# About

In this project, we created an app to demonstrate the capabilities of machine learning and blockchain technology in a topic related to Information Assurance and Security. Our agreed upon idea is to create an app where users could access their accounts using facial recognition and then pay other users using a crypto currency. After the payment is done, block chain is used to save the transactions in a ledger that must be saved in very user’s device. In case of a loss or damage in the original database, this ledger could act as a backup plan to retrieve the data by comparing the ledgers in every user device and vote for the most authentic ledger.

Before diving in, it is worth mentioning that although the use of the logic and the technology are well demonstrated in this app, the app must be considered as merely a prototype or a proof of concept. There are a lot of loose fittings here and there that could be improved. The intention behind the app was to demonstrate the idea itself and not to come up with a high-end-production-ready product. That being said, the app files could be found on github using the following url: <https://github.com/Hindawi91/Payment-App-with-Facial-Recognition-and-Block-Chain->

# How The App Works

## Data Setup

The app starts by manually setting up the database which consists of a number of steps:

1. Adding pictures of the users to the “data” folder available in the working directory of the app. To demonstrate, we added celebrity images and the image of the student responsible for developing the app (Firas Al-Hindawi). Figure1 shows the data folder and the images of the celebrities.

Graphical user interface, text, application, Word

Description automatically generated

1. Adding the names of the users from the previous step into a simple excel sheet “database.xlsx” (which is also available in the working director of the app) alongside their current amount of crypto currency they have available. The excel sheet mentioned in shown in figure2.

Graphical user interface, application, table, Excel

Description automatically generated

## Using The APP

To start the application, we must first run the “main.py” file that is available in the working directory. Once the “main.py” files started running, the Graphical User Interface (GUI) of the application starts. The GUI was built using the PySimpleGUI python library. Figure3 shows the main window of our application.

Graphical user interface, application

Description automatically generated

Once the user clicks on the “Login using face ID” button, the following happens:

1. The app uses the “face\_recognition” library to extract the face encodings from the images inserted in the “data” folder and then save them as a numpy array called “faces\_encodings.npy” in a folder called “face\_encodings”. The app also takes the name of the users from the images names and also stores them in a numpy array called “faces\_names.npy” in the same folder. Figure4 shows a screenshot of the mentioned folder and files.

Graphical user interface, application, table

Description automatically generated

***Note: extracting the faces encodings from the images takes a considerable amount of time, for that reason the app was programmed to check if the “faces\_encodings” folder exists and extract the saved numpy arrays instead of doing the entire process all over again. If you want to test the code on a new data, you must delete the “face\_encodings” folder and add the users as mentioned in the previous section (both data folder and excel database).***

1. After the app retrieves the users names and images encodings, the app uses the cv2 python library to open up the device’s camera and process what is being captured by the camera in a frame by frame manner. For every frame the app does the following processes:
   1. Check if there are any faces in the current frame, if there are no faces in the current frame, move on to process the next frame.
   2. If the current frame does have a face in it, the app locates the face and then extract it’s encodings.
   3. If the encodings of the captured face do not match any of the encodings in our faces encodings database (faces\_encodings.npy file), the app draws a rectangle around the located face and prints “unknown” below the rectangle surrounding the located face. The app then moves on to process the next frame. Figure5 shows an example of when a user who is not in the database trying to login.

Graphical user interface, application

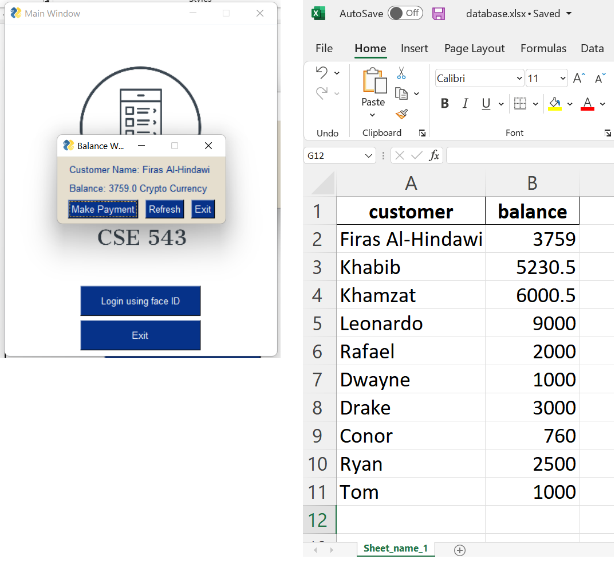
Description automatically generated

* 1. If the encodings of the captured face do in fact match any of the encodings in our faces encodings database (faces\_encodings.npy file), the app draws a rectangle around the located face, prints the user name below the rectangle surrounding the located face and prints “name captured” on the screen. The app then closes the camera and opens the balance window. Figure6 shows an example of when a user who is in the database trying to login.

