Use cases

This section describes a detailed description of the processes provided in the solution and the appearance of the application from the end user point of view.

Wallet Server MDC API

This section describes use cases which can be initiated using Wallet Server MDC API. This API should be used by Customers through integrated Money Transfer Wallet SDK to manage Users and cards data on Wallet Server.

User with Card registration

User with Card Registration is process when user and cards are transferred to Wallet Server to make possible use them in different processes (e.g. money transfer) later in the application. Registration can be done when user starts using Verestro Money Transfer Application. Basically User have to provide all necessary personal and card data to start using application. This data goes through the Mobile SDK to MDC API which will return session token. MDC API is also responsible for the subsequent user authentication.
It is required to perform card strong verification. The 3ds card verification will be started by application after adding card.

This diagram shows high level User registration process.

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold

```
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Mobile Data Core" as mdc
participant "Verestro Data Core" as dc
note right of user: User initiates registration
user->mob: 1. Registration
mob->sdk: 2. Provides User's data
sdk->mdc: 3. Provide User's data
mdc->mdc: 4. Validates request
user<-mdc: 5. Sends SMS OTP / Activation link
user->mob: 6. Activates account
mob->sdk
sdk->mdc: 7. Provides User's data and pair device with User
mdc->dc: 8. Store user
mdc<-dc: 9. User added to Wallet collection
mob<-mdc: 10. Registration success
user<-mob: 11. Your account has been successfully registered
user->mob: 12. Login
mob->sdk: 13. Provides login data
sdk->mdc: 14. Check if provided login data are valid
sdk<-mdc: 15. Success - returns session token
mob<-sdk: 16. Login success
user<-mob: 17. Login success
@enduml
```

This diagram shows high level card registration process.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
```

ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Data Core" as dc
participant "Bank" as bank
note right of user: User initiates card registration and provides necessary data (first and last name, card no, cvc, exp date, password)
user->mob: 1. Provides card data
mob->sdk: 2. Provides obtained card data
sdk->dc: 3. Provide obtained card data
dc->dc: 4. Add card to Verestro Wallet
sdk<-dc: 5. Card has been added to Verestro collection
mob<-sdk: 6. Success
user<-mob: 7. Card added - perform strong verification
note right of user: User starts card verification process
mob->sdk: 8. Verify card
sdk->dc: 9. Verify card
dc->bank: 10. Check if provided card data are valid and perform 3ds
dc<-bank: 11. Success
dc->dc: 12. Changes card verification level
sdk<-dc: 13. Card strong verified
mob<-sdk: 14. Card strong verified
user<-mob: 15. Your card has been successfully verified
@enduml

Wallet Server Money Transfer API

This section describes use cases which can be initiated using Wallet Server P2P API. This API should be used by Customers through integrated Money Transfer Wallet SDK to manage Transactions and Transaction’s Senders/Receivers, Commission calculation and determine Currencies on Wallet Server. Every method below is secured by session token.
or more information about session token, please see "User with Card registration".

Receiver management

In order to make any transaction, it is necessary to determine who the funds will be sent to. This section presents the method of collecting the Receiver’s data so that the Sender can order a

transfer of funds. To illustrate this process, two sequence diagrams were made as there are two Receiver types in the Money Transfer Hub solution – Internal Receiver and External Receiver. *For more information about Receiver types, please see "Terminology".*

This diagram shows high level External Receiver management.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Money Transfer API" as p2p
participant "Verestro Data Core" as dc
note right of user: User initiates money transfer
user->mob: 1. User opens contacts list
mob-->sdk
sdk->p2p: 2. Which contacts exist in Verestro Wallet?
p2p->dc: 3. Which contacts exist in Verestro Wallet?
note left of dc: Check by phone number provided by Wallet SDK
p2p<-dc: 4. Provides existing Receivers IDs
sdk<-p2p: 5. nProvides existing Receivers IDs
mob<-sdk: 6. Returns information which contact exists in Verestro Wallet
note right of user: Receiver existing in Verestro Wallet will be highlighted
user->mob: 7. Chooses Receiver existing in Verestro Wallet from contacts
note right of user: In this case all Receiver's necessary data are provided by Verestro Server
user->mob: 8. Confirms chosen Receiver
@enduml
```

In this process all Receiver’s data are stored in Verestro Wallet Server as Receiver is also Verestro Wallet user.

