Project Title: SafeEx ATM Management System.

Name: Ramy Fekry

GitHub username: Ramy1951

Version	Date
V2	11/03/2020
V1	10/06/2020

Table of Contents

Section I: Project Description

ATMs are supposed to be used for securely dealing with money, you pay a bank to make sure your money is safely stored and also safely shared only with whom you would like to share it with! Today I have a proposal of an idea that will make sure that when you send money to someone, you are absolutely sure that you always have the right person and if anything ever goes wrong, by just one click on your app, or one phone call, you can recycle the credentials. Maybe something went south and money went to the wrong person, you can also go into the app and make a report. What makes this report different? This report will automatically pull all the footage and ID presented in the transaction to incriminate the fraudster. Doesn't that sound like a great plan, the very people you are already paying fees for should also make sure that you are safely transferring your money. Why use a different service to transfer your money and pay them even more fees if the very people you are already banking with can do that for you?

Today I present SafeEx. SafeEx is a feature where an account owner will be able to transfer cash to anyone using their phone or any nearby ATM. The money being transferred will be given to the recipient in cash and can be pulled from any ATM in the world, in any currency. By using any ATM, the recipient can take the cash while being physically recorded, and have their identification verified making sure the name matches the one provided as the recipient. This way there is no need for a teller, or an open bank branch, to complete the transaction. All this convenience without sacrificing any of the security. It is very convenient for both the users and the recipients. SafeEx

adds convenience all with an extra layer of security of recorded transaction with no possibility of human error. Based on a survey conducted by mybanktracker.com, the 3rd most common reason why tellers get fired is because of “missing the details” which goes to show that trusting a machine that will always retrieve the information you put and base transactions on it, ensures that you have the ultimate control of keeping your information accurate, and up to date.

With Paper checks users always have to protect them and if the checks ever get stolen users are dependent on being able to get to the customer service line to wait even longer before they are able to put a hold on the checks. Just when users think that’s it, they still have to wait even longer for the new checks and cards. With SafeEx that is no more an issue, you never have to buy checks and worry about misplacing them, or them falling into the wrong hands. Today your phone is with you wherever you go, if anything happens and your credentials for the transaction are leaked, with a simple press of a button you could easily issue a new set of credentials without having to go through the hassle of waiting for new checks or changing your debit or credit cards.

Section ||: Use Cases

1. John wants to send cash to his relative Khaled that lives overseas using SafeEx. As a prerequisite for the use of the feature, John has to have a bank account that uses SafeEx. SafeEx will ask John about Khaled and is prompted to enter the amount to be sent. John is provided a set of credentials to be sent to Khaled. Khaled goes to pick up the money from the nearest ATM and is prompted to enter his credentials provided by John. Khaled is then prompted to verify his identity using his government issued ID. Khaled‘s identity gets verified using his government issued ID, the ATM keeps a copy of his ID card, and records the transaction. Now Khaled can choose the currency he would like to receive the money in and once the transaction is done the money is dispensed, and notification of completion is sent to John’s cell phone.

2. John would like to buy a fridge from Kevin, who is John’s neighbor that he met through a garage sale. Kevin does not have a bank account that John can send the money to, and Kevin prefers to have a check or cash for the transaction. John suggests that he could give Kevin a set of credentials that he could use to grab the money from an ATM, since John does not like to carry cash and does not have checks. John goes on his phone and initiates a transaction using SafeEx. John sends Kevin the credentials he needs to receive the money from the ATM. Kevin takes his government issued ID and heads over the closest ATM,

Kevin is first prompted to enter the credentials provided by John, and then is prompted to insert his government issued ID. Kevin has been verified and is lastly prompted to choose the currency to receive the money in. Once the money is dispensed to Kevin, John gets a notification on his phone which indicates the end of the transaction.

3. Nina is John's grandmother and she doesn't feel safe using her phone to do transactions that include money. Nina wants to send money to John, her grandson, for his birthday and would prefer to use an ATM to initiate the transaction to send money overseas. Nina locates the nearest ATM and is prompted to insert her debit bank card to initiate the SafeEx transaction. Nina specifies the recipient and the amount, and then is presented with a receipt that has the transaction credentials that are to be sent to John. Nina sends a picture of the credentials to John. John goes to the nearest ATM and enters the credentials first, then is prompted to insert his government issued ID. John chooses the currency to cash the money in. John is dispensed the money and Nina gets a notification of the money release which marks the end of the transaction.

