SDK Documentation

Revision History

Date	Version	Description	Author
07.07.2023	1.0	Initial version	Ivan Yanakiev (Product Manager, SONECT)
03.08.2023	1.1	Expanded initialisation flow	Ivan Yanakiev (Product Manager, SONECT)
16.10.2023	1.2	Elaborated the API calls	Surya Vedula (Head of Engineering, SONECT)
17.10.2023	1.3	Beautification	Ivan Yanakiev (Product Manager, SONECT)
16.05.2024	1.4	Contacts Update	Ivan Yanakiev (Product Manager, SONECT)

Table of Content

Introduction

The scope of this document is to understand how to integrate the SONECT SDK into another app. This document includes instructions for the following:

- Android
- iOS
- Hybrid applications
- Backend (Server-to-server / Webhooks)

Definitions, Acronyms, and Abbreviations

- **API** - Application Programming Interface
- **HMAC** - Hash-based message authentication code
- **HTTPS** - HyperText Transfer Protocol Secure
- **JWT** - JSON Web Token
- **SDK** - Software Development Kit

Contacts

Product Manager	Ivan Yanakiev	i.yanakiev@sonect.net
Senior Backend Developer	Damian Anchidin	d.anchidin@sonect.net

Integration - WEB SDK

SDK Initiation Overview

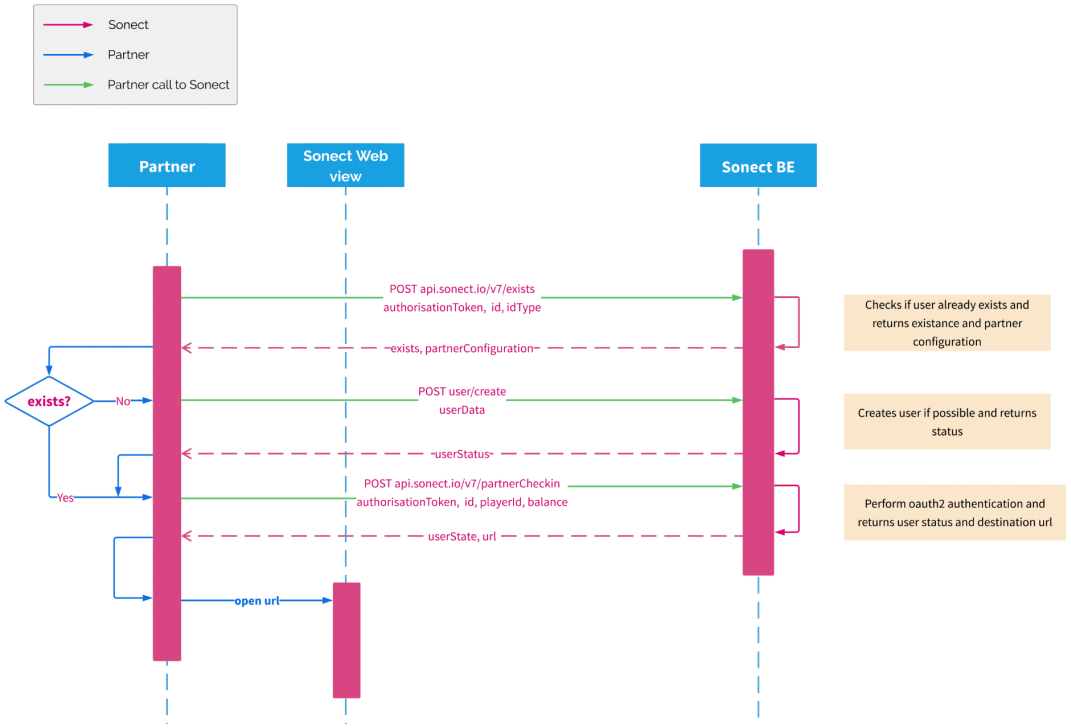


Figure 1: WebSDK initialisation flow

Diagram in **Figure 1** shows the flow of communication between partner application and Sonect to create user if needed and then to receive checkinUrl for opening Sonect's WebSDK. Flow starts by partner initiation **/exists** call to understand if the current user is already registered in Sonect's system. If a user does not exist, the partner needs to call **/user/create** endpoint with sufficient data, in order to create a corresponding user in Sonect's system. After the user exists **/partnerCheckin** endpoint is called to perform authentication and provide a url to open, which includes JWT token for further session requests.

