
9.	Project Clicked (No Access)	Request dialog pops up allowing the user to give a reason and send access request	Passed
10.	Send Request (Already Sent)	An error message appears promoting informing the user a request is already pending	Passed
11.	Project Clicked (Has Access)	User navigated to view the project information	Passed
12.	Searching For Project (Test)	The projects are filtered only displaying the projects that matched the search results	Failed (2)
13.	Clear Search Filter (Clear Icon Clicked)	Delete the search text and display all projects	Passed
14.	Select discovery on navigation bar	Discovery section appears	Passed
15.	Select projects on navigation bar	Project tabs appear (My Projects and Shared Projects)	Passed
16.	Select profile on navigation bar	Profile section appears and information is loaded	Passed

17.	Select My Projects Tab	Slider navigates you to my projects	Passed
18.	Select Shared Projects Tab	Slider navigates you to shared projects	Passed
19.	Click create button on my projects	Opens new project activity	Passed
20.	Select category from drop down	7 different category's appear allowing you to select them	Passed
21.	Click Select File Button	File selector appears allowing you to select a PDF	Passed
22.	Save Project (Missing Fields)	An error message appears promoting informing the user fields are missing	Passed
23.	Save Project (No NDA Form)	Projects is made, NDA security layer not added	Passed
24.	Save Project (With NDA Form)	Project is made with NDA, uploaded to cloud with a reference in database	Passed
25.	Select Access Request Button (My Projects)	Access request dialog should appear showing users project access request, options to accept or decline	Passed
26.	Accept Request (No NDA)	The user is granted access and request status updated to request accepted	Passed
27.	Accept Request (With NDA)	The user request status updated to signature pending	Passed
28.	Decline Request	The user request status updated to access denied, request reaches final state. (User can apply again)	Passed
29.	View project (Owner)	The project information loads and an edit button is visible allowing them to edit the project	Failed (3)
30.	View project (Viewer)	Project information loads but cannot be edited	Passed
31.	View project (With NDA)	Download NDA button appears allowing user to view NDA	Passed
32.	Click Signature Button (Shared Projects)	Opens signature dialog showing pending signatures	Passed
33.	Click On Project Pending Signature	Opens the signature activity informing user about project NDA protection	Passed
34.	Decline Agreement Signature	Set the request state to signature denied, final state	Passed
35.	Accept (Wrong/Missing Terms Password)	Fail and display error, asking for correct credentials	Passed

36.	Accept Terms (NDA Not Downloaded)	The button should not be clickable as they have not read the conditions	Failed (4)
37.	Agree Terms (Authenticated and NDA Downloaded)	Signature is made and stored, user is added to project whitelist	Passed
38.	Agree Terms (Signature verification fails)	Display message informing of verification failure	Passed
39.	Logout Button Clicked	User logged out and returned to login page (unable to return to discovery on back pressed)	Failed (5)

Table 5: Digital Signature Testing

No.	Test	Expected Outcome	Result
1.	Get a public key from key pair generator	Should return a correctly formatted public key of length 2048 bits	Passed
2.	Converting a PDF file into bytes and back	We made a function for this and it returns an identical PDF file to what it was given	Passed
3.	Retrieving the private key from key store	A private key is retrieved that can be used to create signature (unable to see its contents as expected)	Passed
4.	Producing a signature using the private key and hashed file	A signature in byte form is returned that we can convert and store	Passed
5.	Runtime verification server side using the user's public key	A signature should be verified given it has the correct NDA file and public key	Passed
6.	Future verification using the verifier app (correct combination of key, signature and file)	The server processes the request and returns true	Passed
7.	Giving the verifier app a mismatching public key	The server processes the request and returns false	Passed
8.	Giving the verifier app an incorrect file when verifying	The server processes the request and returns false	Failed
9.	Giving the verifier a mismatching signature	The server processes the request and returns false	Passed
10.	Encoding the public key to string (base64) to store in database	We get a public key string that we can store in the database and use in the "verifier" app	Passed
11.	Encoding signature to string (base64) to store in database	We get a signature string that we can store in the database and use in the "verifier" app	Passed