This diagram shows high level External Receiver management.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Money Transfer API" as p2p
participant "Verestro Data Core" as dc
note right of user: User initiates money transfer
user->mob: 1. User opens contacts list
mob-->sdk
sdk->p2p: 2. Which contacts exist in Verestro Wallet?
p2p->dc: 3. Which contacts exist in Verestro Wallet?
note left of dc: Check by phone number
p2p<-dc: 4. Provides existing Receivers IDs
sdk<-p2p: 5. Provides existing Receivers IDs
mob<-sdk: 6. Returns information which contact exists in Verestro Wallet
note right of user: Receiver existing in Verestro Wallet will be highlighted
user->mob: 7. Chooses Receiver which doesn't exist in Verestro Wallet from contacts
note right of user: In this case Sender is responsible for provide all data which application requires to perform transfer
user->mob: 8. Confirms provided necessary Receiver's data
@enduml
```

In this process all Receiver’s necessary data should be provided by Sender. Provided data will be validated by Verestro to confirm they are correct – for example is provided card number is compatible with Luhn alghorithm.

Determine currency and calculate commission

Before making a transfer, Sender needs to know what currencies the card supports. This is the first step in the Money Transfer card to card transaction process. Information about the currencies supported by the card is provided by Acquirer by contacting the card Issuer.

The next and equally important step is to calculate the commission for the transfer and possibly currency conversion. The currency conversion fee and information about the current exchange rate are provided by Acquirer.
Acquirer receives a file with current rates from Mastercard or VISA every day and keeps it on his side.

This diagram shows high level determine currency and calculate commission processes.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Money Transfer API" as p2p
participant "Acquirer" as acq
note left of mob: User has chosen Receiver
user->mob: 1. Chooses Receiver
```

```
mob->sdk: 2. Provides chosen Receiver data
sdk->p2p: 3. Determines available currencies
note left of sdk: All data was provided during the Receiver selection process
p2p->acq: 4. Gets available currencies from Acquirer
p2p<-acq: 5. Provides available currencies
sdk<-p2p: 6. Provides available currencies
mob<-sdk: 7. Returns obtained currencies
user<-mob: 8. Shows available currencies
note right of user: User sees chosen Receiver and available currencies
user->mob: 9. Provides amount to be transferred and chooses currency
mob->sdk: 10. Provides chosen amount and currency
sdk->p2p: 11. Calculate commission
p2p->acq: 12. Get actual currency rates if chosen currencies are different
note left of acq: Acquirer has actual currency rates which is provided to him by MC or VISA
p2p<-acq: 13. Returns requested actual currency rates
sdk<-p2p: 14. Provides actual currency rates
mob<-sdk: 15. Provides actual currency rates
user<-mob: 16. Shows actual currency rates
user->mob: 17. Confirms Money Transfer
@enduml
```

In this process, the Sender sees what additional charge will be made after the transfer is processed and then decides whether the transfer of funds is to be made.

3DS Authentication

The Money Transfer Hub Solution supports the 3DS 2.0 process and it is required when user is initiating a transaction. This is an authentication method based on the alleged cardholder data check, biometric authentication and improved customer experience.

As mentioned in the "Configuration" paragraph, a specific 3DS integration is required depending on which ACQ Verestro will connect to when making a transaction.

- If the integration with a given ACQ has already been made, Verestro only need to configure the appropriate merchant identifier, which should be provided by the Customer.

If the integration with the required billing agent has not yet been completed, it will be one of the activities for which Verestro will be responsible. Note that integration with 3DS increases the scope of required development.
In card to card transfer, the authentication process is required. It is to confirm that the user is definitely the owner of the card.

This diagram shows high level 3ds authentication processes.

```
@startuml
skinparam ParticipantPadding 30
```

```
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Money Transfer API" as p2p
participant "Acquirer" as acq
participant "Mastercard/VISA" as mcvisa
participant "ACS/Issuer" as acs
note left of mob: User has confirmed currencies and accepted shown currency rate
user->mob: 1. Enters CVC code
note left of mob: In this step the application begins 3DS authentication process.
note left of mob: If the bank decides that 3ds is not required, not any action will be performed.
note left of mob: Diagram shows the option requiring the user to authenticate.
mob->sdk: 2. Initialize 3DS process
sdk->p2p: 3. Initialize 3DS process
p2p->acq: 4. Initialize 3DS process
note left of acs: Bank returns decision whether it is necessary for the user to authenticate
acq->mcvisa: 5. Is 3DS authentication required?
acq<-mcvisa: 6. ThreeDS Method
p2p<-acq: 7. ThreeDS Method
p2p->acq: 8. Continue 3DS process
acq->acs: 9. Continue 3DS process
acq<-acs: 10. Challenge required
p2p<-acq: 11. Challenge required
sdk<-p2p: 12. Challenge required
mob<-sdk: 13. Challenge required
user<-mob: 14. Informs user that challenge is required
note left of mob: Bank's page content is provided
user->user: 15. Performs challenge
```