APIs Breakdown

Exists

POST /exists

Checks if the user already exists and returns existence status and partner configuration.

Endpoint destination URL

TEST	https://api-test.sonect.io/v7/exists
PROD	https://api.sonect.io/v7/exists

Header parameters

Authorization: string
Authorization token derived from Sonect provided credentials. Example token creation:
```
const basicToken = 'Basic ' + btoa(clientId:clientSecret);
```

Request parameters

id: string
User identifier in Sonect's system, to be checked for existence.

idType: string
Type of id to be checked, one of three values "id", "phone" or "vat".

userType: string
Type of user to be checked, one of three values "user", "atm" or "worker".

Response parameters

HTTP Responses
200 - Success
Check for user existence has been successful. JSON response will contain **payload** field including following:
result: 0
exists: Bool

400 - Bad Request
Check for request parameters. Response will contain empty **payload** field including following:
result: -1

403 - Forbidden
Action not allowed. Response will contain empty **payload** field including following:
result: -6

500 - Internal Server Error
Server Error. Response will contain empty **payload** field including following:
result: -1

Create user

POST /user

Creates user in Sonect's system with provided details.

Endpoint destination URL

TEST	https://api-test.sonect.io/v7/user
PROD	https://api.sonect.io/v7/user

Header parameters

Authorization: string
Authorization token derived from Sonect provided credentials. Example token creation:
```
const basicToken = 'Basic ' + btoa(clientId:clientSecret);
```

Request parameters

referenceId: string **(required)**
User identifier to be created in Sonect's system.

firstName: string **(required)**
Natural person's first name.

middleName: string
Natural person's middle name.

lastName: string **(required)**
Natural person's last name.

address: object
JSON object with following structure:
{ line1: string,
line2: string,
houseNumber: string,
zip: string,
city: string,
country: string, **(required)**
countryCode: string, **(required)**
formatted: string }

email: string **(required)**
Email of the user.

phone: string **(required)**
Phone of the user.

dateOfBirth: string
Date of birth of the user.

Response parameters

HTTP Responses
200 - Success
User creation has been successful.

400 - Bad Request
Check for request parameters. Response will contain empty **payload** field including following:
result: -1

403 - Forbidden
Action not allowed. Response will contain empty **payload** field including following:

Message: “Error”(or) “Email already exists”
result: -177 (or) -1

500: Internal Server Error
Server Error. Response will contain empty **payload** field including following:
result: -1

Partner check-in

POST /partnerCheckin

Perform oauth2 authentication and return user status and destination url.

Endpoint destination URL

TEST	https://api-test.sonect.io/v7/partnerCheckin
PROD	https://api.sonect.io/v7/partnerCheckin

Header parameters

Authorization: string
Authorization token derived from Sonect provided credentials. Example token creation:
```
const basicToken = 'Basic ' + btoa(clientId:clientSecret);
```

Request parameters

referenceId: string **(required)**
User identifier in Sonect’s system, for which authorization is performed.

additionalData: object
JSON object that includes any partner specific data that should be sent to Sonect’s system. I.e. it could contain user balance, specific ids for further communication and others.

Response parameters

HTTP Responses
200 - Success

JSON response will contain **payload** field including following:
checkinUrl: **string**

400 - Bad Request
User not found. Response will contain empty **payload** field including following:
result: -21

500: Internal Server Error
Server Error. **payload**:
{ checkInUrl: null }

Open checkinUrl `in an inline web browser(e.g. SFSafariViewController for iOS) to finish the process of opening Sonect's WebSDK.`

Integration - Server

Every bank who integrates the SONECT SDK will have the following (unique) credentials received from Sonect during the Integration Partner setup process :

