**ESC Final Report - PayPal Identity Verification App**

**Cohort 5 Group 7**

|  |  |
| --- | --- |
| 1005398 | Rohit Raghuram Murarishetti |
| 1005337 | Acqquilaa Bathumalai |
| 1004877 | Wang Xilun |
| 1005467 | Yogesh Vivek Shelgaonkar |

## 

## 

## 

## 

## 

# 

# Table of contents

1. [The Problem Statement and Solution](#_2w87d3kxi1nt)
2. [Use Case Diagram](#_Use_Case_Diagram)
   1. [Detailed Use Case Documentation](#_Detailed_Use_Case)
3. [UML diagrams](#_v9xs8jbv98qa)
   1. [Class Diagram](#_xvy5yy5hb9ln)
   2. [Sequence Diagram](#_hu7x8wcdt2dj)
   3. [State Diagram](#_cxjy0u1aobgr)
4. [Challenges](#_tw8iiyf526b2) 
   1. [Implementation challenges](#_ejxcs858y7it)
      1. [Backend](#_7df5yyp5vqzy)
      2. [Frontend](#_6n5pp1qzfc5q)
   2. [Testing challenges](#_lczgyaywyvnz)
      1. [Testing Backend](#_7df5yyp5vqzy)
      2. [Testing Frontend](#_6n5pp1qzfc5q)
5. [Lesson Learnt](#_vzwgg04184vo)
   1. [Software development process followed](#_ovjnssuzlv1j)
   2. [Challenges faced following the process](#_1hv8hgk4pgj0)
   3. [Reflection](#_o2hewzxbovb3)
6. [Deliverables](#_61lc0u4mi1j7)
7. [Contributions](#_Contributions)

## 

## 

## 

## 

## 

## The Problem Statement and Solution:

Context of the Problem: When people migrate, their classical credentials like proof of address, credit score, and financial statements are rendered irrelevant. Many times, the proof of identity loses its legitimacy as well e.g., national id cards, and passports cease to be valid. This creates a barrier for accessing financial services like banking, remittance, credit and insurance in migrant’s new home.

Problem Statement: How may we create an identity enrolment and issuance module that migrant populations can avail of and may use it to avail of traditional financial services from traditional providers?

Our solution:

Our solution is a mobile application that solves three main issues:

1. Portable Digital Identity: Our users can submit a copy of their identification documents and get a verified digital identity through a mobile-first solution.
2. Scalable platform for regulatory usage: Admin side services on our mobile platform for regulators to easily verify users documents and identities.
3. Accessible Service for Financial Institutions: As a Digital Identity Verification platform, we can service our financial institutional clients a KYC service, and enable our users to access financial products which otherwise would be harder to access.

## Use Case Diagram

Miroboard : <https://miro.com/app/board/uXjVO9XggVc=/?share_link_id=832846791257>

# 

### Detailed Use Case Documentation:

|  |  |
| --- | --- |
| Name | Verification of user’s credentials |
| Objective | To verify the user’s credentials such as ID number, bank account number, credit card number |
| Pre-Conditions | The user must have a pre-existing account with Paypal and must have the documents required for verification |
| Post-Conditions | Success:  The user’s credentials are successfully stored in the database  Failure:  The user’s credentials cannot be stored in the database |
| Actors | Primary:  1. User  Secondary:  1. Administrator |
| Trigger | The user needs to upload their credentials to be verified so that they can use Paypal’s services |
| Nominal Flow | 1. The user logs into their existing Paypal account  2. The user chooses the relevant documents that are required for verification and uploads them to a temporary database  3. The administrator submits the documents to the relevant institution for verification  4. The documents are verified by the institutions  5. The institution informs the administrator that the documents have been verified  6. The administrator adds the documents into the database  7. The user’s credentials are verified |
| Alternative Flow | 1. The user logs into their existing Paypal account  2. The user chooses the relevant documents that are required for verification and uploads them to a temporary database  3. The administrator submits the documents to the relevant institution for verification  4. The institution finds fault with the documents submitted.  5. The institution informs the administrator about the faults.  6. The user’s credentials are not verified.  7. The administrator informs the relevant authorities to follow up, in case of illegal activity. |
| Interacts With | User, Administrator, Database, External Institutions |
| Open Issues | What if the documents submitted do not belong to the user? |