4. John would like to send money to Nina for her surgery that is coming up soon and would like to send the money as soon as possible. John initiates the transaction through his mobile app and sends the credentials to Nina. Nina then goes to the closest ATM and tries to withdraw the money. She is first is prompted

to put in the credentials that John had sent her, but she is having hard time seeing inputting the required credentials. A bystander offers his help and puts in the credentials for her. Nina is then prompted to insert her government issued ID and is verified. The bystander asks her in what currency would she like to receive the money in and makes the choice based on her answer. The bystander sees the money dispensed and grabs the money and flees the scene. John gets a notification that the transaction is complete and calls Nina to make sure she got the money. Nina explains that the bystander who helped her get the money has taken the money and ran with it. John immediately takes out his phone and marks the transaction as fraud which produces a set of evidence material like the footage of the transaction from multiple camera angles and the government issued ID that was inputted into the ATM which proves helpful when Nina goes to make the police report. The police are able to make out the face and profile of the suspect of the robbery and is able to be located within hours of the incident, all due to the footage that was taken from different angles.

5. John wanted to send money to a stranger that he would like to buy something from on the internet, and he wants to avoid any fees so he decides to use the SafeEx. John initiates the transaction using the SafeEx app and incorrectly inputs the amount of money. John does not realize that he has inputted the wrong amount until his phone runs out of battery. John does not need his phone to edit the transaction information, John only needs his debit card. John knows his neighborhood and locates the nearest ATM and using his debit card he retrieves

the open transaction and changes the amount without changing the set of credentials. When John charges his phone he receives a notification of the money and goes ahead to pick up the purchased item.

Section |||: Database Requirements (Business Rules)

1. ATM
    i. An ATM can initiate zero or one Transaction.
    ii. An ATM must have one Address.
    iii. An ATM must be to provide money in only one currency.
2. Bank Account
    i. A Bank Account belongs to only one User
    ii. A Bank Account must have at least one Bank Card
3. SafeEx Account
    i. A SafeEx account can be created by only one Receiver.
    ii. A SafeEx account can be created by only one Sender.
    iii. A SafeEx account shall have one and only one government issued ID.
    iv. A SafeEx account shall have only one logged in User.
4. User.
    i. User can send only one currency.
    ii. User can create zero or more bank account.
    iii. User can have zero or more transactions.
    iv. User can be the sender and receiver.
    v. A sender must have at least one Bank Account
    vi. A receiver can have zero or more Bank Accounts

vii. A sender must have only one SafeEx Account.

viii. A user shall be able to mark the status (manage) of many transactions

ix. A user shall have only one unique government ID

x. A user can make multiple police reports

xi. A user can change any transaction from any ATM.

xii. A user can have zero or many phone numbers.

xiii. A user can login to only one SafeEx Account.

5. Currency.
    i. A currency can be one payment type.
    ii. A currency can be US Dollar, Euro, Egyptian Pound.
    iii. A currency can be withdrawn from one or more ATM.
    iv. A currency can be sent by many users.
6. Payment type.
    i. A payment type can be Cash, Credit Card, debit Card.
    ii. Cash can be provided in 1 or more currencies.
7. Transaction.
    i. A transaction shall have one and only one user.
    ii. A transaction shall have one and only one credential
    iii. A transaction shall have one copy of receiver government issued ID.
    iv. A transaction shall send one notification when the status is set to completed.
    v. A Transaction must have an amount of money dispensed.
    vi. Many Transactions can be initiated by one Phone App.

vii. Many Transactions can be initiated by one ATM.

viii. A Transaction can have one or more security footage.

ix. A transaction shall be managed by only one user.

x. A transaction can be found by only one Bank Card

xi. A transaction edit does not change the credentials.

xii. A transaction can create only one receipt.