note right of user: After a successful challenge, the Bank sends PaRes/cRes, which is intercepted by the Wallet SDK and provided to the Money Transfer API
user-->mob
mob-->sdk
sdk->p2p: 16. Provides intercepted PaRes/cRes
p2p->acq: 17. Provides obtained PaRes/cRes
acq->acs: 18. Check authentication for provided PaRes/cRes
acq<-acs: 19. Authentication success
p2p<-acq: 20. Authentication success
sdk<-p2p: 21. Authentication success
mob<-sdk: 22. Authentication success
user<-mob: 23. Informs about the successful completion of the authentication process
@enduml

To facilitate understanding of the process flow, each of the steps is described below in the correct order *(points highlighted in blue are performed outside the Verestro Server)*:

1. Mobile application contacts the Verestro Server via the Verestro Mobile SDK to start 3ds process.
2. Mobile SDK provides all necessary data to the Money Transfer API (including user/card id and 3ds authentication request id) while calling the 3ds initialization method.
3. Having all the necessary information, Money Transfer API orders Acquirer to start the 3ds authentication process.
4. Acquirer transfers the card and user details to ACS.
5. If the user is the owner of the card, the ACS returns a positive answer and a decision whether the continuation of the authentication process is required (according to the diagram above, the scenario with the necessity to continue the process is described).
6. ACS informs the Acquirer that the 3ds process continue is required.
7. Acquirer informs Money Transfer API about ACS decision.
8. Money Transfer API requests Acquirer to continue the authentication process (the authentication id is provided.
9. Acquirer provide the request to continue the process to the ACS.
10. ACS informs about the necessity to perform the Challenge process and returns necessary parameters such as:
    1. challengeHtmlFormBase64 - this field is a BASE64 encrypted html source file containing the ACS’ challenge 3DSecure frame
    2. cReq - data for building a form such as challengeHtmlFormBase64
11. Acquirer informs Verestro Money Transfer API that ACS requires Challengeand provides above parameters.
12. Verestro Money Transfer API forwards the obtained information to Verestro Mobile SDK.
13. Verestro Mobile SDK decodes the received challengeHtmlFormBase64 parameter and transmits the received frame of the mobile application.
14. The user is redirected to the bank's website where he performs the Challenge process.
15. After a successful Challenge process, the bank sends the cRes / PaRes parameter. This response is intercepted by the Verestro Mobile SDK and forwarded to the Verestro Money

Transfer API.
16. Verestro Money Transfer API provides the received cRes / PaRes to Acquirer.
17. Acquirer provides the above parameters to ACS for verification.
18. If everything was done correctly, the ACS informs Acquirer about the successful completion of the 3ds authentication. Among other things, the following parameters are included in the response
    1. cavv - property determined by the ACS. The value may be used to provide proof of authentication
    2. eci - property is determined by the ACS. This property contains the two digit Electronic Commerce Indicator (ECI) value, which is to be submitted in a credit card authorization message
19. Acquirer provides information about successful completion of the 3ds authentication among with the above parameters obtained from the ACS to Verestro Money Transfer API.
20. Verestro Money Transfer API provides information about successful completion of the 3ds authentication among with the above parameters obtained from the ACS to Verestro Mobile SDK.
21. Verestro Mobile SDK provides positive response to mobile application. The information is shown to the user.

Transaction process

If the user and his card are present in the Verestro system, such a user can perform transactions using the Money Transfer Hub solution.

Card to card money transfer

This type of transaction allows Sender to make money send transfers from card to card. The transaction is secured by the 3DSecure process.

This diagram shows high level money transfer card to card processes.