Graphical user interface, website

Description automatically generated

1. Once the user is recognized and granted access, the app opens up the balance window. This window shows the user’s name and balance amount available in Crypto Currency as stored in the excel database file. Figure7 shows the balance window that should appear once the user is authorized alongside a screenshot of the excel data base.

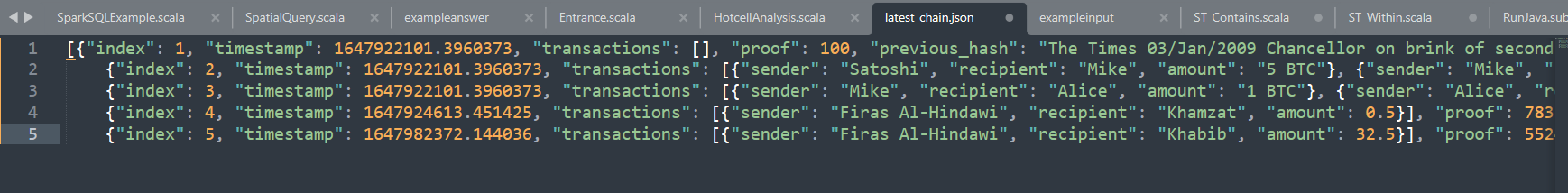


1. If the user wants to make a payment to another user, he must click on the “Make Payment” button in the balance window. Once the button is clicked, the app opens up the “Payment Window” as shown in figure8. The payment window consists of two textboxes. The first textbox requires the user to specify the name of the receiver and the second textbox requires the user to specify the payment amount in crypto currency.

Graphical user interface, application

Description automatically generated

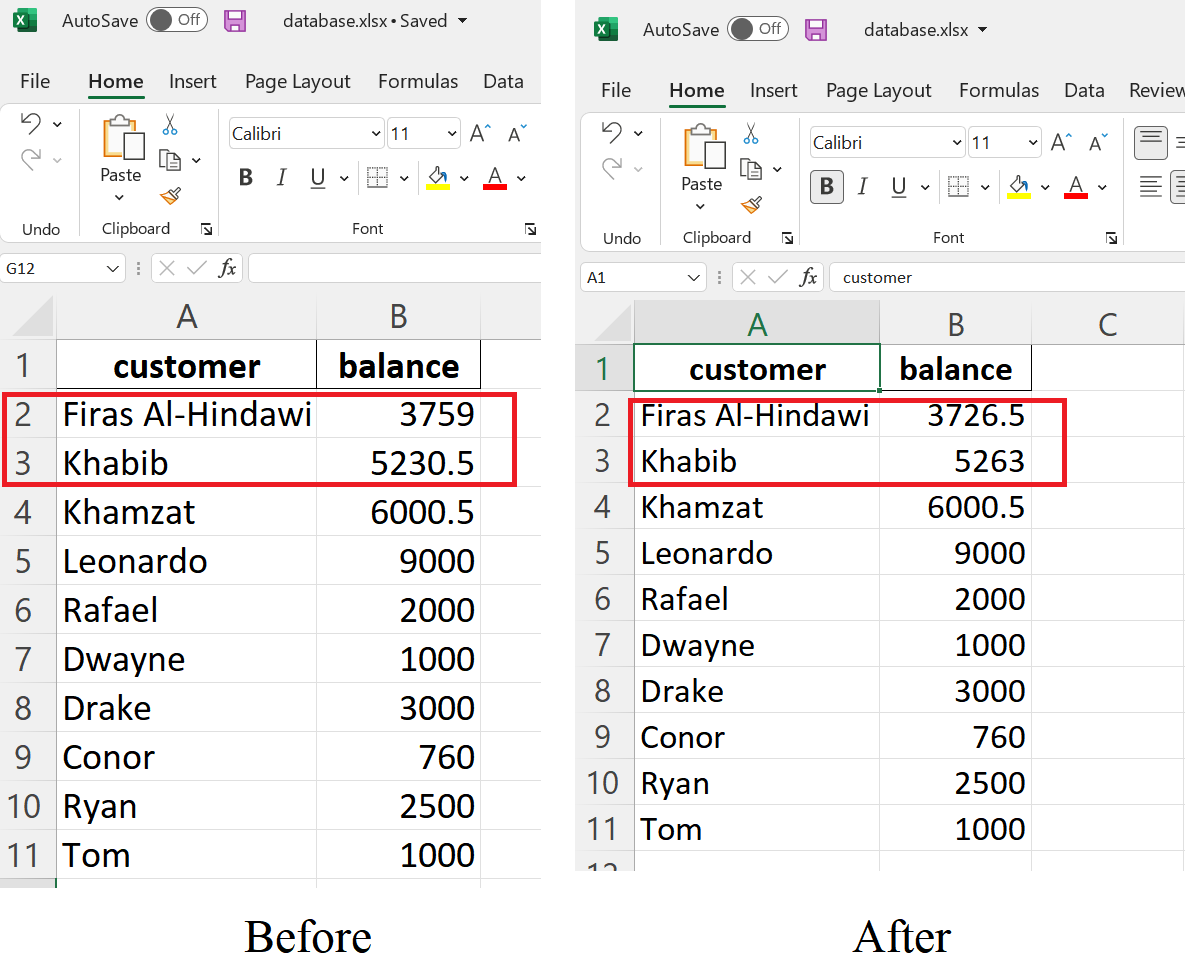
1. If the receiver’s name entered already exists in the database and if the amount to be sent is not larger than the user’s available balance, then the app does the following:
   1. It loads the latest saved block chain from the JSON file called 'latest\_chain.json'.
   2. It creates a new block using the current transaction with the index, timestamp, proof and previous hash and attached the new block to the pre-existing blockchain and then updates the blockchain JSON file. In theory, the app should also update the new chain in all registered users’ devices. Figure9 shows a screenshot of the JSON file (it supposed to be in one long line, but it is shown here re-arranged for viewing purposes only)