- clientId
- clientSecret
- hmacKey

Payment Process Overview

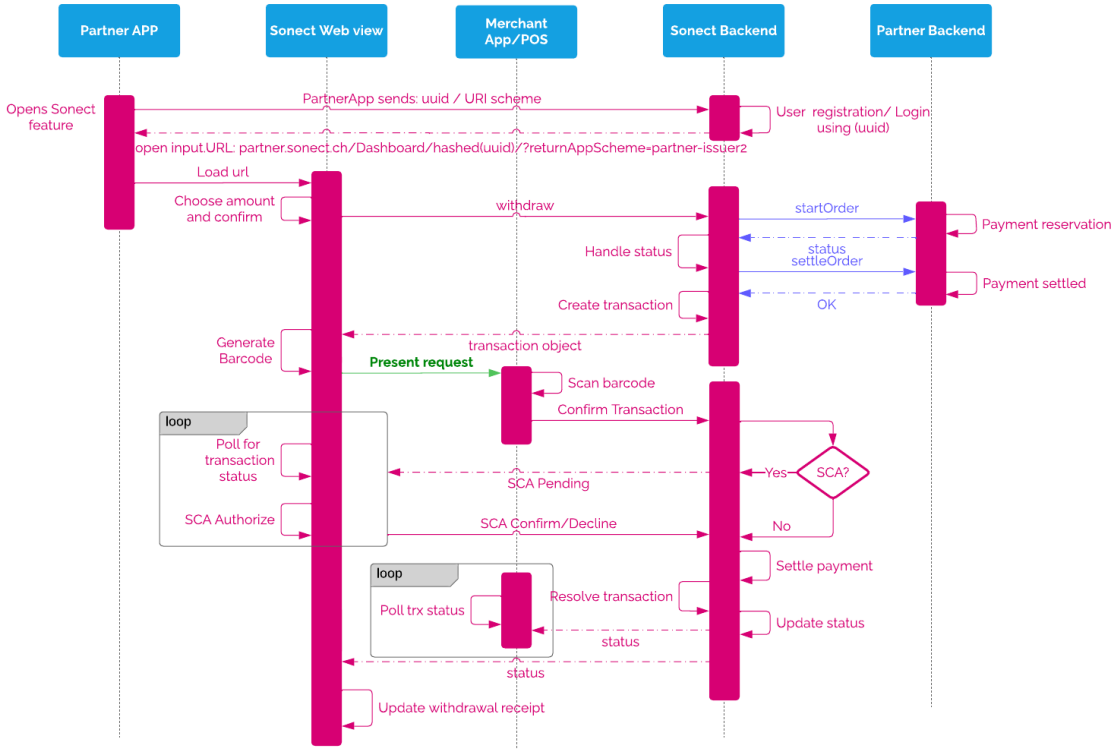


Figure: Transaction withdrawal flow

User cash withdrawal flow consists of following steps, from user/merchant perspective:

1. **Initiation of Withdrawal by User**
   User specifies the amount to withdraw and starts the process by tapping “Confirm” on the withdrawal screen.
2. **Barcode Generation**
   Upon confirming the withdrawal amount, the user is presented with a unique barcode on their device's screen.
3. **Barcode Scanning by Merchant**
   The user visits a participating merchant and presents the barcode to the merchant for scanning.
   The merchant scans the barcode using their Point-of-Sale (POS) system or a dedicated app.
4. **Merchant Confirmation**
   Once scanned, the merchant confirms the amount and transaction details on their system.
5. **Strong Customer Authentication (SCA)**
   After the merchant confirms the transaction, the user receives a prompt for SCA verification.
6. **User Completes SCA Verification**
   The user follows the in-app instructions to complete the SCA verification. This involves entering a previously set passcode that is stored on Sonect's system for verification purposes..

7. **Transaction Completion**
   After successful SCA verification, the transaction is marked as complete in the system. Both user and merchant apps will have the updated transaction state.
8. **Cash Handover by Merchant**
   The merchant receives confirmation of the transaction completion and hands over the specified cash amount to the user.
9. **Transaction Receipt**
   Both the merchant and the user receive digital receipts for the completed transaction.

During this process there are 2 touch points between the Sonect's system and integrating partner, and both happen during transaction initiation:

1. **StartOrder**
   Sonect's backend will call startOrder API to ask the integrator to book payment and verify the user has enough funds for the transaction.
2. **SettleOrder**
   Before a transaction is created, settleOrder will be called to finalise payment from the integrator to Sonect's user account.

APIs Breakdown

Note: The exact API endpoints are provided by integrating partner, so following documentation is just an example interface and should be synchronised between Sonect and integrating partner.

startOrder

POST /api/sonect/startwithdrawal

Informs the integrating partner that a transaction is initiated for a given amount. If response is positive funds are considered secured, but payment is not yet executed.

