ACCOUNT

Create, Read, Update, Delete Account.

The account entity can be divided into 3 types namely admin, partner, and user. Therefore, when we want to create an account, we need to specify the type of the account such as ‘Account Type’ : ‘USER’ or ‘PARTNER’.

- Endpoint 1 :
  Description : Create an account
  URL : http://localhost:3000/api/account
  Method: POST
  Input: [{
  ‘Username’ : string,
  ‘Password’ : string,
  ‘Email’ : string,
  ‘DOB’ : date,
  ‘Fname’ : string,
  ‘Lname’ : string,
  ‘CurrencyId’ : int,
  ‘Account Type’ : string
  }]
  Output: [{
  AccountID : string
  }]

- Endpoint 2:
  Description : Read an account.
  URL : http://localhost:3000/api/account
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘AccountID’ : int
  ‘Username’ : string,
  ‘Email’ : string,
  ‘DOB’ : date,
  ‘Fname’ : string,
  ‘Lname’ : string,
  ‘AccountType’ : string,
  ‘CurrencyName’ : string
  }]

- Endpoint 3:
  Description : Update an account
  URL : http://localhost:3000/api/account
  Method: UPDATE
  Input: [{
  AccountID : string
  'Email' : string,
  'DOB' : date,
  'Fname' : string,
  'Lname' : string
  }]
  Output: [{
  'AccountID' : string
  'Username' : string,
  'Email' : string,
  'DOB' : date,
  'Fname' : string,
  'Lname' : string,
  'AccountType' : string,
  'CurrencyName' : string
  }]

- Endpoint 4:
  Description : Delete an account
  URL : http://localhost:3000/api/account
  Method: DELETE
  Input: [{
  AccountID : string
  }]
  Output: [{
  'AccountID' : string
  }]

If the 'AccountType' : 'ADMIN' then we can not allow anyone to create an account with type admin; therefore, the request for creating an account is going to be sent by an authenticated and authorized account. We can say that we have a specific ID or USERNAME, who is allowed to create this type of account. Other operations such as Read Update Delete can use the mentioned APIs endpoints (as a usual account)

- Endpoint 5 :
  Description : Create an admin account
  URL : http://localhost:3000/api/admin/account
  Method: POST
  Input: [{
  'AuthorizedAccountID' : string
  'Username' : string,
  'Password' : string,
  'Email' : string,
  'DOB' : date,
  'Fname' : string,
  'Lname' : string,
  'CurrencyId' : int,
  'Account Type' : string
  }]
  Output: [{
  AccountID : string
  }]

PAYMENT METHODS (PM)

Any account that exists in the database is allowed to have more than PMs.
Currently, there are 3 different PMs such as Credit Card (CC), Debit Card (DC), and Paypal.

We need a valid accountID to perform CRUD operations for PMs, and all accounts are authorized to interact with PM endpoints.

- Endpoint 6:
  Description : Create PaymentMethod for an account.
  URL : http://localhost:3000/api/paymentMethod
  Method: POST
  Input: [{
  'AccountID' : string
  'PaymentAccountNo' : string,
  'PaymentType' : string
  }]
  Output: [{
  'AccountID' : string
  }]

- Endpoint 7:
  Description : Read PaymentMethod.
  URL : http://localhost:3000/api/paymentMethod
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘AccountID’ : string
  ‘PaymentAccountNo’ : string,
  ‘PaymentType’ : string
  }]

- Endpoint 8:
  Description : Update PaymentMethod.
  URL : http://localhost:3000/api/paymentMethod
  Method: UPDATE
  Input: [{
  ‘AccountID’ : string
  ‘PaymentAccountNo’ : string,
  ‘PaymentType’ : string
  }]
  Output: [{
  ‘AccountID’ : string
  }]

- Endpoint 9:
  Description : Delete PaymentMethod.
  URL : http://localhost:3000/api/paymentMethod
  Method: DELETE
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘AccountID’ : string
  }]

Currency

Currency’s endpoints are not available for any account type. ADMINS are allowed to CRUD currency. However, since we have a currency for each account then we need all USERS and PARTNERS to set or get their accounts’ currencies if they want.

To access api/currency → accountType must be admin

To access api/account/currency → no restrictions needed.