8. Credential
    i. A credential shall be created by only one transaction.
    ii. A credential shall not be changed by an edited transaction.
9. Government issued ID
    i. A government issued ID can belong to zero or many transactions.
    ii. A government issued ID can belong to one SafeEx Account.
    iii. A government issued ID belongs to only and only one user
10. Notification
    i. A notification shall be created by only one transaction.
11. Phone App
    i. A phone App can initiate zero or more transactions.
12. Address
    i. An Address must have one and only one ATM.

13. Bank Card
    i. A Bank Card shall have one and only one Bank Account.

ii. A Bank Card can be Debit Card or Credit Card.

14. Security footage
    i. A Security footage can be tied to only one transaction.
    ii. A Security footage shall have multiple camera angles
15. Camera Angle
    i. A camera angle can be used by many Security footages
16. Police Report
    i. A police report belongs to only one user
    ii. A police report can have one incident
17. Phone Number
    i. A phone number shall have one or more users.
18. Incident
    i. An Incident shall be created by only one police report.
19. receipt
    i. A receipt can be created by one and only one transaction.

Section IV: Detailed List of Main Entities, Attributes and Keys

---

1. User (Strong)
    - user_id: key, numeric
    - name: composite
        1. first_name: alphanumeric
        2. middle_name: alphanumeric
        3. last_name: alphanumeric
    - phone number: weak key, alphanumeric, Phone Number
    - date_of_birth: composite, alphanumeric
        1. month: numeric
        2. year: numeric
        3. day: numeric
    - gov issued ID: key, alphanumeric
    - email: key, alphanumeric
    - password: alphanumeric
    - last_login: alphanumeric, timestamp
2. Police Report (Strong)
    - id: key, numeric
    - created_at: alphanumeric, timestamp
    - description: alphanumeric

3. Address (Strong)
    - id: key, numeric
    - street_address_1: alphanumeric
    - street_address_2: alphanumeric
    - city: alphanumeric
    - state: alphanumeric
    - zip_code: numeric
4. Phone Number (Strong)
    - id: key, numeric
    - country: alphanumeric
    - country_code: alphanumeric
    - number: numeric
5. Transaction (Weak)
    - transaction_id: key, numeric.
    - credential: weak key
    - amount: numeric
    - created_at: alphanumeric, timestamp
    - completed_at: alphanumeric, timestamp
    - title: alphanumeric
    - description: alphanumeric
    - status: alphanumeric

6. SafeEx Account (Weak)
    - safeex_account_id: key, alphanumeric.
    - created_at: alphanumeric, timestamp
    - user: weak key, numeric
7. Currency (Strong)
    - currency_id: key, numeric.
    - ATMs_available: key, alphanumeric
8. ATM (Strong)
    - atm_id: key, alphanumeric
    - address: weak key, alphanumeric
    - money_available: numeric
    - money_capacity: numeric
    - currency: weak key, alphanumeric
9. Bank Account (Weak)
    - bank_account_id: key, alphanumeric
    - balance: numeric
    - created_at: alphanumeric, timestamp
    - bank_card: weak key, alphanumeric, Bank Card
10. Payment type (Strong)
    - payment_type_id: key, alphanumeric
    - transaction: key, alphanumeric

- paid_at: timstamp

11. Notification (Strong)
    - notification_id: key, alphanumeric
    - sent_at: alphanumeric, timestamp
    - transaction: key, alphanumeric, Transaction
12. Phone App (Strong)
    - phone_app_id: key, alphanumeric
    - SafeEx Account: weak key, alphanumeric, SafeEx Account
    - user: weak key, alphanumeric
    - login_date: timestamp
13. Bank Card (Weak)
    - bank_card_type_id: key, alphanumeric
    - card_number: numeric
    - card type: weak key, alphanumeric.
    - expiry_date: timestamp
    - PIN: numeric
    - display_name: alphanumeric
14. Security footage (Strong)
    - security_footage_type_id: key, alphanumeric
    - created_at: timestamp
    - description: alphanumeric
    - title: alphanumeric
15. Camera Angle (Strong)

- camera_footage_type_id: key, alphanumeric
- description: alphanumeric
- title: alphanumeric