Header parameters

X-Auth-Token: string
Authorization token.

Request parameters

userId: integer **(required)**

User id in partner system.

amount: { value: Number } **(required)**
Amount of transaction.

dateOfTransaction: string **(required)**
Date of transaction initiation. 2023-10-05T05:31:04.278Z format.

expiryTimeOfTheOrder: string **(required)**
Date of transaction expiration. 2023-10-05T05:31:04.278Z format.

sonectTransactionReferenceId: string
Transaction reference id in Sonect's system.

store: string

HTTP Responses
200 - Success
User creation has been successful. Response contains:

Message: string

ResultCode: integer
- 4: Cash limit exceeded
- 1: Generic error
- 21: Transaction is already in final state
1222: Insufficient balance

Skip: bool

TransactionId: integer
Transaction id in partner's system to be used for future reference.

500 - Internal Server Error

settleOrder

POST /api/sonect/settlewithdrawal

Confirms transaction is finalised and payment should be made to given Sonect account. If response is positive then transaction is created presented to user. If negative transaction is not created.

Header parameters

X-Auth-Token: string
Authorization token.

Request parameters

TransactionId: integer **(required)**
Id of transaction in partner's system.

UserId: integer **(required)**
User id in partner system.

DateOperation: string **(required)**
Date of transaction initiation. 2023-10-05T05:31:04.278Z format.

SonectTransactionReferenceId: string
Transaction reference id in Sonect's system.

HTTP Responses
200 - Success
User creation has been successful. Response contains:

Message: string

ResultCode: integer
- 4: Cash limit exceeded
- 1: Generic error
- 21: Transaction is already in final state
1222: Insufficient balance