- Endpoint 10:
  Description : Create currency.
  URL : http://localhost:3000/api/admin/currency
  Method: POST
  Input: [{
  ‘AccountID’ : string
  ‘CurrenyName’ : string
  ‘CurrencyFromUSD’ : int
  }]
  Output: [{
  ‘AccountID’ : string
  }]

- Endpoint 11:
  Description : Read currencies.
  URL : http://localhost:3000/api/admin/currency
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘CurrenyName’ : string
  ‘CurrencyFromUSD’ : int
  }]

- Endpoint 12:
  Description : Update currency.
  URL : http://localhost:3000/api/admin/currency
  Method: UPDATE
  Input: [{
  ‘AccountID’ : string,
  ‘CurrenyName’ : string
  ‘CurrencyFromUSD’ : int
  }]
  Output: [{
  ‘CurrenyName’ : string
  ‘CurrencyFromUSD’ : int
  }]

- Endpoint 13:
  Description : Delete currency.
  URL : http://localhost:3000/api/admin/currency
  Method: DELETE
  Input: [{
  ‘AccountID’ : string,
  ‘CurrenyName’ : string
  }]
  Output: { ‘CurrenyName’ : string }

- Endpoint 14:
  Description : Read currency for an account.
  URL : http://localhost:3000/api/account/currency
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘CurrenyName’ : string
  ‘CurrencyFromUSD’ : int
  }]

- Endpoint 15:
  Description : Update currency for an account.
  URL : http://localhost:3000/api/account/currency
  Method: UPDATE
  Input: [{
  ‘AccountID’ : string,
  ‘CurrenyName’ : string
  ‘CurrencyFromUSD’ : int
  }]
  Output: [{
  ‘CurrenyName’ : string
  ‘CurrencyFromUSD’ : int
  }]

COUNTRY
Admins are authorized to CRUD countries.

- Endpoint 16:
  Description : Create Country.
  URL : http://localhost:3000/api/admin/country
  Method: POST
  Input: [{
  'AccountID' : string
  'countryName' : string
  'CurrencyID' : int
  }]
  Output: [{
  'AccountID' : string
  }]
- Endpoint 17:
  Description : Read Countries.
  URL : http://localhost:3000/api/admin/country
  Method: GET
  Input: [{
  'AccountID' : string
  }]
  Output: [{
  'countryName' : string
  'CurrencyID' : int
  }]
- Endpoint 18:
  Description : Update Country.
  URL : http://localhost:3000/api/admin/country
  Method: UPDATE
  Input: [{
  'AccountID' : string
  'countryName' : string
  'CurrencyID' : int
  }]
  Output: [{
  'countryName' : string
  'CurrencyID' : int
  }]
- Endpoint 19:
  Description : Delete Country.
  URL : http://localhost:3000/api/admin/country
  Method: DELETE
  Input: [{
  'AccountID' : string
  'countryName' : string
  }]
  Output: { 'countryName' : string }

FEES

Admins are authorized to CRUD Fees.

- Endpoint 20:
  Description : Create Fee.
  URL : http://localhost:3000/api/admin/fee
  Method: POST
  Input: [{
  'AccountID' : string
  'FeeName' : string
  'countryName' : string,
  'feeRate' : int,
  'transActionType' : string
  }]
  Output: [{
  'AccountID' : string
  }]
- Endpoint 21:
  Description : Read fees.
  URL : http://localhost:3000/api/admin/fee
  Method: GET
  Input: [{
  'AccountID' : string
  }]
  Output: [{
  'feeID' : string
  'feeName' : string
  'countryName' : string,
  'feeRate' : int,
  'transActionType' : string
  }]
- Endpoint 22:
  Description : Update Country.
  URL : http://localhost:3000/api/admin/fee
  Method: UPDATE
  Input: [{
  'AccountID' : string
  'FeeName' : string
  'countryName' : string,
  'feeRate' : int,
  'transActionType' : string
  }]
  Output: [{
  'FeeName' : string
  'countryName' : string,
  'feeRate' : int,
  'transActionType' : string
  }]

- Endpoint 23:
  Description : Delete Country.
  URL : http://localhost:3000/api/admin/fee
  Method: DELETE
  Input: [{
  ‘feeID’ : int
  }]
  Output: { feeName: int}

USER REWARD

Admins are authorized to CRUD rewards. Users can read only their rewards.