|  |  |
| --- | --- |
| Name | Creation of new account |
| Objective | To create a new account for login |
| Pre-condition | User must have downloaded the app in their mobile phone and have a stable internet connection |
| Post-condition | Success:  The user has an account with relevant details that can be used to log into their accounts in the future  Failure:  Invalid details given for the creation of account |
| Actors | User |
| Trigger | User wants to create a new account to upload credentials |
| Normal Flow | 1. The user clicks on the button to create a new account 2. The user is prompted to fill out blanks with relevant information such as name, current country of residence, email, and password 3. User gives valid email address 4. The user provides a password that has met the requirements set such as at least one uppercase, at least one lowercase, and at least one number to ensure the password is not easy to guess. 5. User re-enters the same password a second time to confirm the password 6. The user clicks submit and a new account is created |
| Alternative Flow | 1. The user clicks on the button to create a new account 2. The user is prompted to fill out blanks with relevant information such as name, current country of residence, email, and password 3. The user provides invalid email address and is then prompted to enter a valid email address 4. The password provided by the user has not met the requirements set such as at least one uppercase, at least one lowercase, and at least one number to ensure the password is not easy to guess. 5. User is prompted to enter a different password that meets the requirements 6. The user re-enters a different password a second time 7. The user is informed that the two passwords do not match and is then prompted to re-enter password 8. The user clicks submit and a new account is created |
| Interacts with | Captcha, Firewall, Two-factor authentication, User Database |
| Open Issues | What if the user creates an account with fake details? |