16. Recordings (Strong)
    - recording_ID: key, alphanumeric
    - created_at: timestamp
    - security_footage: key, alphanumeric
    - camera_angle: key, alphanumeric
17. User_Addresses (Strong)
    - user_address_ID: key, alphanumeric
    - user: weak key, alphanumeric
    - address: weak key, alphanumeric
    - created_at: timestamp
18. Incident (Strong)
    - incident_ID: key, numeric
    - location: multivalue
    - description: alphanumeric
19. Receipt (Strong)
    - receipt_ID: key, numeric
    - issued_at: timestamp, alphanumeric
    - transaction: key, alphanumeric
20. Sender (Weak)
    - sender_ID: key, numeric

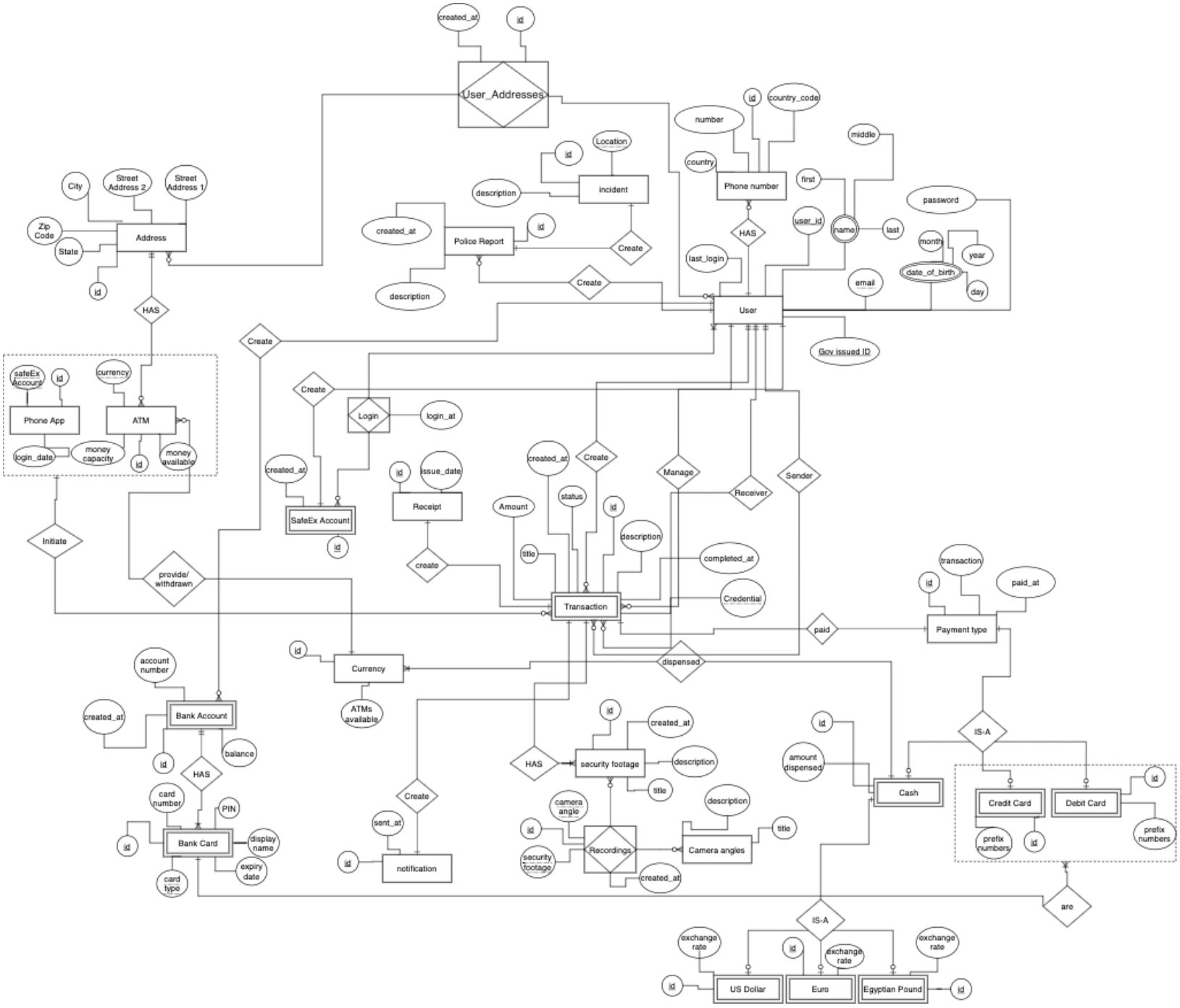
- user: weak key, alphanumeric, User
- transaction_receipt: key, alphanumeric, Receipt