Skip: bool

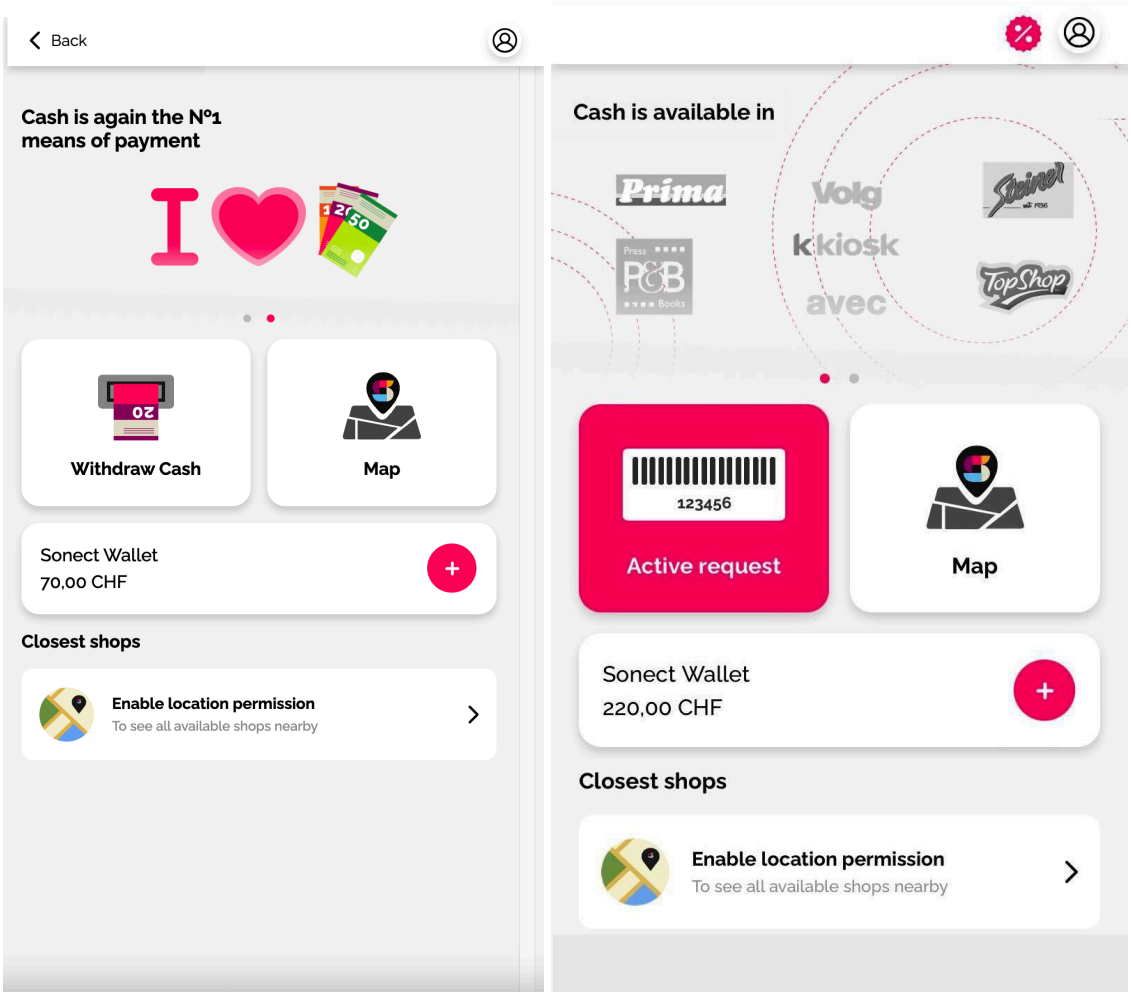
500 - Internal Server Error

SDK Screens

The following SDK screen will give an overview of the logic flow and UI within the SDK. All in all the SONECT SDK will have a dark and a light theme.

Dashboard

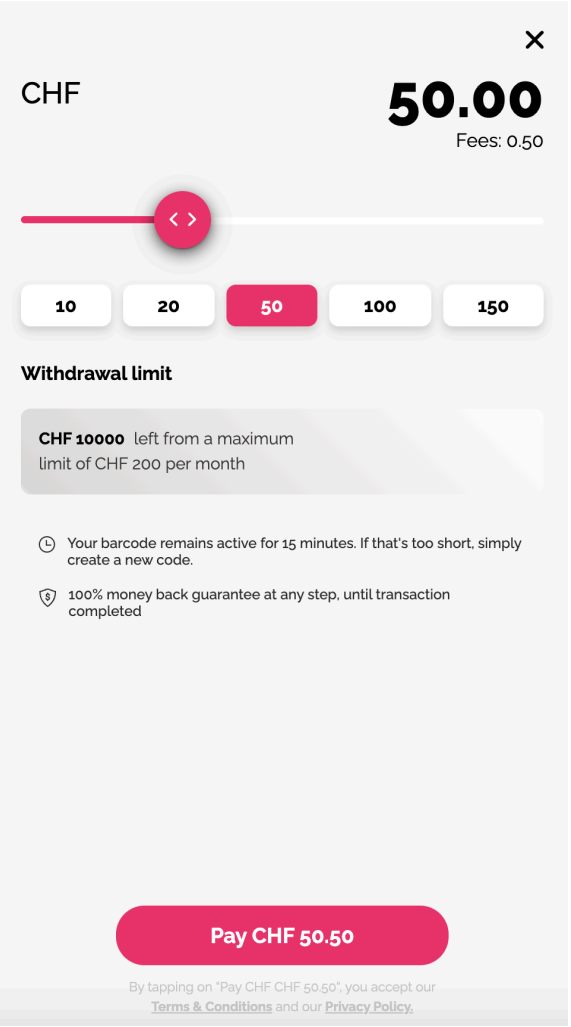
SONECT SDK v1 will come with one main feature for cash withdrawal and supporting features map with SONECT Shops (Map) and profile button in the upper left corner. List of nearby shops is present if user has enabled locations and there are shops within certain distance.