## UML diagrams

### Class diagram

# 

### Sequence diagram

# 

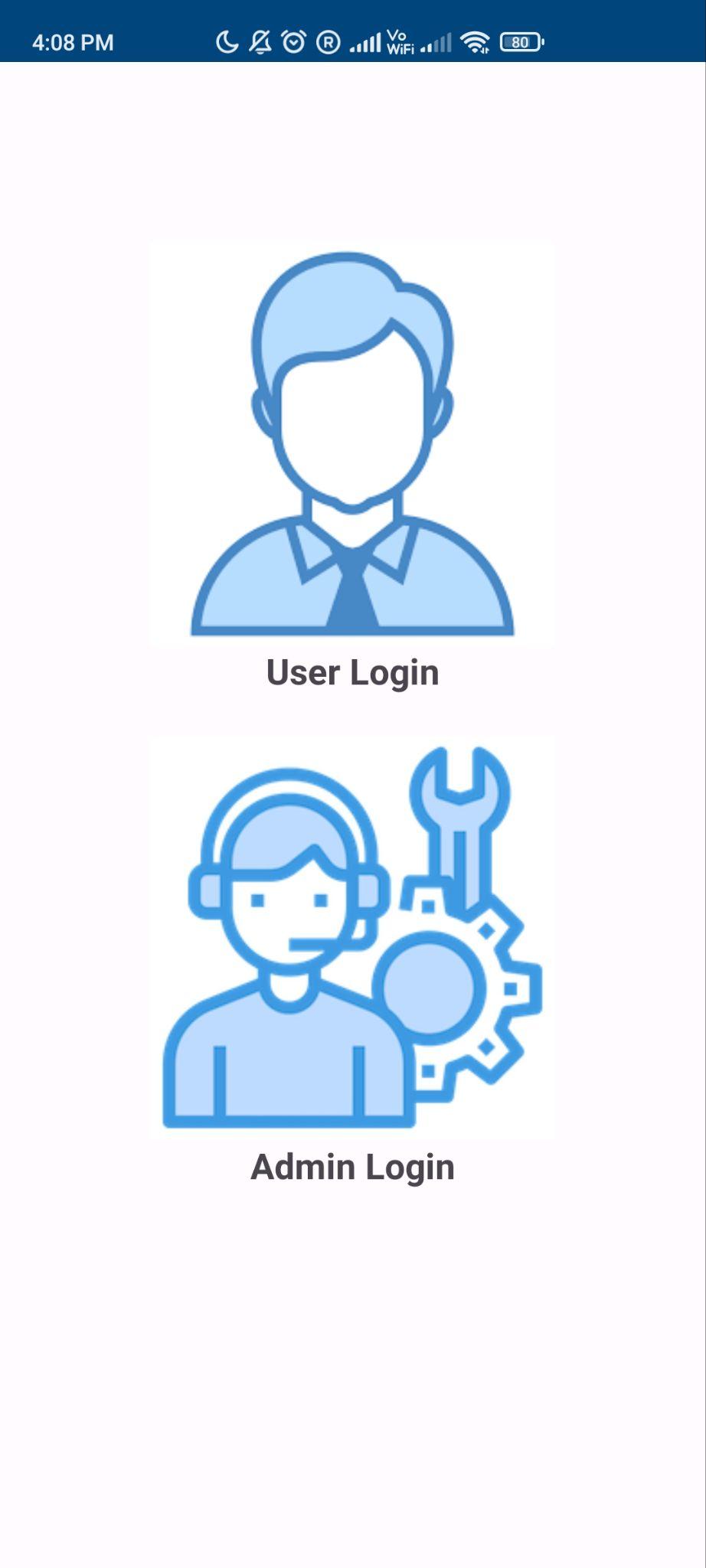