21. Receiver (Weak)
    - receiver_ID: key, numeric
    - user: weak key, alphanumeric, User
    - transaction: weak key, alphanumeric, Transaction
22. US Dollar (Weak)
    - US_Dollar_ID: key, numeric
    - Cash_ID: key, numeric
    - exchange_rate: numeric
23. Euro (Weak)
    - Euro_ID: key, numeric
    - Cash_ID: key, numeric
    - exchange_rate: numeric
24. Egyptian Pound (Weak)
    - Egy_pound_ID: key, numeric
    - Cash_ID: key, numeric
    - exchange_rate: numeric
25. Cash (Weak)
    - cash_ID: key, numeric
    - amount_dispensed: numeric
    - currency: weak key, alphanumeric
26. Credit Card (Weak)

- credit_card_ID: key, numeric
- prefix_numbers: numeric
- payment_type: weak key, alphanumeric

27. Debit Card (Weak)
    - debit_card_ID: weak key, numeric
    - prefix_numbers: numeric
    - payment_type: key, alphanumeric

Section V: Entity Relationship Diagram (ERD)



Section VI: Testing Table

Rule	Entity A	Relation	Entity B	Cardinality	Pass/ Fail	Error Description
1	SafeEx Account	Has	Gov issued ID	1-to-1	Fail	Gov ID cannot be an entity. It has only to an attribute
2	Transaction	Has	Gov issued ID	1-to-1	Fail	Gov ID is turned into attribute, no need for cardinality
3	Transaction	Creates	Receipt	1-to-1	Pass	None
4	Transaction	create	Credentials	1-to-1	Fail	Credentials are only connected to a transaction; only needs to be a composite not an entity.
5	Transaction	Creates	notification	1-to-1	Pass	None
6	Transaction	Has	Security footage	1-to-N	Pass	None
7	User	IS-A	Sender	M-to-1	Pass	None
8	User	IS-A	Receiver	M-to-1	Pass	None

Rule	Entity A	Relation	Entity B	Cardinality	Pass/ Fail	Error Description
9	User	Create	SafeEx Account	1-to-1	Fail	Only a sender needs to have a SafeEx Account, Receiver does not.
10	User	User_Addresses	Addresses	M-to-N	Pass	None
11	User	Belong	Gov issued ID	1-to-1	Fail	Gov ID is only an attribute, no relation of cardinality needed.
12	User	Create	Police Report	1-to-N	Pass	None
13	Police Report	Create	Incident	1-to-1	Pass	None
14	Sender/ Receiver	Create	SafeEx Account	1-to-1	Pass	None
15	Aggregate	Initiate	Transaction	1-to-M	Pass	None
16	Currency	Dispensed	ATM	1-to-N	Fail	A currency can be a strong entity and does not need to be dispensed by at least

Rule	Entity A	Relation	Entity B	Cardinality	Pass/ Fail	Error Description
						one ATM, it can be by zero or more ATMs.
17	Incident	Create	Police Report	1-to-1	Pass	None
18	Currency	IS-A	US Dollar/ Euro/ Egyptian Pound	1-to-1	Fail	Cash is in US Dollar/ Euro/ Egyptian Pound, not Currency.
19	Security footage	Recordings	Camera_angles	M-to-N	Pass	None
20	Aggregate	Create	SafeEx Account	1-to-1	Pass	None
21	ATM	Provide/ withdrawn	Currency	M-to-1	Pass	None
22	Currency	Dispensed	Payment type	M-to-1	Fail	Currency can only be dispensed in cash.
23	ATM	Has	Address	M-to-1	Pass	None
24	Transaction	Paid	Payment type	1-to-1	Pass	None
25	Bank Card	Are	Aggregate	1-to-N	Pass	None