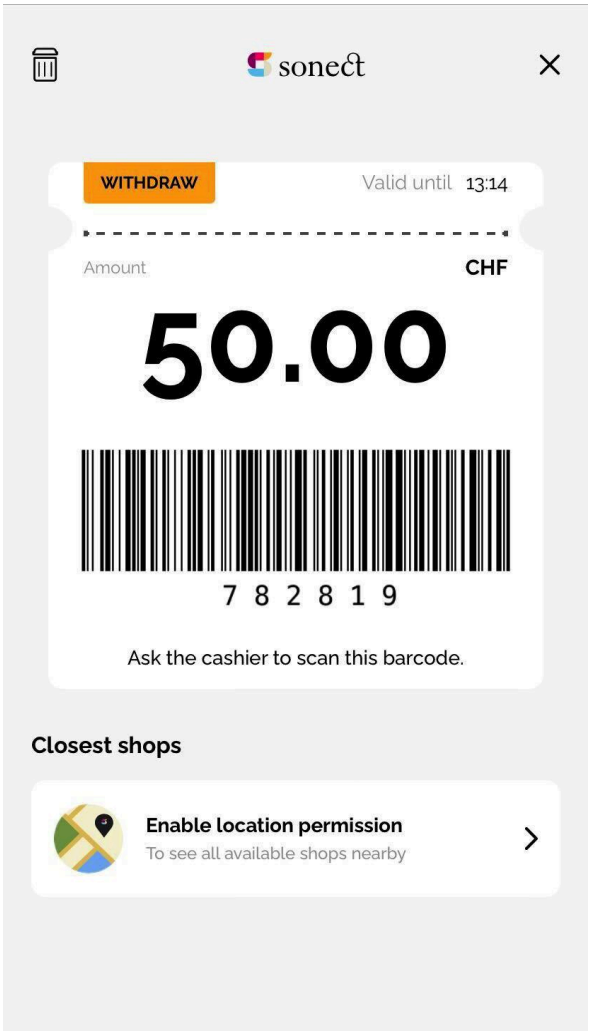


In the active barcode button there will be the open withdrawal request stored. It is only possible to generate one request at a time

Cash withdrawal

Main feature cash withdrawal contains to choose the amount, through quick selection buttons or slider, and a tile with current limit and info disclaimer. By pressing Pay the user initiates the withdrawal request with barcode.