### 

### State diagram

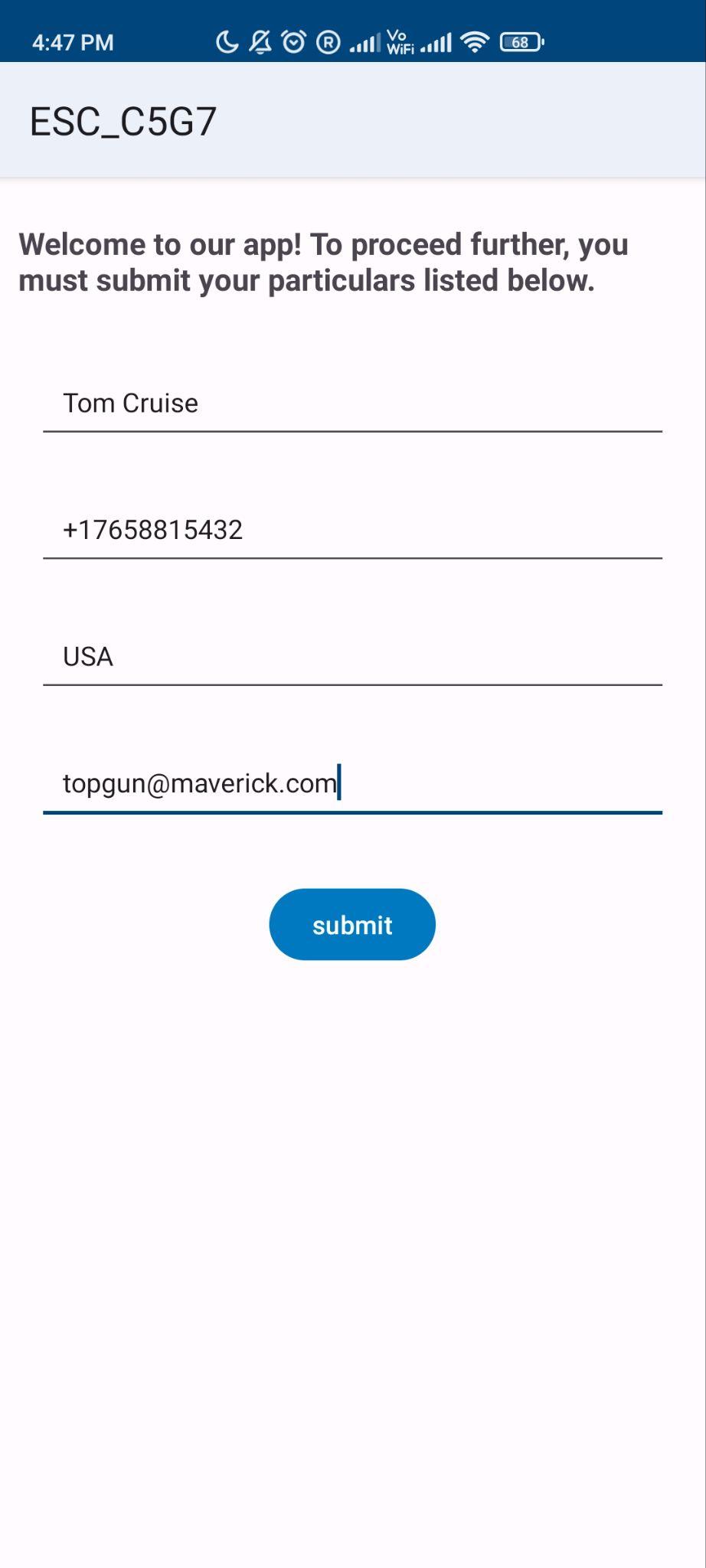
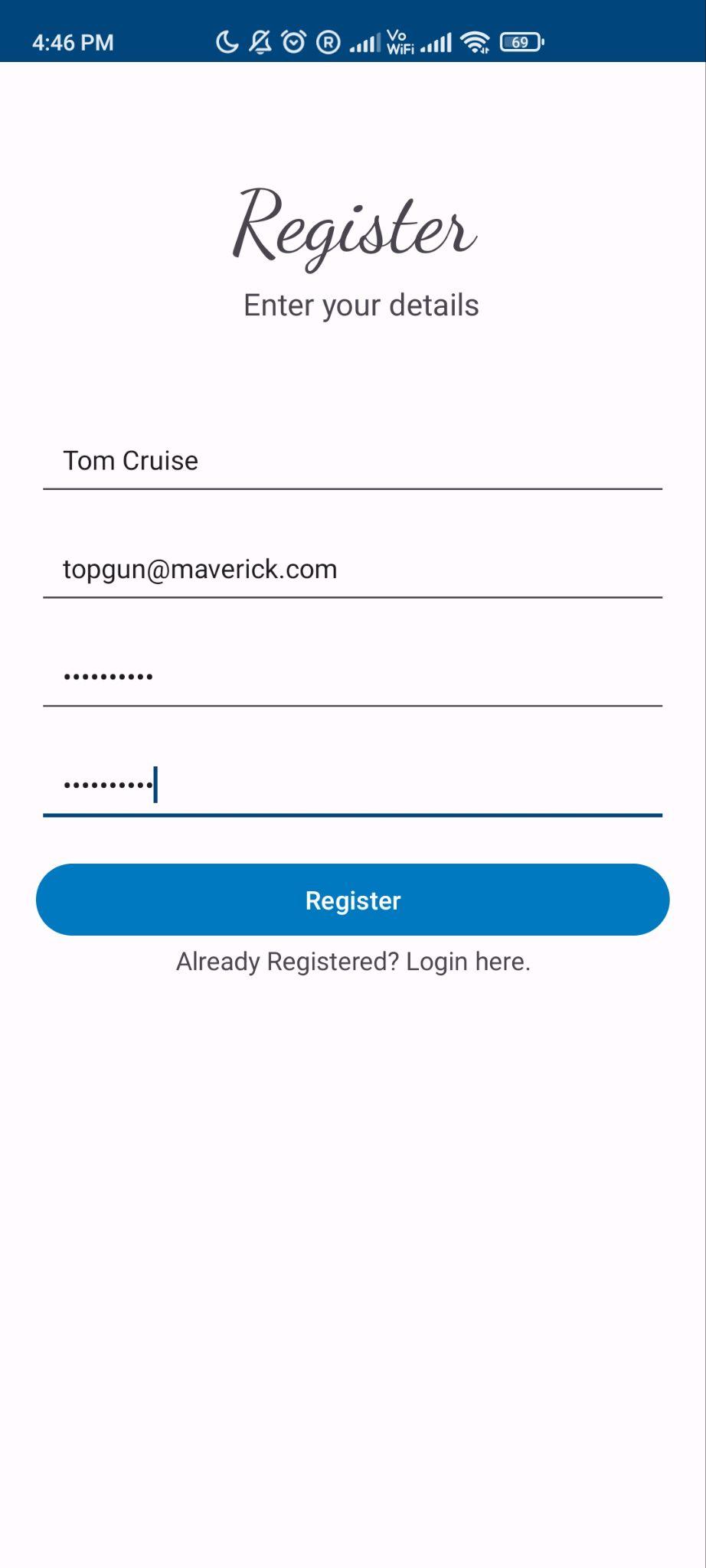
## 