10.2.2 Acceptance Testing

Table 6: Functional Requirements Testing

No.	Requirement	Result	Reason
1a.	The user must be able to create an account (personal details and password)	Passed	The user can create an account in the sign up section of the application, by providing some information, an email, and password
1b.	The user must be able to login and access the contents of their account	Passed	The user can login by putting their email and password in the login credentials and click “signin” to access their account
1c.	The user should be able view their profile details	Passed	After entering their credentials on the login page the user can navigate to the profile section to view account details
2a.	The user must be able to create a new project	Passed	In the “My Projects” section the user can create the “+” button to navigate to the new project page
2b.	The user must be able to develop the project from within the app (name, description, category)	Passed	After creation the user can access the project and enter edit mode to add and edit sections
2c.	The user should be able to delete a project	Passed	In the “My Projects” section the user can swipe a project box to the right in order to delete it (after confirmation)
2d.	The user should be able to update sections of the project (using a provided framework)	Passed	In the project view 4 development sections are present, this framework will help the user consider important questions and document their idea
3a.	The user must be able to upload an NDA file to protect their project	Passed	When on the create “new project” page the user can select a PDF file from their phone and upload it with the project, it can then be viewed in before signing or in the “view project section”
3b.	The user must be protected by measures that ensure other users are verified	Passed	In order to request access to project users must have a verified email address and if required to sign an NDA will have their signature verified before being granted access
3c.	The user could be able to sign and validate forms digitally	Passed	Users can select pending signatures in the dialog found in the “Shared Projects” view and are able to sign the NDA after having read it if their authentication and signature passes

4a.	The user must be able to view other public projects	Passed	After having requested access and being added accepted the user can view other users projects and read their breakdown.
4b.	The user must be able to request access to other public projects	Passed	Once their account is verified the user can click on projects and request access accompanied by a supporting reason
4c.	The user should be able to revoke access from projects	Passed	The user can revoke access from a project in the “Shared Projects” tab this is done by swiping right on a project and confirming their decision

Table 7: Non-Functional Requirements Testing

No.	Requirement	Result	Reason
1.	The performance of the application must be reasonable (to avoid negatively impacting usability)	Passed	The application runs without bugs and has error messages to prompt uses to handle any issues
2.	The application should be responsive to any interrupt (saving the state if the user tabs out)	Passed	The activity states across the application are saved stopping any loss of work, data persistence is implemented to ensure work is uploaded even after a connection loss.
3.	The use-ability should be self-explanatory ensuring users can instinctively know how to navigate through the application without any expert guidelines or manuals	Passed	Many concepts have been followed when designing and implementing the app, to ensure users can intuitively use its features
4.	A level of reliability should be implemented (Not allowing users to accidentally make mistakes that could destroy a project)	Passed	Confirmation dialogs are implemented to ensure users have done actions intentionally. In addition we have added a revert option after a project has been edited in case information has been mistakenly deleted.
5.	The application should be secure with a reasonable level of encryption and other security measures to protect sensitive information within the app	Passed	By using the firestore back-end for authentication and the android libraries for encryption we have created a secure application with a signature verifier if proof needs to be provided

Table 8: Software Requirements Testing

No.	Requirement	Result	Reason
1a.	The application must be developed for android users	Passed	The application has been developed on android studio and is currently exclusive to the platform
1b.	The application should be free for everyone to download	Failed	We have not been able to deploy the application due to the reasons stated in the limitations (section 7.5)
1c.	The application could be available on the google play store	Failed	We have not been able to deploy the application due to the reasons stated in the limitations (section 7.5)
2a.	The application must not crash unexpectedly	Passed	After having ran the application testing and fixed any bugs we can confidently say the application does not crash unexpectedly
2b.	The application should switch between activities swiftly	Passed	By using a mixture of activities and fragment and optimising the navigation the user experience is smooth and enjoyable
2c.	The application should allow the user to navigate to their destination within 5 inputs	Passed	To test this we tried doing some complex tasks such as requesting access to a project and this was done in 4 actions
3a.	The application must save project progress if user exits	Passed	We used a android feature called shared preferences to save the data when a user not on the app, and it then reloads on return.
3b.	The application should be intuitive to use minimising mistakes and confusion	Passed	We have placed hints around the application for the more complex actions and the user of error messages and snack bars notify users if something needs changing.
3c.	The application should prompt the user to confirm deletion of a project	Passed	This is done using dialogs that ask the user to confirm their action, used in cases such as deleting projects, revoking access, and editing a project