- Endpoint 24:
  Description : Create Reward.
  URL : http://localhost:3000/api/admin/reward
  Method: POST
  Input: [{
  ‘AccountID’ : string
  ‘rewardName’ : string
  ‘rewardValue’ : int
  }]
  Output: [{
  ‘AccountID’ : string
  }]
- Endpoint 25:
  Description : Read Rewards.
  URL : http://localhost:3000/api/admin/reward
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘rewardName’ : string
  ‘rewardValue’ : int
  }]
- Endpoint 26:
  Description : Update Reward.
  URL : http://localhost:3000/api/admin/reward
  Method: UPDATE
  Input: [{
  ‘AccountID’ : string
  ‘rewardName’ : string
  ‘rewardValue’ : int
  }]
  Output: [{
  ‘rewardName’ : string
  ‘rewardValue’ : int
  }]
- Endpoint 27:
  Description : Delete Reward.
  URL : http://localhost:3000/api/admin/reward
  Method: DELETE
  Input: [{
  ‘AccountID’ : string
  ‘rewardName’ : string
  }]
  Output: { ‘rewardName’ : string }

- Endpoint 28:
  Description : Read Rewards.
  URL : http://localhost:3000/api/user/reward
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘rewardName’ : string
  ‘rewardValue’ : int
  }]

USER FRIENDS

USERS are authorized to CRD rewards.

- Endpoint 29:
  Description : Create a Friend.
  URL : http://localhost:3000/api/user/friend
  Method: POST
  Input: [{
  ‘AccountID’ : string
  ‘FriendAccountID’ : string
  }]
  Output: [{
  ‘AccountID’ : string
  }]
- Endpoint 30:
  Description : Read Friends.
  URL : http://localhost:3000/api/user/friend
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘FriendAccountID’ : string
  }]
- Endpoint 31:
  Description : Delete a Friend.
  URL : http://localhost:3000/api/user/friend
  Method: DELETE
  Input: [{
  ‘AccountID’ : string
  ‘FriendAccountID’ : string
  }]
  Output: { ‘FriendAccountID’ : string }

PARTNER REWARD

Partners can read only their rewards. There is no CRUD for admins because my team decides to put the logic partner reward as a hardcoded calculation in the backend side of the website; therefore, there is no fixed rewards system as the one that we have for the user. In other words, the INSERT keyword can be used from the backend directly to give a partner a reward.

- Endpoint 32:
  Description : Read Rewards.
  URL : http://localhost:3000/api/partner/reward
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘rewardName’ : string
  ‘rewardValue’ : int
  }]

PARTNER LOCATION
Partners can CRUD their locations

- Endpoint 33:
  Description : Create Location.
  URL : http://localhost:3000/api/partner/location
  Method: POST
  Input: [{
  ‘AccountID’ : string,
  ‘locationName’: string,
  ‘phoneNumber’ : string,
  ‘City’: string,
  ‘Country’ : string,
  ‘fName’: string,
  ‘lName’ : string
  }]
  Output: [{
  ‘AccountID’ : string
  }]

- Endpoint 34:
  Description : Read Locations.
  URL : http://localhost:3000/api/partner/location
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘locationID’: string,
  ‘locationName’: string,
  ‘phoneNumber’ : string,
  ‘City’: string,
  ‘Country’ : string,
  ‘fName’: string,
  ‘lName’ : string
  }]

- Endpoint 35:
  Description : Update Location.
  URL : http://localhost:3000/api/partner/location
  Method: UPDATE
  Input: [{
  ‘AccountID’ : string,
  ‘locationName’: string,
  ‘phoneNumber’ : string,
  ‘City’: string,
  ‘Country’ : string,
  ‘fName’: string,
  ‘lName’ : string
  }]
  Output: [{
  ‘locationName’: string,
  ‘phoneNumber’ : string,
  ‘City’: string,
  ‘Country’ : string,
  ‘fName’: string,
  ‘lName’ : string
  }]

- Endpoint 36:
  Description : Delete Location.
  URL : http://localhost:3000/api/partner/location
  Method: DELETE
  Input: [{
  ‘AccountID’ : string
  ‘locationID’ : string
  }]
  Output: { ‘loctionName’ : string }