## 

Application Interface



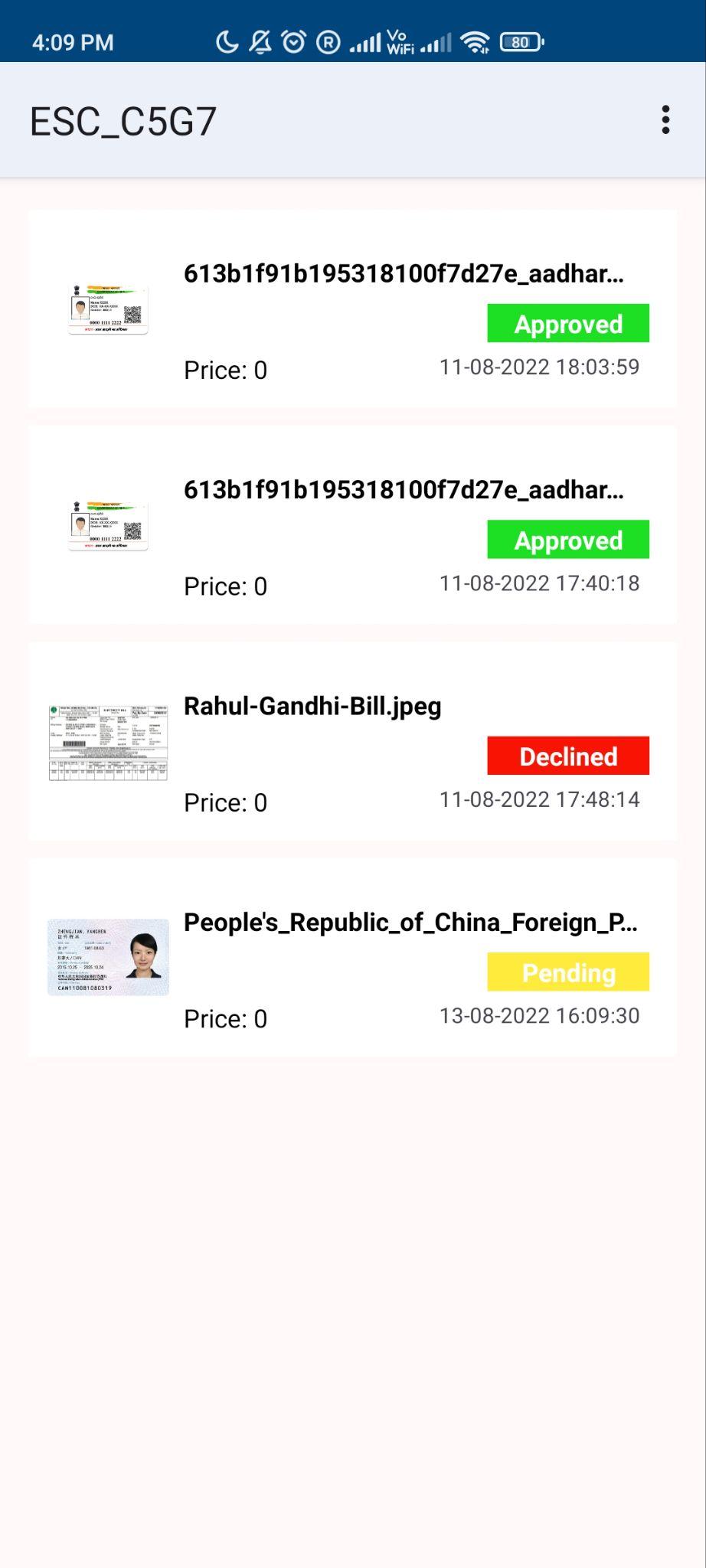
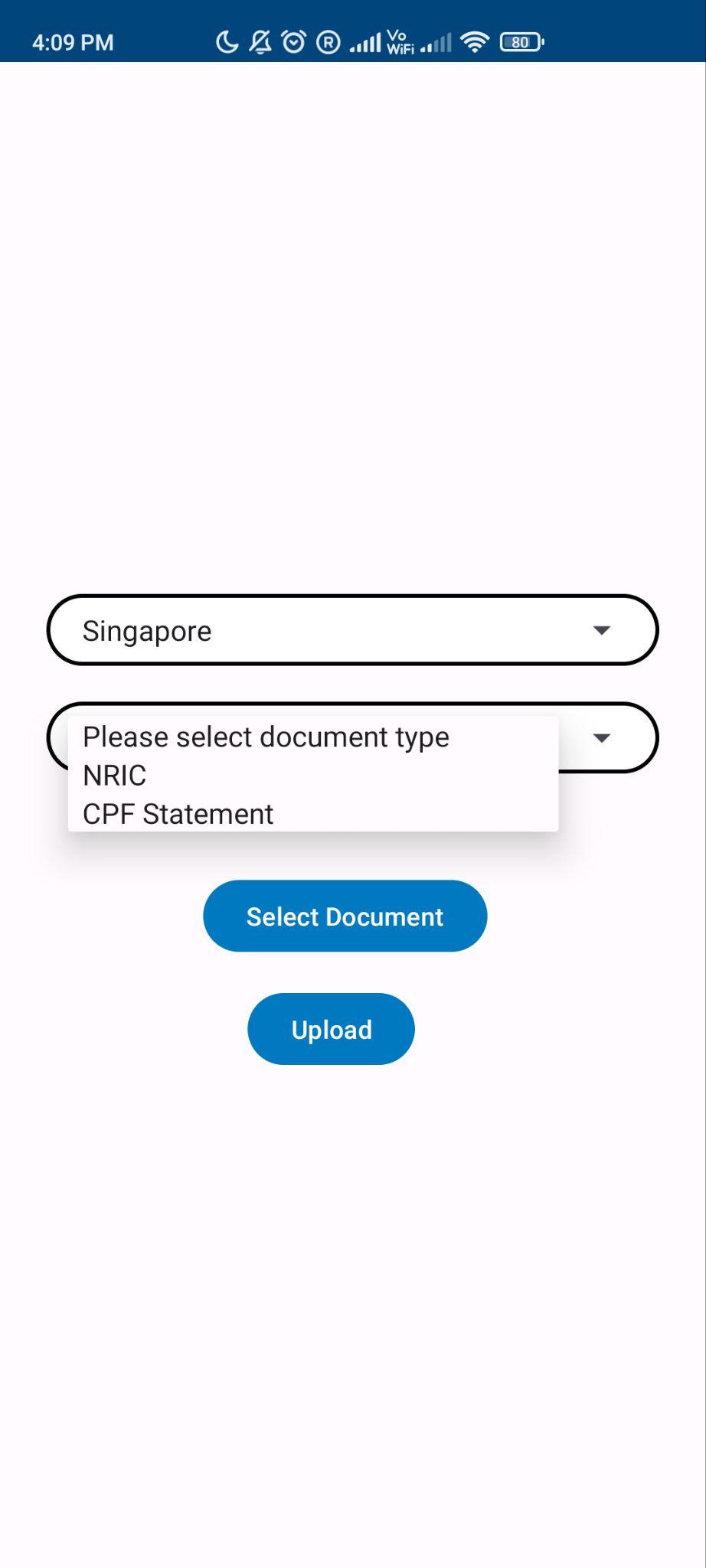
**Homepage and Choose Login Page**