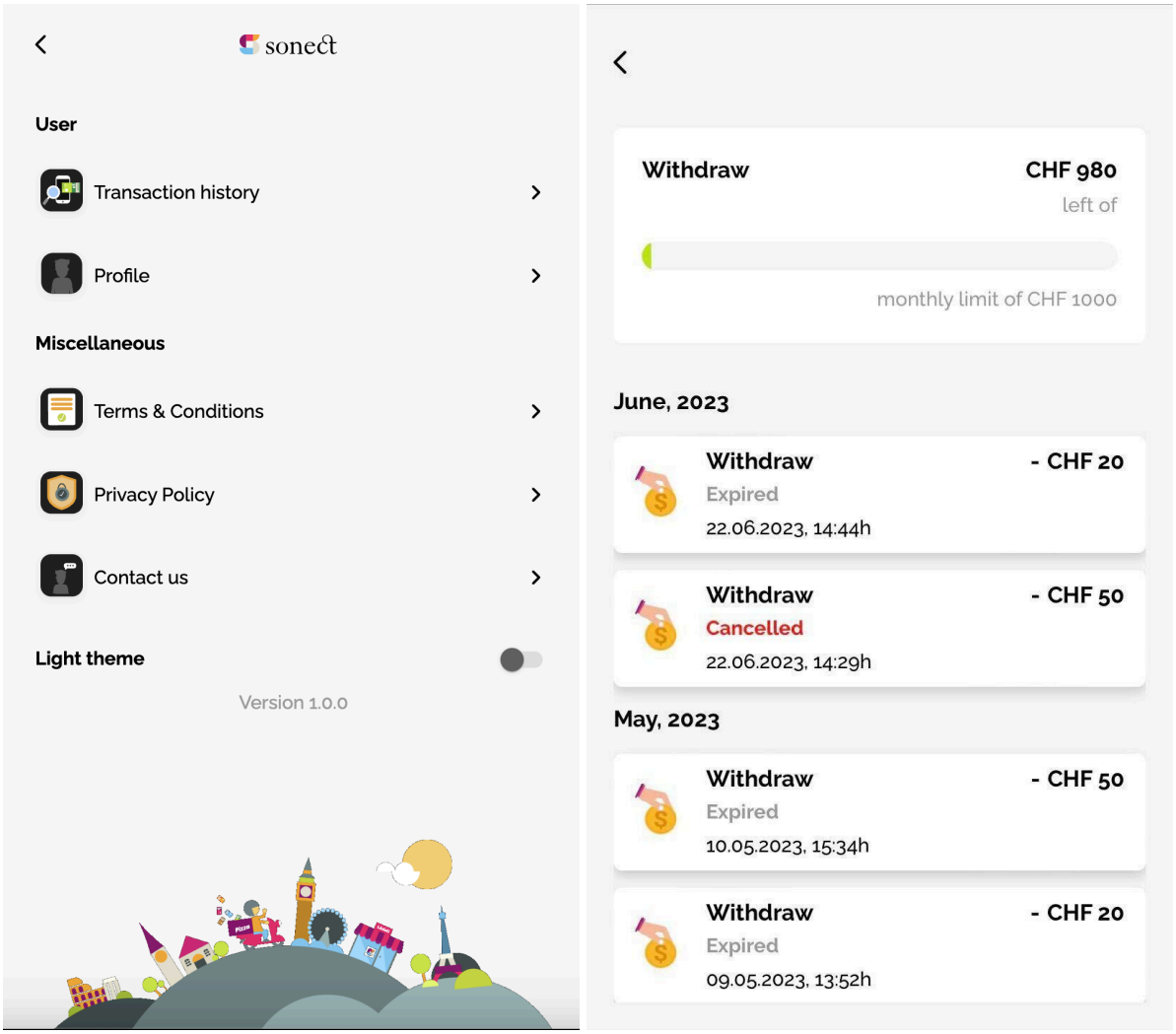




By pressing on trash icon, users can cancel the withdrawal request. User can also browse nearby shops if location permissions are granted.