10.2.3 Resolved Issues

Table 9: Application Resolved Issues

No.	Issues Description	Resolution	Result
1.	The user received no feedback when not being able to access request to a project due to the not having verified their email, nothing happens leaving them confused	A snack bar has been implemented that appears informing the user verification is required	Fixed
2.	The results come back as null, consequently no projects are shown	The search was caps sensitive so an extra search variation of the title is stored in the database which is lowercase just for searching. The search is then also set to lowercase. This fixed the issue and when searching the filtered results only showed projects with the word displayed in them	Fixed
3.	The project information loads but the edit button doesn't appear	The issue was that the user UID string comparison was being done using "==" operator instead of ".equals()" function, which was failing the if condition, upon fixing the project information loaded and the "edit mode" button appeared	Fixed
4.	The button was clickable causing the application to crash as the file was not downloaded meaning that the signature could not occur	We fixed this by setting the button as disabled and only enabling it once the NDA file download is complete	Fixed
5.	When the user signs out of their account and gets navigated to the login page they can press back to get back into the application unauthenticated.	To fix this we just overwrote the "onBackPressed()" function in the login activity changing its functionality to just exit out of the application instead	Fixed

10.3 Project Management

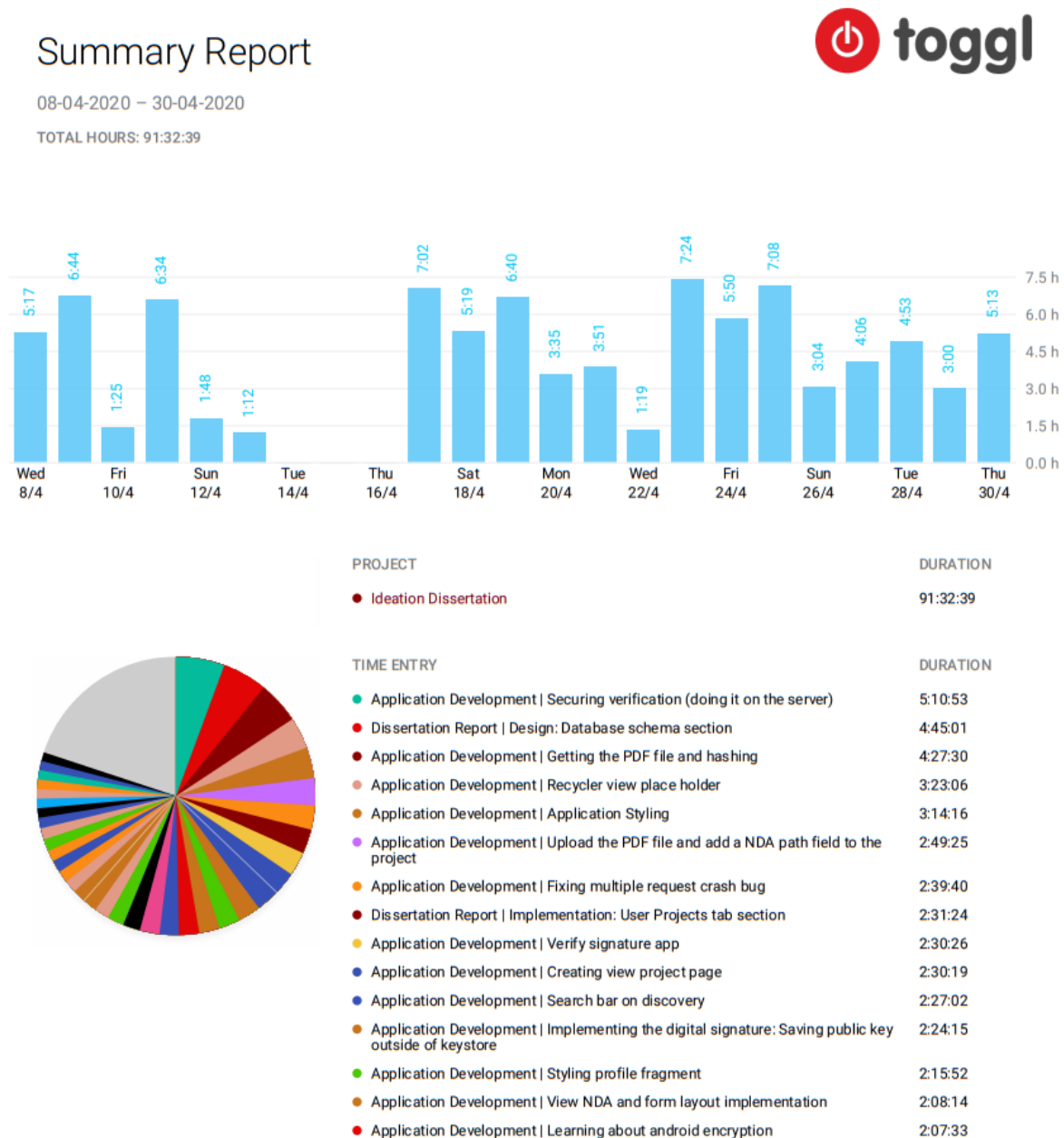


Figure 36: Project Management - Toggl Tracking (April)

84 commits

2 branches

0 packages

0 releases

0 contributors

Branch: Development

New pull request

Find file

Clone or download

This branch is 80 commits ahead of master.