****

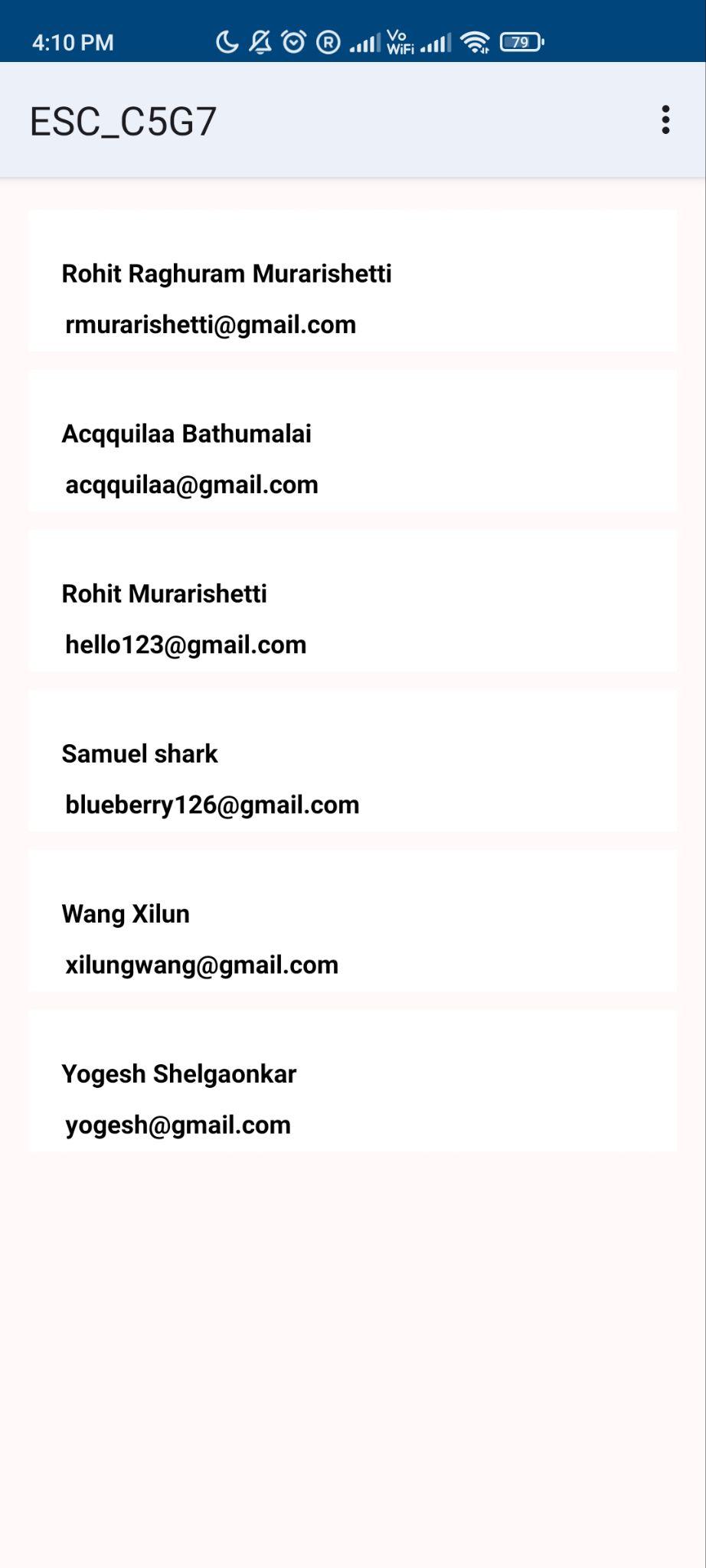
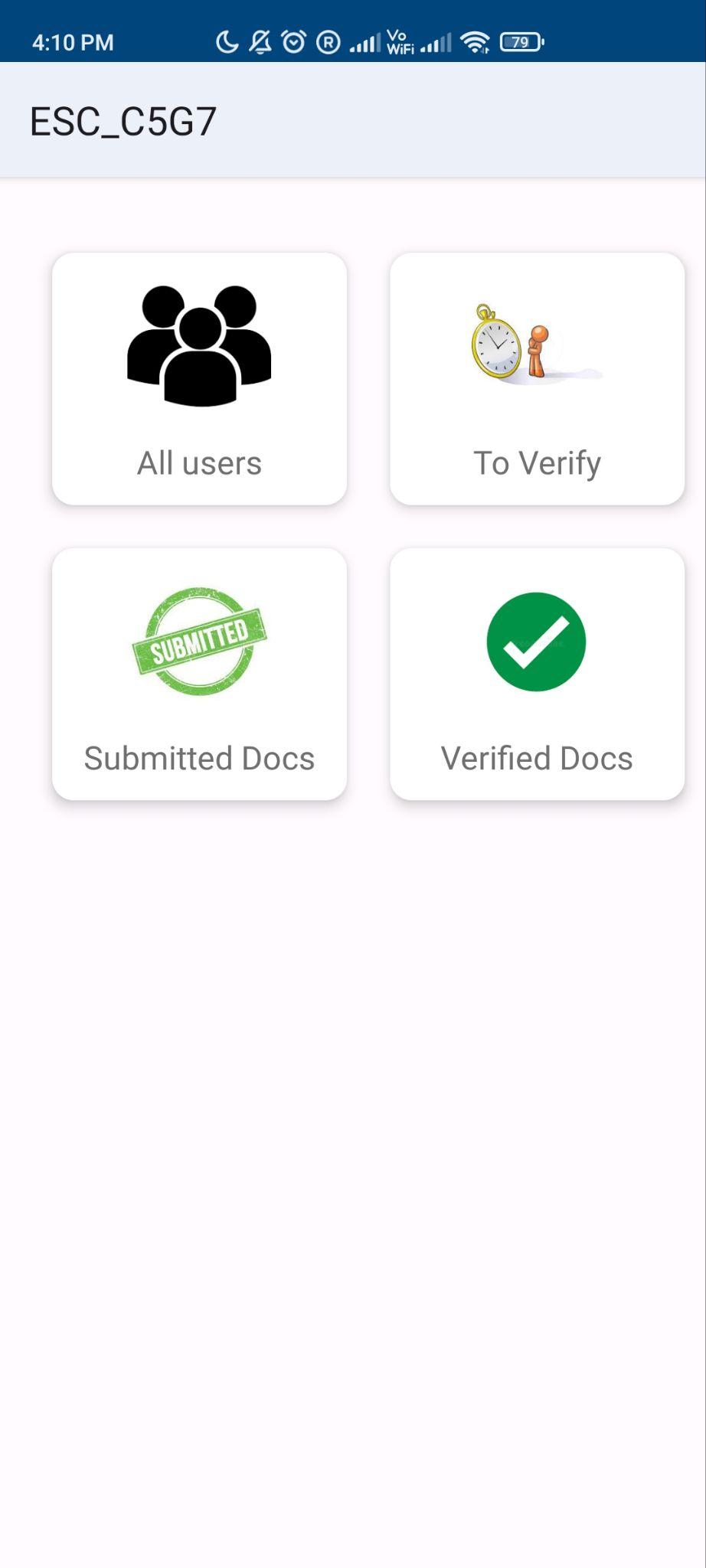
**User Registration and User Particulars Page**