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF

```
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Money Transfer API" as p2p
participant "Verestro Data Core" as mdc
participant "Verestro THC API" as thc
participant "Acquirer" as acq
participant "Issuer" as acs
note left of mob: User has passed 3DS authentication process
mob->sdk: 1. Perform money transfer
sdk->p2p: 2. Perform money transfer
p2p->mdc: 3. Check if provided card belong to the user
p2p->mdc: 4. Check if provided card belong to the receiver
p2p<-mdc: 5. OK
p2p<-mdc: 6. OK
p2p->acq: 7. Money send
acq->acs: 8. Peform Funding from provided card if possible
acq<-acs: 9. Success
acq->acs: 10. Perform Credit for Receiver account
acq<-acs: 11. Success
p2p<-acq: 12. Success + transaction id
note left of p2p: Receiver will be added to "favourite" in Senders contacts if checkbox has been
checked - optional
p2p->thc: 13. Store transaction with status Funding
p2p<-thc: 14. Transaction has been stored successfully
sdk<-p2p: 15. Transaction success
mob<-sdk: 16. Transaction success
user<-mob: 17. Your transaction has been sent
@enduml
```

After ordering the transfer, Wallet Mobile SDK forwards request to the Wallet API. Wallet API in turn forwards the request to the Acquirer. Based on the data received, the Acquirer contacts the bank to complete the transaction. If the bank agrees, the funding process is carried out - collecting funds from the Sender's account. After the funding is completed, the funds are transferred to the appropriate Receiver.

To facilitate understanding of the process flow, each of the steps is described below in the correct order *(points highlighted in blue are performed outside the Verestro Server)*:

1. After going through the 3ds process, the funds transfer process begins,
2. The application orders the transfer of funds by communicating with the Verestro backend via the Verestro Mobile SDK,
3. Money Transfer API communicates with Data Core to check whether the card details belong to the user and the receiver,
4. After receiving a positive response from Data Core, Money Transfer API performs money send transaction,
5. Acquirer receives a request to make a transfer of funds,
6. The Acquirer communicates with the Issuer regarding the execution of the transfer,
7. The Sender's account is debited,
8. The Receiver’s account is credited,
9. The Acquirer receives information from the Issuer that the operation has been performed,
10. Acquirer informs Money Transfer API about the positive status of the ordered operation,
11. Money Transfer API saves transaction details in the Transaction History Core API,
12. Money Transfer API informs Mobile SDK about the positive status of the ordered operation,
13. Mobile SDK informs Mobile Application about the positive status of the ordered operation,
14. The Mobile Application displays to the user a successful transfer of funds.

> Verestro Money Transfer Hub also supports transfers using a QR code.
> Transfers using the QR code are described in a separate document.

Application flow

This chapter introduces the main actions and processes in the application from the end user point of view.

User registration flow

Invoking registration is usually the primary activity right after the first launch of the application. User registration, depending on the configuration, requires providing data, including authentication, which are at a later stage his login to the application. Required data are e-mail address, telephone number, accepting tos, assigning a password to the account and assigning a pin to the application.

The pictures below show the first step of the registration process:

image-1650446717200.png	image-1650446725423.png	image-1650446735943.png

On the screen above user registers in to application by clicking “Sign up” button.	After clicking "Sign up" button, the user is redirected to the screen with the fields that must be filled in to register in the application. Fields which are required: • E-mail, • Phone number, • Password, • Repeat password, • Accept Terms & Conditions.	After providing all the necessary data, the user can go to the next step of registration by clicking the "Next" button. The “Next” button remains inactive until all required fields are correctly filled in.

image-1650446991254.png
In the screen beside the pop-up with regulations has been shown. This pop-up is displayed to the user after checking the "Accept Terms & Conditions" checkbox. The user can accept the regulations by clicking the "Accept" button or reject it by clicking the "Discard" button. Acceptance of the regulations is required to complete the registration process, and thus to use the application.

The pictures below show the “Set up PIN” view:

image-1650447077291.png	image-1650447083737.png

On the screen above user sets PIN of his account. To do such the user need to provide four digits and repeat them - this prevents from making a mistake. To save new PIN the user must confirm changes by clicking Confirm button. Button remains inactive unless
user provides new PIN and repeat it correctly. User is able to exit this section by clicking “<” button in the upper left corner.