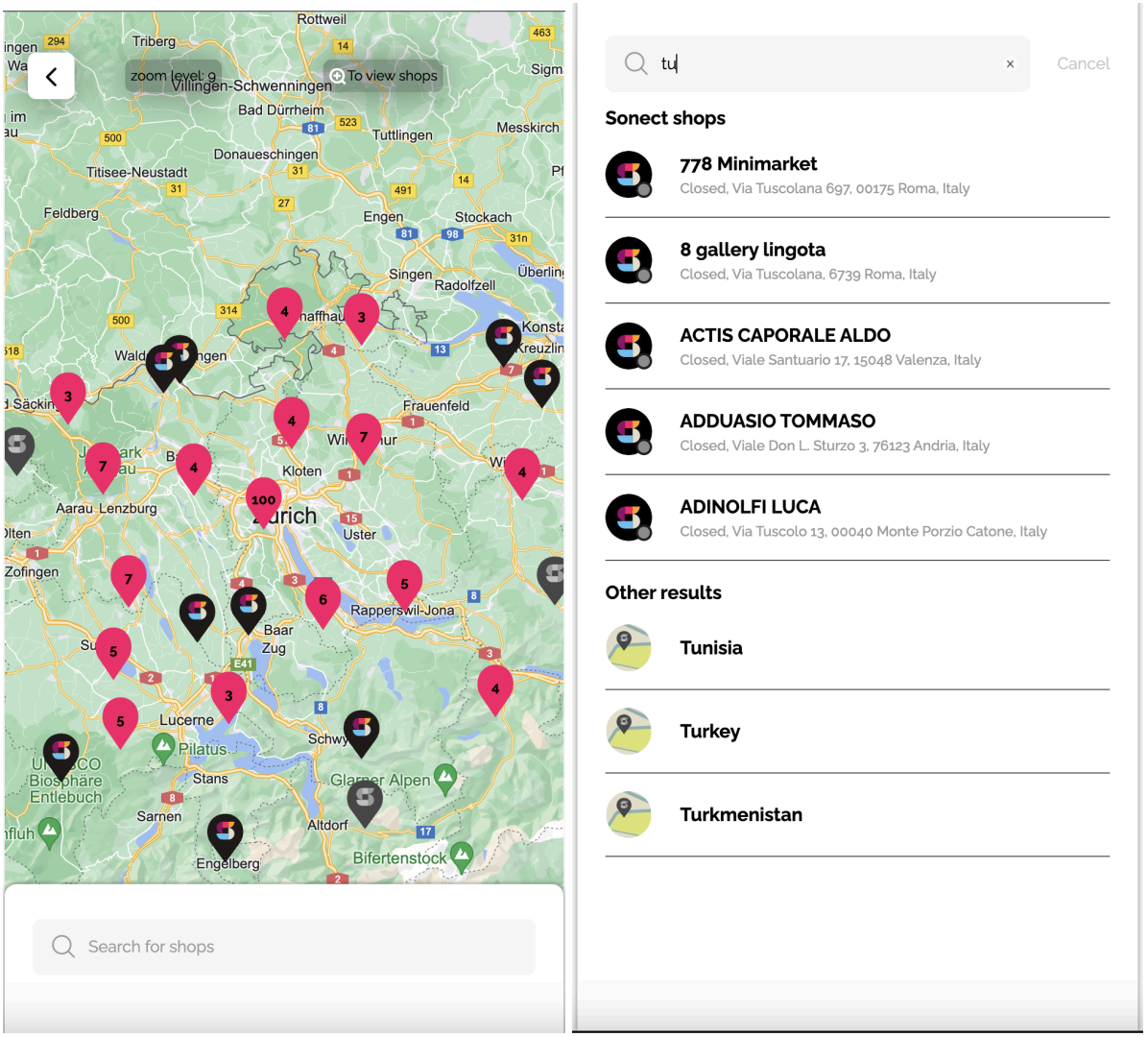
Profile and transaction history

In this supporting feature users will be able to review profile informantion their transactions.



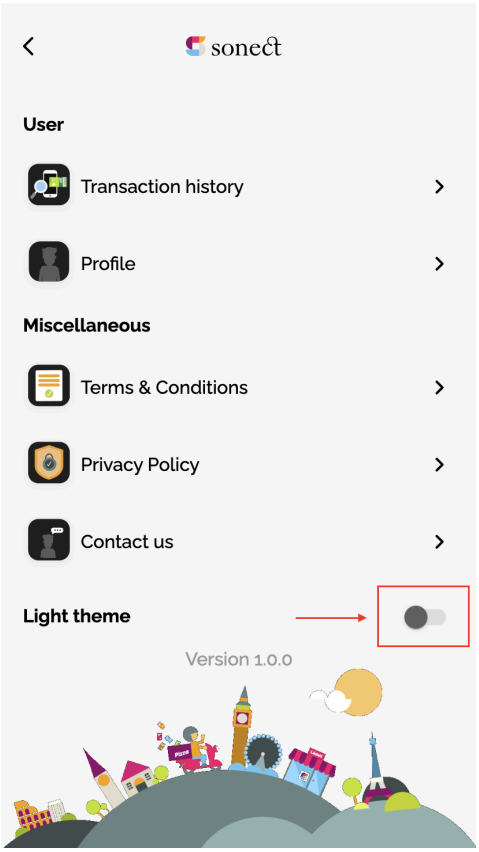
Map

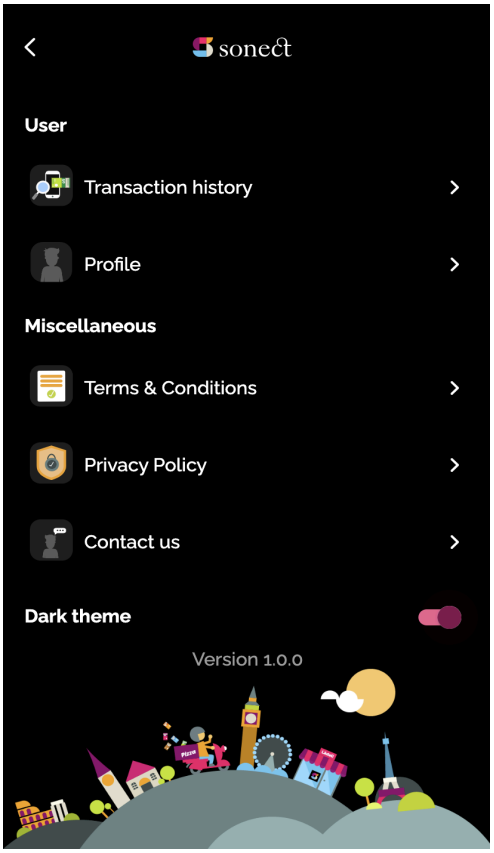
In this supporting feature users can look for a SONECT ATM. Users can scroll in the map, search for a specific ATM or search for ATMs.. This view indicates open and closed shops on the map.



Themes

Web SDK uses the White theme by default.
You can switch themes between White or Dark on the Profile page. (screen below)


sonect
User
Transaction history
Profile
Miscellaneous
Terms & Conditions
Privacy Policy
Contact us
Light theme
Version 1.0.0


sonect
User
Transaction history
Profile
Miscellaneous
Terms & Conditions
Privacy Policy
Contact us
Dark theme
Version 1.0.0