Pull request

Compare

psyas8 Toasts removed from code

Latest commit ecd27ce 7 days ago

.idea

Project UID and Request UID being passed through

24 days ago

app

Toasts removed from code

7 days ago

firecast

Server Verification Function Implemented

9 days ago

gradle/wrapper

Open selected project and retrieve project info

2 months ago

.gitignore

Initial Commit - Logo Made, Fragments made, and Firebase Connected

2 months ago

build.gradle

Added upload project view components

25 days ago

gradle.properties

Initial Commit - Logo Made, Fragments made, and Firebase Connected

2 months ago

gradlew

Initial Commit - Logo Made, Fragments made, and Firebase Connected

2 months ago

gradlew.bat

Initial Commit - Logo Made, Fragments made, and Firebase Connected

2 months ago

settings.gradle

Initial Commit - Logo Made, Fragments made, and Firebase Connected

2 months ago

Figure 37: Github Repository

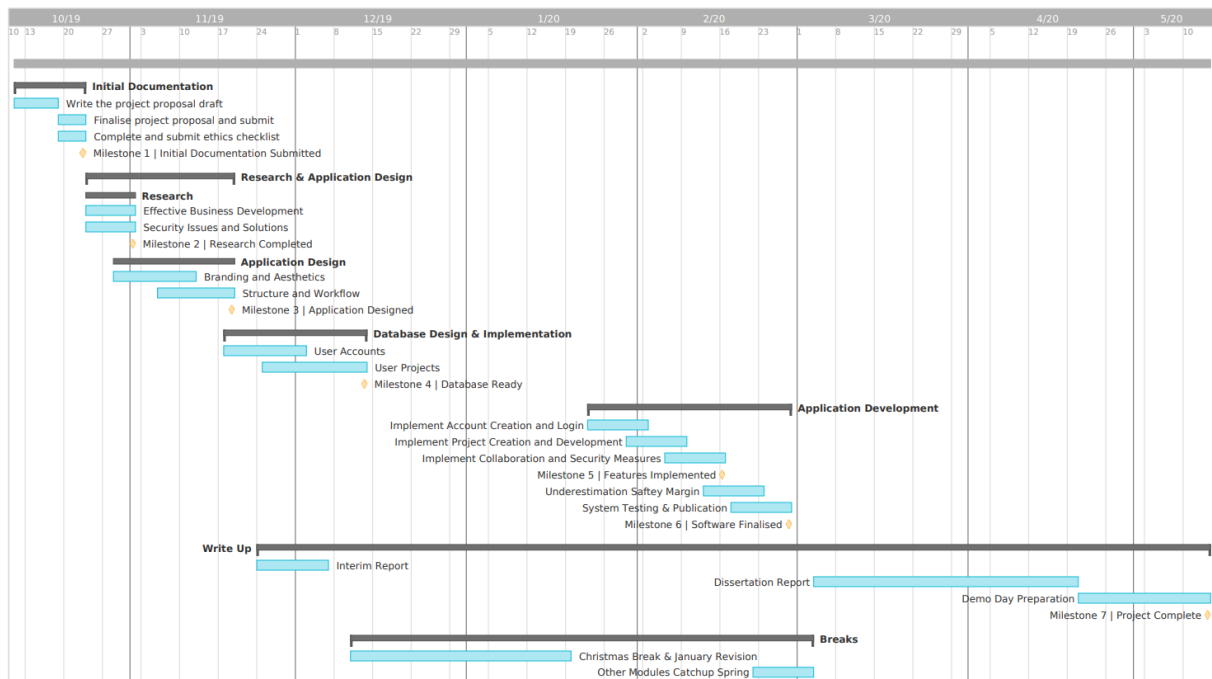


Figure 38: Project Gantt Chart - Begin State

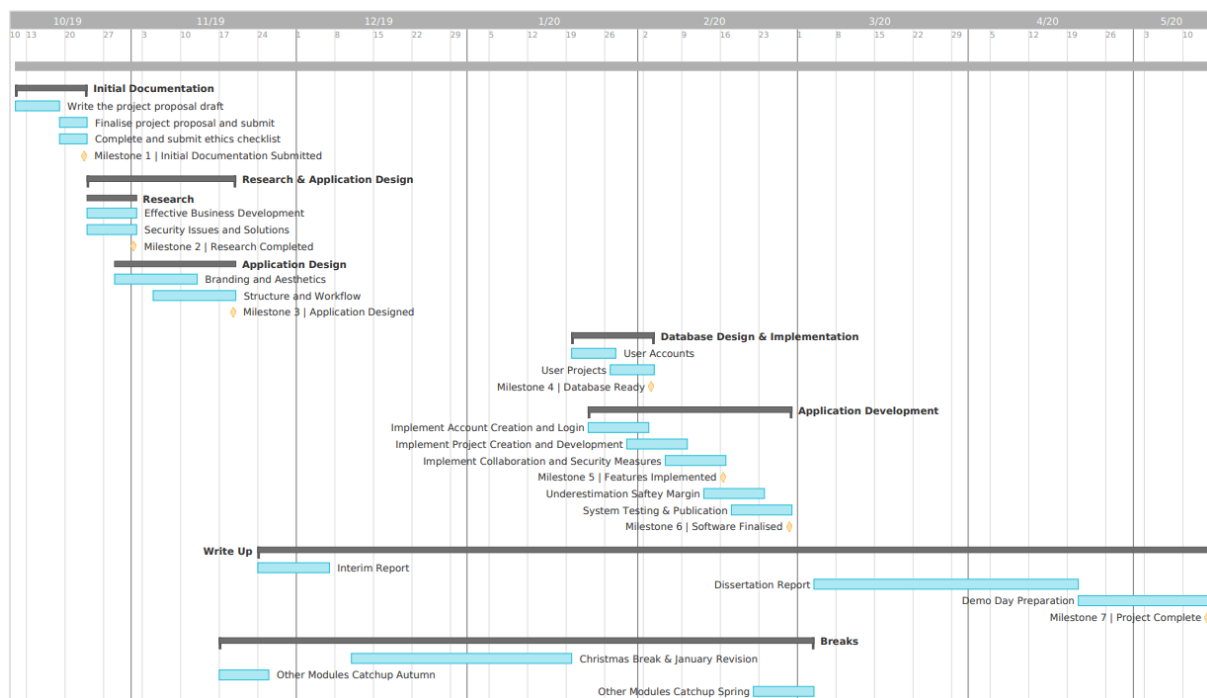


Figure 39: Project Gantt Chart - Interim State