Rule	Entity A	Relation	Entity B	Cardinality	Pass/ Fail	Error Description
26	Bank Account	HAS	Bank Card	1-to-N	Pass	None
27	User	Login	SafeEx Account	1-to-1	Pass	None

Section VII: Database Model/EER

- **Transaction:**
  - Transaction_Receipt_FK:
    - ON UPDATE: *CASCADE*
      - I used cascade because the receipt should stay up to date with the transaction table to make sure the transactions are up to date with latest printed receipt.
    - ON DELETE: *SET NULL*
      - I used *SET NULL* due to the fact that deleting the receipt should cause the whole transaction to get wiped.
  - Transaction_sender_User_FK & Transaction_recipient_User_FK:
    - ON UPDATE: *CASCADE*
      - I used cascade because the receipt should stay up to date with the transaction table to make sure the transactions are up to date with latest printed receipt.
    - ON DELETE: *SET NULL*
      - I chose to *set null* the sender and recipient because only deleting one of them would cause the transaction to be completely deleted which would be bad user experience, the transaction should only be deleted if both the sender and recipient don't exist.
  - Transaction_Phone_app_FK & Transaction_ATM_FK:
    - ON UPDATE: *RESTRICT*
      - The initiator of the transaction should not be modified, it would make sense to keep track and not change the transaction initiator.
    - ON DELETE: *SET NULL*
      - Deleting the payment method that the user used to pay for the transaction should also not cause the transaction to be deleted.
  - Transaction_Bank_account_FK:

- ON UPDATE: *RESTRICT*
  - The initiator of the transaction should not be modified, since this is the way that the transaction was paid for and it should be recorded at the time of payment and not be modified after that.
- ON DELETE: *SET NULL*
  - Deleting a user account should not delete transactions because the user still exists on the system.

❖ **ATM:**

➢ ATM_Address_FK:

- ON UPDATE: *CASCADE*
  - updating the address should also update the address on the ATM that is referring to it. The address of an ATM can change.
- ON DELETE: *SET NULL*
  - I used *SET NULL* because it makes more sense to only delete the address from the ATM and not the whole entity. The action of deleting an Address should not prompt deleting the whole ATM entity.

❖ **Payment_type:**

➢ payment_type_Transaction_FK:

- ON UPDATE: *CASCADE*
  - The transaction referred to can be updated, an example if the updatable fields are user friendly attributes like title or description or such which should reflect in the payment_type table as well.
- ON DELETE: *CASCADE*
  - If a transaction is deleted then its corresponding payment type should also be deleted. A payment type without a transaction seems obsolete.

❖ **Cash:**

➢ Cash_Currency_FK:

- ON DELETE: *CASCADE*
  - Deleting the currency should also delete the Cash entity because you cannot dispense Cash without a Currency.

➢ Cash_Payment_type_FK:

- ON UPDATE: *CASCADE*
  - Information on the payment type can be updated and should be reflected in the Cash entity set. It would make sense to update a user-friendly name for the payment type and for it to reflect in the Cash table.
- ON DELETE: *CASCADE*

- If a payment type is deleted the Cash entity corresponding to it would make sense to be deleted as well. You cannot have a Cash entity with you its payment type.

❖ **Credit_card:**
- ➢ Credit_card_Payment_type_FK:
  - ▪ ON UPDATE: *No Action*
    - Once the transaction is paid for it should not be updated because it is supposed to set so that later the history is accurate of the transactions.
  - ▪ ON DELETE: *CASCADE*
    - A deletion from the payment_tyoe should also deleted this corresponding entity since it would not make sense to have a credit card with its parent payment_type.

❖ Credit_card_Bank_card_FK
- ▪ ON UPDATE: *CASCADE*
  - Updating the bank card in your bank account should be an option, it does not need to be stored for history like for a payment made on a transaction or such.
- ▪ ON DELETE: *CASCADE*
  - It would make sense to delete a credit card if the parent entity which is bank card is also deleted.