* 1. Finally, the app updates the balances for both the sender and the receiver and shows a message confirming that the transaction was successful like in Figure10. Figure11 shows how the amount sent was reflected in the excel database.

Graphical user interface

Description automatically generated



## Remarks about the App

As previously mentioned, this app is merely a demonstration of the capabilities of machine learning and blockchain technology in the fields of Information Assurance and security. It is by no means an error proof product. That being said, in this section we will be discussing the future work necessary for this app to be scaled for a real world application and for it to work better.

The following is a list of the suggestions that we suspect could help the app reach the next level:

1. Because this is a prototype, the app was created only as a windows application and not as a mobile application. Most users nowadays rely heavily on mobile applications for their financial transactions It would be very important to scale this app to run on Android and iOS systems as well.
2. Using an excel sheet as a database will only work for a very small app such as the one we have on hand which only serves as a prototype or a proof of concept. In reality, we would need a data base management software to help serve a large number of users. We suggest using an online software solution such as Google’s Firebase-Realtime Database system.
3. The app relies on an off the shelf machine learning library specialized in facial recognition. This is all well in good for our purpose to demonstrate the capabilities of machine learning, but for an end product app this is far from acceptable because while we were experimenting with the app we noticed some shortcomings of this library that would be critical for a financial application. For example:
   * Sometimes when a user that is not registered in the database tries logging in, the app will allow him to access if he resembles any of the users. This is of course a major threat for the information security of the app, but it could be solved by tuning the settings and parameters in the functions provided by the library, such as the tolerance parameter responsible for the sensitivity of the facial recognition feature.
   * Another major security problem is that the library allows for facial recognition of non-live images. For example, in Figure5 I was able to use an image of Mohammad Ali from the screen of my phone and the app was not immune to recognize the image. If a malicious attacked used an image of a user registered in the database, he will be able to access his wallet and transfer the money as he please. This problem could be solved by using other different sensors on the phone that would allow to detect whether the it is a 2D or 3D image. Another solution is to train a CNN classifier that is able to predict if the person in the camera video frame is real or not.
4. The block chain part of the application definitely has a lot of space for improvement. Our use of the technology is as a backup to save the authenticity of the ledger in case of a damage in our database. One of the things that should be improved is to have the app update the JSON file containing the latest saved chain on every user’s device. Since we do not have actual users to send the file to, we did not bother with that step. The concept however remains consistent and serves the prototype purpose behind this app.
5. The Graphical User Interface (GUI) of the application was made to be very simple and basic. Just good enough to serve it’s purpose as a proof of concept. Therefore, it has a lot of space for adding more graphics and features. It will also need more debugging to improve the performance and user experience.