#### 

**User Login and User Home Page**

****

**Upload Document and Show Record Page**

****

**Admin Home and All Users Page**

#### 

**Database Uploaded Record Document Structure**

#### Challenges

#### Backend

For creating an Identity verification platform, we needed a robust backend implementation to tackle three issues:

1. Seamless user registration and user creation.
2. Admin Side Viewing and Handling of Document Submissions.
3. Database and Document Structures to store User Particulars, Submissions records and Credentials.

We made sure our application has a seamless and easy user registration process for an easy adaption to the application. This was done using the firebase API for user authentication. Then we store our user particulars in the Firebase Firestore Database and can retrieve important information like Email, Phone Number and Country particulars for each of our user.

We have also come up with an elaborate document structure for our database to support upload of documents and understand its attributes easily. This was important for us since we had to keep a track of which user submits what records and how an admin user can view this on the admin side of our application.

On the admin side of the application, we query the document submissions in our real time database based on the user ID and the document verification status. For example, if we want to find all the verified documents to display on the admin side of our application, we query for all the users and the document submissions stemming from those User ID’s and check their submission status.

#### Frontend

The biggest challenge we faced for implementing the front end of the application was to maintain simplicity. It is important for even remotely savvy users to understand and use our application. The whole aim of the application is to bring universal access to a globally accepted digital identity.

For this, we kept our user registration simple and elegant. The document submission page is also very simple which doesn’t bother the user to fill in extra details apart from submitting the document.

They can also check their documents submission status by just a simple click on the user home page and use it as a digital identity when the status turns verified.

On the admin side, our regulators can view how many users are there and the submissions attributed to their accounts. We have further made it easier for them to find submissions which are pending only or verified only through the Pending and Verified tabs on our admin home page. We have used simple recycler view adapters to display all our results in the admin side.

### 

### Testing Challenges

#### Testing Backend

**Create Junit Test**

When testing the backend, we think about how to create the test cases. According to the Junit test we learned in class, we choose to use the Mockito framework. We give input and compare the output with the exception. But when we used the Junit test, we found that Junit can not fit our app.

**Junit Test challenges,**

In conclusion, the JUnit test cannot behave well in this project.

People use JUnit tests depending on the clear Test case(Input and output). For a normal unit test, people give an input and then use Assert API to check the output.

However, In this project, the input and output are not clear enough.

Input: Not an exact number or string.

Output: The exception should be like “The method works”.

In our project, we test the accuracy of Register and Login activities.

we try to use JUnit in these 3 activities:

* UserLoginAvtivity
* AdminLoginActivity
* RegisterActivity

If we want to test other activities, we need to observe the output/result on the virtual device. Our project is an app for uploading files to the database. As an app on the phone, if we want to test whether the activity works or not, we should observe the result on the virtual device(Android Studio). In this case, the espresso test fits our project better.