❖ **Debit_card:**
- ➢ Debit_card_Payment_type_FK
  - ▪ ON UPDATE: *No Action*
    - Once the transaction is paid for it should not be updated because it is supposed to set so that later the history is accurate of the transactions.
  - ▪ ON DELETE: *CASCADE*
    - A deletion from the payment_tyoe should also deleted this corresponding entity since it would not make sense to have a debit card with its parent payment_type.
- ➢ Debit_card_Bank_card_FK
  - ▪ ON UPDATE: *CASCADE*
    - Updating the bank card in your bank account should be an option, it does not need to be stored for history like for a payment made on a transaction or such.
  - ▪ ON DELETE: *CASCADE*
    - It would make sense to delete a credit card if the parent entity which is bank card is also deleted.

- **Bank_card:**
  - Bank_card_Bank_account_FK:
    - ON UPDATE: *CASCADE*
      - It would make sense to cascade an update on a card and not to restrict it, this information need to stay up to date and not suddenly point to an outdated transaction after an update.
    - ON DELETE: *CASCADE*
      - A bank card needs to belong to a bank account and sometimes ATMs need to find a bank account by that card, so if the bank account is deleted then so should be the its card.

- **ATM:**
  - ATM_Address_FK:
    - ON UPDATE: *CASCADE*
      - It would make sense to cascade an update of the address and not to restrict the address from being updated. Sometimes a typo in address would need some updating or such.
    - ON DELETE: *SET NULL*
      - Deleting an address of an ATM should not delete the ATM itself.

- **User_Address**
  - User_Address_User_FK:
    - ON UPDATE: *CASCADE*
      - It would make sense to update the record when updating a user in that record when it is updated.
    - ON DELETE: *CASCADE*
      - It would make sense to delete the record from the middle table because deleting the User which should also delete the "connection" from the User table. Having the User refer to an empty address record would not make sense.
  - User_Address_Address_FK:
    - ON UPDATE: *CASCADE*
      - It would make sense to update the middle record when updating an address in that record when it is updated.
    - ON DELETE: *CASCADE*
      - It would make sense to delete the record from the middle table because deleting the Address which should also delete the "connection" from the Address table. Having the User refer to an empty address record would not make sense.

- **SafeEx_account:**
  - SafeEx_account_User_FK:
    - ON UPDATE: *CASCADE*
      - Updating a user should also update the account that the referring to that user.
    - ON DELETE: *CASCADE*
      - Deleting the User should also delete the account since this account will get orphaned otherwise.

- **Login:**
  - SafeEx_account_has_User_SafeEx_account_FK:
    - ON UPDATE: *CASCADE*
      - The account connected to that login activity should always point a valid account, there is no point of having a record of login activity without an up to date account tied to it.
    - ON DELETE: *CASCADE*
      - The login records are only used to show the user their login activity so there is no point of keeping the record of login if the account does not exist.

  - SafeEx_account_has_User_SafeEx_account_FK:
    - ON UPDATE: *CASCADE*
      - The user connected to that login activity should always point a valid account, there is no point of having a record of login activity without an up to date User tied to it, especially since the user is what looks at the login activity record.
    - ON DELETE: *CASCADE*
      - The login records are only used to show the user their login activity so there is no point of keeping the record of login if the user does not exist.

- **Notification:**
  - SafeEx_account_has_User_SafeEx_account_FK:
    - ON UPDATE: *CASCADE*
      - The account connected to that login activity should always point a valid account, there is no point of having a record of login activity without an up to date account tied to it.
    - ON DELETE: *CASCADE*
      - The login records are only used to show the user their login activity so there is no point of keeping the record of login if the account does not exist.

- SafeEx_account_has_User_User_FK:
  - ON UPDATE: *CASCADE*
    - The user connected to that login activity should always point a valid account, there is no point of having a record of login activity without an up to date User tied to it, especially since the user is what looks at the login activity record.
  - ON DELETE: *CASCADE*
    - The login records are only used to show the user their login activity so there is no point of keeping the record of login if the user does not exist.

❖ **Phone_number:**

- Phone_number_User_FK:
  - ON UPDATE: *CASCADE*
    - The user needs to stay up to date in the phone number entity since the user can be updated and the phone should still tied to the user.
  - ON DELETE: *CASCADE*
    - Deleting the user should also delete their personal info such as the phone number, it would be a waste of memory to have a phone number without its users.