User login flow

The first login of the user takes place using the login and password. The login, depending on the configuration, may be a telephone number or an e-mail address. When logging in, the user may agree to log in to the application using biometrics, as long as his device supports such a solution. Additionally, it is possible to reset the password in case the user forgets it.

The pictures below show the login process view with “use fingerprint” checkbox:

image-1650448497416.png	image-1650448506875.png	image-1650448580854.png
On the screen above user logs in to application. Sign in button remains inactive unless user provides his phone number and password. Phone number and password must be valid. Password can be reseted. User may also login by fingerprint by checking “Use fingerprint” checkbox.	On the screen above user tries to reset his password by providing his email. Email must be valid.	Until the user enters the email, the "SEND" button remains inactive. The authorization code will be sent to provided email.

The next login takes place in accordance with the settings selected by the user.

The pictures below show other possibilities of the login:

image-1650453306410.png	image-1650453312442.png	image-1650453318217.png
On the screen above user logs in to application using his fingerprint. To do this, the user has to tap his finger on the screen. It is also possible to use a PIN by clicking “Use PIN” button in the down left corner of the screen.	On the screen above user logs in to application by PIN. PIN must be valid. User may select more login options by clicking “MORE OPTIONS” button.	On the screen above user chooses how he wants to log in.

Selecting the "Reset PIN" option will redirect the user to the following subpage:

image-1650453419929.png
In the screen beside user is able to reset PIN. To perform this action the user must authorize himself with his standard password. The option of logging in with a standard password and resetting the password was presented earlier in the chapter "User login flow". Possible errors: • Invalid fingerprint, • Invalid PIN, • Too many failed attempts, user PIN, • Too many failed attempts, user login and password.

Add card flow

The application, in cooperation with Verestro backend compliant with PCI DSS, provides the possibility to perform operations on cards belonging to its owner. Adding a card from the outside can be done by scanning its data with a camera built into the device or by manually entering its data into the form. Adding such a card requires its verification by calling 3ds and is successfully completed after the card issuer is authorized to use the card.

The pictures below show the possibility of the add card by scanning its data:

image-1650453901514.png	image-1650453909876.png

On the screen above user adds his card by scanning its data using device camera. The user can also add the card manually by selecting the option "enter card data manually".

The pictures below show the possibility of the add card by providing its details manually:

image-1650454118693.png	image-1650454126225.png

On the screen above user provides card data and chooses whether this card will be the default one or not. After providing all the necessary data, the user can confirm adding a card with the "Confirm" button. The “Confirm” button remains inactive until all required fields are correctly filled

in.

After adding the card, the application will require its verification with the 3ds standard. To authenticate, an SMS will be sent with a code to the phone number assigned to the user’s account. This code should be entered in the designated place.

The pictures below show the 3ds process that the user must go through in order to be able to use the added card in the application:

image-1650454217524.png	image-1650454225732.png

Add address

The address module in the application can be used as a private, highly secured section for the user's needs. In addition, it acts as a storage of the user's address, which, due to the legal requirements regarding AML, is used during card transactions from the application level. The data to be provided in the add address section are listed below:

- First name,
- Last name,
- Company name,
- Street,
- House number,
- Local number,
- Postal code,
- City,
- Phone number,
- NIP,
- Country,
- Checkbox - default address.

The pictures below show “adding address” view:

image-1650456748251.png	image-1650456754093.png

On the screen above user provides his address details. After providing all the necessary data, the user can confirm adding an address with the "Confirm" button. The “Confirm” button remains inactive until all required fields are correctly filled in.

image-1650456795698.png

On the screen beside user is able to whether update his address or delete it. In case of update each field can be changed. User can confirm his changes by clicking “Confirm” button. This button remains inactive until all required fields are correctly filled in. By checking “My address” checkbox, the user confirms that this is his default address. User can delete his address by clicking “waste-

basket” button.

Receiver management flow

Receivers is a module that streamlines and automates getting the recipient object to generate a transaction via Money Transfer Hub. The usefulness of this module takes place in 2 key functions:

- Active users – feature which allow to provide the user with information which contacts from his personal contact list also use this application. Thanks to this solution, users maintained within one infrastructure can transfer funds between themselves "to the telephone number". Receivers which also use the application are marked with the “V” logo,
- Last used / favorite receivers – feature which allow to store a personal, secure contact book with cards. Communication between the application and the server after a single transmission of the encrypted card number takes place later on the basis of unique identifiers, where the card data itself on the servers are stored and used in accordance with the PCI DSS standard, ensuring the security of all data entrusted to it.