PARTNER LOCATION HOURS

Partners can CRUD their locations HOURS

- Endpoint 37:
  Description : Create Location.
  URL : http://localhost:3000/api/partner/location/hours
  Method: POST
  Input: [{
  'AccountID' : string,
  'locationID': string,
  'dayOfWeek' : string,
  'openTime': string,
  'closeTime' : string,
  }]
  Output: [{
  'AccountID' : string
  }]
- Endpoint 38:
  Description : Read Locations.
  URL : http://localhost:3000/api/partner/location/hours
  Method: GET
  Input: [{
  'AccountID' : string
  }]
  Output: [{
  'AccountID' : string,
  'locationID': string,
  'dayOfWeek' : string,
  'openTime': string,
  'closeTime' : string,
  }]
- Endpoint 39:
  Description : Update Location.
  URL : http://localhost:3000/api/partner/location/hours
  Method: UPDATE
  Input: [{
  'AccountID' : string,
  'locationID': string,
  'dayOfWeek' : string,
  'openTime': string,
  'closeTime' : string
  }]
  Output: [{
  'locationID': string,
  'dayOfWeek' : string,
  'openTime': string,
  'closeTime' : string,
  }]

- Endpoint 40:
  Description : Delete Location.
  URL : http://localhost:3000/api/partner/location/hours
  Method: DELETE
  Input: [{
  ‘AccountID’ : string
  ‘locationID’ : string,
  ‘dayOfWeek’ : string
  }]
  Output: { ‘dayOfWeek’ : string }

TRANSACTIONS

Partners and Users can CR their transactions. Admin can delete and update a transaction.

- Endpoint 41:
  Description : Create a Transaction for a user or partner.
  URL : http://localhost:3000/api/account/transaction
  Method: POST
  Input: [{
  ‘AccountID(SenderID)’ : string,
  ‘receiverID’: string,
  ‘value’ : int,
  ‘senderCurrency’: string,
  ‘receiverCurrency’ : string,
  ‘feeID’ : int
  }]
  Output: [{
  ‘AccountID’ : string
  }]
- Endpoint 42:
  Description : Read Transactions for a particular partner or user.
  URL : http://localhost:3000/api/account/transaction
  Method: GET
  Input: [{
  ‘AccountID’ : string
  }]
  Output: [{
  ‘TransID’ ; string
  ‘AccountID(SenderID)’ : string,
  ‘receiverID’: string,
  ‘value’ : int,
  ‘senderCurrency’: string,
  ‘receiverCurrency’ : string,
  ‘feeID’ : int
  }]

- Endpoint 43:
  Description : Update transaction ADMIN ONLY.
  URL : http://localhost:3000/api/admin/transaction
  Method: UPDATE
  Input: [{
  ‘AccountID(SenderID)’ : string,
  ‘receiverID’: string,
  ‘value’ : int,
  ‘senderCurrency’: string,
  ‘receiverCurrency’ : string,
  ‘feeID’ : int
  }]
  Output: [{
  ‘transID’ ; string
  ‘AccountID(SenderID)’ : string,
  ‘receiverID’: string,
  ‘value’ : int,
  ‘senderCurrency’: string,
  ‘receiverCurrency’ : string,
  ‘feeID’ : int

  }]

- Endpoint 44:
  Description : Delete transaction ADMIN ONLY.
  URL : http://localhost:3000/api/partner/admin/transaction
  Method: DELETE
  Input: [{
  ‘AccountID’ : string
  ‘transID’ : string,
  }]
  Output: { ‘transID’ : string }

AUTHENTICATIONS

For authentication, I am following the basic authentication technique. Moreover, all passwords are going to be hashed using sha-256 or bcrypt.

https://mixedanalytics.com/knowledge-base/api-connector-encode-credentials-to-base-64/

- Endpoint 45:

  Description : for authentication purposes
  URL : http://localhost:3000/api/auth/logIn
  Method: Post
  Input: [{
  ‘Username’ : Base64 Encoded,
  ‘Password’ : Base64 Encoded,
  }]

  Output: [{
  AccountID : string
  }]

AUTHORIZATIONS

For authorization, I am taking advantage of the account type attribute, which allows me to implement role based authorizations.

RESET PASSWORD

- Endpoint 46:

  Description : for authentication purposes
  URL : http://localhost:3000/api/account/resetPassword
  Method: Post
  Input: [{
  ‘AccountID’ :string,
  ‘oldPassword’ : string,
  ‘newPassword’: string
  }]

  Output: [{
  AccountID : string
  }]