❖ **Bank_account:**

- Bank_account_User_FK:
  - ON UPDATE: *CASCADE*
    - A user's bank account should be updated when the user is updated since this bank account is not tied to a transaction that should be accurate in a historical perspective.
  - ON DELETE: *CASCADE*
    - Deleting a User should also delete their bank account since it does not make sense to have a bank account without having a user to own that bank account.

❖ **Police_report:**

- Police_report_User_FK:
  - ON UPDATE: *CASCADE*
    - A user update should cause the police reports to not get updated; the user is free to keep their reports throughout the lifetime of the account.
  - ON DELETE: *CASCADE*
    - A police report is made for a user's record and convenience, it would not make sense to keep it if the user is deleted.

- Police_report_Incident_FK:
  - ON UPDATE: *CASCADE*
    - An update on an incident description should also be up to date and not unpredictably get deleted.
  - ON DELETE: RESTRICT
    - An incident should not get deleted if its parent police report still has not been deleted.

- **Security_footage:**
  - Security_footage_Transaction_FK:
    - ON UPDATE: *CASCADE*
      - The footage should stay up to date with the transaction that it was pertaining to.
    - ON DELETE: *CASCADE*
      - Deleting a transaction should also delete the data pertaining to it because it does not make sense to have some footage to a transaction that does not exist.
- **Recordings:**
  - Recordings_Security_footage_FK:
    - ON UPDATE: *CASCADE*
      - The Security footage should stay up to date with the different camera angles such that if there is a better description of the angle, it updates in the security footage it would reflect on all records.
    - ON DELETE: *CASCADE*
      - Deleting a piece of footage should delete the relation between the camera angle and the footage such that the relation should be disconnected and not the footage or angle themselves deleted.
  - Recordings_Camera_angle_FK:
    - ON UPDATE: *CASCADE*
      - The Security footage should stay up to date with the different camera angles such that if there is a better description of the angle, it updates in the security footage it would reflect on all records.
    - ON DELETE: *CASCADE*
      - Deleting a piece of footage should delete the relation between the camera angle and the footage such that the relation should be disconnected and not the footage or angle themselves deleted.

Section XI: EER Testing Table

Entity	SQLQuery	OK/Failed	Error Description	Possible Solution
Bank_account	Delete	OK	None	None
Bank_account	Update	OK	None	None
Bank_card	Delete	OK	None	None
Bank_card	Update	OK	None	None
User	Delete	OK	None	None
User	Update	Failed	Cannot delete or update a parent row: a foreign key constraint fails	Change ON UPDATE.
Phone_number	Delete	OK	None	None
Phone_number	Update	OK	None	None
Incident	Delete	OK	None	None
Incident	Update	OK	None	None
Police_report	Delete	OK	None	None
Police_report	Update	OK	None	None
Address	Delete	OK	None	None
Address	Update	OK	None	None

Entity	SQLQuery	OK/Failed	Error Description	Possible Solution
ATM	Delete	OK	None	None
ATM	Update	OK	None	None
Phone_app	Delete	OK	None	None
Phone_app	Update	OK	None	None
Transaction	Delete	Fail	Cannot delete or update a parent row: a foreign key constraint fails	Change the ON DELETE attribute.
Transaction	Update	OK	None	None
Currency	Delete	OK	None	None
Currency	Update	Fail	Cannot delete or update a parent row: a foreign key constraint fails	Change the ON UPDATE attribute.
Notification	Delete	OK	None	None
Notification	Update	OK	None	None
Security_footage	Delete	OK	None	None
Security_footage	Update	OK	None	None
Camera_angle	Delete	OK	None	None
Camera_angle	Update	OK	None	None
Cash	Delete	OK	None	None

Entity	SQLQuery	OK/Failed	Error Description	Possible Solution
Cash	Update	OK	None	None
SafeEx_account	Delete	OK	None	None
SafeEx_account	Update	OK	None	None
Login	Delete	OK	None	None
Login	Update	OK	None	None
Credit_card	Delete	OK	None	None
Credit_card	Update	OK	None	None
Debit_card	Delete	OK	None	None
Debit_card	Update	OK	None	None
Recordings	Delete	OK	None	None
Recordings	Update	OK	None	None