#### Testing Frontend

We made use of Espresso to write frontend tests in Android Studio. This made it easier for us to automate the UI testing to simulate usage by users. The table below shows the user activities we have simulated.

|  |  |
| --- | --- |
| Class | Tests |
| Choose Activity | 1. Choose Admin Login 2. Choose User Login |
| Admin Login | 1. Admin should be able to login with the correct details 2. Admin should not be able to login with no email entry 3. Admin should not be able to login with incorrect or empty password |
| User Login | 1. User should be able to login with the correct details 2. User should not be able to login with no email entry 3. User should not be able to login with incorrect or empty password 4. Unregistered User should not be able to login |
| User Registration | 1. User should be able to register with required details correctly entered 2. User should not be able to register with empty fields 3. User password should be of length longer than 6 characters |
| Particulars | 1. All fields (name, email, number, country) should be filled in correctly |
| Upload Document | 1. Document type should match required type |
| Check Record | 1. User should be able to check status of uploaded documents 2. User should be able to view previously uploaded documents |

When simulating the tests above, we need to ensure that the app does not crash when an invalid input is entered by the user and works as expected when the user has entered valid tests.

One of the difficulties we faced was testing if the intents were working as the testing was done simultaneously with the development of the application. We resolved this by constantly updating each other on our progress in application development and coordinating with each other when we finish the testing and activity assigned to us. Once we integrated all our activities, we did a full UI test to simulate a full user experience.

## Lessons Learnt

### Software Development Process followed

After looking at the problem statement, it was clear that we had to develop an android application. However, we were unsure of the details required for the application to fulfill the requirements. Thus, we took an iterative and incremental approach, where a certain amount of code and functions were implemented weekly. We reassessed whether there were any additional features or any improvements to the existing features required. Both the frontend and backend were developed using iterative and incremental approaches. The relevant stages of development are described below.

The initial application tried to replicate the functionality of PayPal, whether a user is able to log in and make use of certain financial services. We decided this was a good starting point, and we should implement new features to meet our end goal of creating a digital identity for the users.

The next stage involved verifying the identity of the users. We quickly realized that the main way of doing this was through official documents, which could be verified by official institutions. It was also important that the application should be able to take in documents from many countries, as we wanted to allow immigrants to use their digital identities in foreign countries. We zoned in on a few popular countries such as India, China, USA, and Singapore. They had to be sent to the relevant authorities to verify the documents, informing us whether the document had been verified. As we could not access real government or bank databases, we decided to simulate this process using google firebase. The documents submitted would be stored in firebase, where we can manually verify or reject the submitted documents, which would be similar to the responses between PayPal and the relevant institution.

The last feature we chose to implement was an admin side to our app. A system administrator would be able to log in and see the list of users with the documents that are pending verification and those that have been verified. This was implemented for ease of convenience for the admins, where they could see the statuses using their phones, compared to logging on using their computers.

### Challenges faced following the process

When we had first started coding, it was difficult to split the work among team members, as the application had to be developed every week. There were classes to be implemented that would extend another class that had to be coded that week. To overcome this, we split the work into UI development and backend development. While some progress would still overlap, it was a far better alternative than waiting on one another to finish their part of the code.

Another issue we faced was the lack of knowledge about using firebase to reflect a change in the application, as part of our application had to show, in real-time, whether a document had been verified. We had to learn how to implement the functionality we needed through google and youtube. It was a long process, and we were inexperienced in this area.

### Reflection and Future Improvements

When we first chose this project proposal, we underestimated the amount of work and resources that building this application and testing it would take. The document upload protocol and the document structure were the most time taking sprints we had on the application development side. We also had to research the most universally accepted identification documents from India, China, the EU, and the USA to list those documents for submission on our document submission page.

The document structure had to store all the critical attributes of country, document type, user details, and submission status. While we sprinted to finish the user side of the application, there remained a deficiency of resources that we could allocate for comprehensive admin side features. In hindsight, this was a significant problem to tackle as a robust admin side would allow regulators to verify documents directly rather than from the database. Another problem left unaddressed was the integration with financial institutions as an API for identity verification, and KYC services help users access complicated financial services that are otherwise not accessible.

In conclusion, we had dedicated extreme focus to the ease of user registration and simplicity of using our app, which may have diverted our attention from some other important aspects of the problem. In the future, we hope to improve the integration with finance institutional clients for user identity verification.

## Deliverables

GitHub link: <https://github.com/rmurarishetti/ESC_C5G7>

Application APK:

<https://drive.google.com/file/d/139efKXRz4AfRjoCK0JXrjrmiafk6U6SM/view?usp=sharing>

## Contributions

|  |  |
| --- | --- |
| **Team Member** | **Contribution** |
| Acqquilaa Bathumalai | UI Testing, Admin Home Page |
| Rohit Raghuram Murarishetti | UI Testing, User Login, Admin Login, User Particulars Page, Fuzzing |
| Yogesh Shelgaonkar | Fuzzing, Document Submission, User Verification Page |
| Wang Xilun | Junit Testing, Verify Document Page |