USERS and DRIVERS table

```
==================================================================================
Client                       | Process               | Database
==================================================================================
* On Sending:
----------------------------------------------------------------------------------
- Username              ---> Encrypt            ---> Store (displayed)
- Full Name             ---> Encrypt            ---> Store (displayed)
- Password              ---> Hash               ---> Store (displayed)
- Security Pin          ---> Hash               ---> Store (NOT displayed)
- Balance               ---> Encrypt            ---> Store (displayed)
- BalanceVerification   ---> Hash               ---> Store (NOT displayed)
==================================================================================
```

```
==================================================================================
Client                       | Process               | Database
==================================================================================
* On Retrieval:
----------------------------------------------------------------------------------
- Username              <--- Decrypt            <--- Store (displayed)
- Full Name             <--- Decrypt            <--- Store (displayed)
- Password              <--- No Process         <--- Store (displayed)
- Security Pin          <--- No Process         <--- Store (NOT displayed)
- Balance               <--- Decrypt            <--- Store (displayed)
- BalanceVerification   <--- No Process         <--- Store (NOT displayed)
==================================================================================
```

Pseudocode for Verifying Data:

```
// Storing
input() -> rawData;
encrypted(rawData) -> secureData;
hash(rawData) -> verificationData;
sendToDatabase(secureData,VerificationData);
CheckIfEqual(secureData,verificationData)
if(CheckIfEqual)
    return Proceed();
else {
    return NotProceed();
}

// Retrieval
database -> encryptedData;
database -> verificationData;
decrypted(encryptedData) -> rawData
hash(rawData) -> hashedInputData;

if (checkIfEqual(hashedInputData, verificationData)) {
    return Proceed();
} else {
    return NotProceed();
}
```

Notes for Passwords, Security Pins, and Verification:

- On input, the raw data is **hashed**.
- Retrieved data from the database is **compared** to the hash of the input.
- If **equal** ➔ Proceed.
- If **different** ➔ Possible wrong input or database tampering.
- BalanceVerification is a data retrieve from balance needed to check if data has been manipulated or not
    - Using hash to prevent manipulation, since hash use a disruption method
        - any slight manipulation would change a lot inputted data on the hash

Access

- Admin

- It's owner

`ORDERS` table

```
=================================================================================
Client                      | Process           | Database
=================================================================================
* On Sending:
---------------------------------------------------------------------------------
- id                        ---> No Process     ---> Store (NOT displayed)
- idVerification            ---> HASH           ---> Store (NOT displayed)
- username                  ---> Encrypted      ---> Store (displayed)
- driver_username           ---> Encrypted      ---> Store (displayed)
- madeTime                  ---> Encrypted      ---> Store (displayed)
- finishTime                ---> Encrypted      ---> Store (displayed)
- from                      ---> Encrypted      ---> Store (displayed)
- fromVerification          ---> HASH           ---> Store (NOT displayed)
- destination               ---> Encrypted      ---> Store (displayed)
- destinationVerification   ---> HASH           ---> Store (NOT displayed)
=================================================================================
```

```
=================================================================================
Client                      | Process           | Database
=================================================================================
* On Retrieval:
---------------------------------------------------------------------------------
- id                        <--- No Process     <--- Store (NOT displayed)
- idVerification            <--- No Process     <--- Store (NOT displayed)
- username                  <--- Decrypted      <--- Store (displayed)
- driver_username           <--- Decrypted      <--- Store (displayed)
- madeTime                  <--- Decrypted      <--- Store (displayed)
- finishTime                <--- Decrypted      <--- Store (displayed)
- from                      <--- Decrypted      <--- Store (displayed)
- fromVerification          <--- No Process     <--- Store (NOT displayed)
- destination               <--- Decrypted      <--- Store (displayed)
- destinationVerification   <--- No Process     <--- Store (NOT displayed)
=================================================================================
// Storing
input() -> rawData;
encrypted(rawData) -> secureData;
hash(rawData) -> verificationData;
sendToDatabase(secureData,VerificationData);
CheckIfEqual(secureData,verificationData)
if(CheckIfEqual)
    return Proceed();
else {
    return NotProceed();
}

// Retrieval
database -> encryptedData;
database -> verificationData;
decrypted(encryptedData) -> rawData
hash(rawData) -> hashedInputData;

if (checkIfEqual(hashedInputData, verificationData)) {
    return Proceed();
} else {
    return NotProceed();
}
```

Notes for Verification:

- On input, the raw data is **hashed**.
- Retrieved data from the database is **compared** to the hash of the input.
- If **equal** ➔ Proceed.
- If **different** ➔ Possible wrong input or database tampering.
- Verification is a data retrieve from balance needed to check if data has been manipulated or not
  - Using hash to prevent manipulation, since hash use a disruption method
    - any slight manipulation would change a lot inputted data on the hash

Access

- Admin
- It's own User
- It's own Driver

USER_PAYMENTS and DRIVER_PAYMENTS table

```
=================================================================================
Client                       | Process           | Database
=================================================================================
* On Sending:
---------------------------------------------------------------------------------
- id                         ---> No Process     ---> Store (NOT displayed)
- idVerification             ---> HASH           ---> Store (NOT displayed)
- orders_ID                  ---> Encrypted      ---> Store (displayed)
- orders_IDVerification      ---> HASH           ---> Store (NOT displayed)
- price                      ---> Encrypted      ---> Store (displayed)
- priceVerification          ---> HASH           ---> Store (NOT displayed)
- paymentTime                ---> Encrypted      ---> Store (displayed)
- paymentTimeVerification    ---> HASH           ---> Store (NOT displayed)
=================================================================================
```

```
=================================================================================
Client                       | Process           | Database
=================================================================================
* On Retrieval:
---------------------------------------------------------------------------------
- id                         <--- No Process     <--- Store (NOT displayed)
- idVerification             <--- HASH           <--- Store (NOT displayed)
- orders_ID                  <--- Decrypted      <--- Store (displayed)
- orders_IDVerification      <--- HASH           <--- Store (NOT displayed)
- price                      <--- Decrypted      <--- Store (displayed)
- priceVerification          <--- HASH           <--- Store (NOT displayed)
- paymentTime                <--- Decrypted      <--- Store (displayed)
- paymentTimeVerification    <--- HASH           <--- Store (NOT displayed)
=================================================================================
// Storing
input() -> rawData;
encrypted(rawData) -> secureData;
hash(rawData) -> verificationData;
sendToDatabase(secureData,VerificationData);
CheckIfEqual(secureData,verificationData)
if(CheckIfEqual)
    return Proceed();
else {
    return NotProceed();
}

// Retrieval
database -> encryptedData;
database -> verificationData;
decrypted(encryptedData) -> rawData
hash(rawData) -> hashedInputData;

if (checkIfEqual(hashedInputData, verificationData)) {
    return Proceed();
} else {
    return NotProceed();
}
```

Notes for Verification:

- On input, the raw data is **hashed**.
- Retrieved data from the database is **compared** to the hash of the input.
- If **equal** ➔ Proceed.
- If **different** ➔ Possible wrong input or database tampering.
- Verification is a data retrieve from balance needed to check if data has been manipulated or not
    - Using hash to prevent manipulation, since hash use a disruption method
        - any slight manipulation would change a lot inputted data on the hash

Access

- Admin
- User_Payment only it own user
- Driver_Payment only it own driver