The pictures below show the user’s contact list:

image-1650456966776.png

In the screen beside user is able to choose transfer receiver from his contact list. Receiver which uses Money Transfer Application is marked with “V” logo. The user has the option of filtering potential receivers by entering the receiver’s data (name, surname) or by marking the "V" filter which means that only active users will be displayed in the contact list. There is also an option to add new
receiver but clicking “+” button.

The pictures below show the user’s contact list with active filters:

image-1650457156844.png	image-1650457164176.png
On the screen above the user’s contact list with active “Money Transfer Application users” filtering has been shown.	On the screen above the user’s contact list with active “Your recipients” filtering has been shown.

Adding a new receiver can be divided into two cases. New and existing receiver. If a receiver already exists in user’s list of recipients, an appropriate message is displayed and there is an option to add another card which will be assigned to him. If receiver does not exist in the user's list of recipients, the user must assign him a card. In both cases the "Confirm" button remains inactive until all required fields are completed.

The pictures below show case of adding receiver:

image-1650457330593.png	image-1650457336627.png	image-1650457341313.png

On the screen above user adds new receiver by providing necessary data – first name, last name and phone number with prefix. User can confirm adding new receiver by clicking the "Confirm" button. The “Confirm” button remains inactive until all required fields are correctly filled in.		On the screen above user is informed that his new receiver has a card assigned to him. The user can choose whether he use this card or add new one.

The pictures below show case of next step of adding receiver (this step occurs only if the user wants to assign new card to his receiver):

image-1650457428569.png	image-1650457435242.png

On the screen above user adds new card to the receiver by providing necessary data – display name and card number. There is also “Save recipient” checkbox which is responsible for save the receiver in “favourite ones” list. User can confirm adding new receivers card by clicking the "Confirm" button. The “Confirm” button remains inactive until all required fields are correctly filled in.

Transaction flow

The function of this module is to generate a payment transaction from data entered manually by the user or supplied from other modules such as QR or Friends. Transaction are secured by 3ds protocol.

Card to card transaction

This module allows you to make a transfer from card to card. The user selects the card which will be debited and the recipient's card which will be topped up.

The pictures below show “money transfer” view:

image-1650540852315.png	image-1650540859358.png	image-1650540866062.png
On the screen above user chooses which of his cards will be debited and then confirm his choice. Chosen receiver is shown on the screen.	On the screen above user chooses currency and the amount of the transfer. The transfer button remains inactive until all required fields are correctly filled in.	On the screen above user filled in all required fields and is able to perform the transfer. In addition the information about debit amount is shown.

image-1650541058468.png	image-1650541065810.png	image-1650541072109.png
On the screen above user is able confirm the transfer using his fingerprint or PIN.	On the screens above the 3ds authentication is required to perform money transfer. Bank sends the authentication code via SMS.	

image-1650541140975.png	image-1650541195027.png

On the screen above user is informed about the successfully money transfer. Return button will redirect the user to the main page.	On the screen above user is informed about the unsuccessfully money transfer. In this case user is able to try again the transfer or return to the main page. Example reasons of the transaction fail: • Luck of funds, • 3ds failed, • Invalid card data.

> *To see the appearance of the QR transaction component from the end user level please visit QR Payments Application flow*

Transaction history flow

In the Payment History section, the user can check the history of his transactions in the application as well as make statements of charges and credits from given time periods.

The pictures below show some of “payment history” view:

image-1651756901486.png
The picture above shows the contents of the "Transactions" tab. After entering this tab, the history of all transactions sorted by the newest and divided per day is shown to the user.

image-1651756895522.png

The picture above shows the contents of the "Spends" tab. After entering this tab, a list of all expenses for a given month is shown to the user, broken down by type of expense.

image-1651756930461.png

On the screen above there is transaction history filtering option. The user can narrow down the number of results by selecting different filtering criteria. In the section, there is also possibility to remove all selected filters by clicking the "Remove all filters" option.

image-1651756932836.png

On this screen, the user sees information about which card the transaction was related to with its thumbnail, who is the addressee, transaction category, transaction ID, as well as its status, transaction amount, date and time of the transfer.

Revision #2
Created 21 March 2023 07:06:00
Updated 27 March 2023 08:03:20