NOTE : AT THE MOMENT, ALL SUCCESS CODES ARE 200 AND ALL ERROR CODES ARE 400.

* newAPI : /generate-otp – generates new otp. Clickatell API needs to be called to send text message. Creates new user object as well.

Sample Request Body.

 {

"countryCode": "+44",//required  
    "mobileNumber": “9482930193”, //required

}

Sample Response Body.

 {

"response” : “success”,

“userID” : “5”,

“otp” : “1234”

}

Sample Response Body if failed

 {  
    "response": "mobile number not valid"   
}

Sample Response Body if failed 2

{  
    "response": "country code not valid"   
}

* newAPI : /validate-otp – validates user entered otp. Make sure OTP is same.

Sample Request Body.

 {

"userID”: 4, //required

“otp":"8991" //required

}

(can have “4” or 4)

Sample Response Body 1.

 {

"response” : “OTP matched”

}

Update isValidate = ‘Y’

Sample Response Body if failed 1.

 {

"response” : “mobile number not valid”

}

Sample Response Body if failed 2 .

 {

"response” : “userID not valid”

} // checks if userId is a number, not does this user exist?

Sample Response Body if failed 3 .

 {

"response” : “otp not found for this user”

}

Sample Response Body if failed 4.

 {

"response” : “Error in updating validation”

}

Sample Response Body if failed 5.

 {

"response” : “OTP doesn’t match”

}

* newAPI : /register-ap-go – Allows registration through apple or google.

Sample Request Body. : unclear about the nature of auth.

 {

"authTknSecret": “",//required  
    "authTknUserID": “”, //required

"authTknName": “",//required  
    "authTknFamilyName": “”, //required

"mobileDevice": “",//required  
    "pushNotificationToken": “”, //required

"loggedInWith": “",//required

}

Sample Response Body

 {

"response” : “Success”,

“userID” : “5”

}

Sample Request Body if rejected pending.

* New API - /get-location-by-coords - is used only for getting data from Google API Based on Coords.
* Called if user current location is used.

Sample Request Body

{

"coords" : "23.2618794,77.3868446",

“userID” : “7”

}

Sample Response Body

{

“response” : “Success”,

“address” : “example address”

}

Sample Response Body if failed 1 .

 {

"response” : “Error in updating location”

}

Sample Response Body if failed 2 .

 {

"response” : “userID not valid”

} // checks if userId is a number, not does this user exist?

Sample Response Body if failed 3 .

 {

"response” : “Coordinates invalid. Not a string.”

}

Sample Response Body if failed 3 .

 {

"response” : “Unable to get location from coordinates given. (Google API).”

}

Any other sample response body means there is a db error and the error itself is sent through.

* New API - /get-location-by-post – getting all possible addresses of postcode.

Sample Request Body

{

"postcode" : "SY23 6GK",

“userID” : “7”

}

Sample Response Body

{

“response” : “Success”,

"address" : {"123 Main Road", “124 Main Road”, “125 Main Road”}

}

Sample Response Body if failed 1

 {

"response” : “userID not valid”

} // checks if userId is a number, not does this user exist?

Sample Response Body if failed 2 : initial string check

 {

"response” : “Postcode invalid. Not a string.”

}

Sample Response Body if failed 3 : postcode just isn’t valid.

 {

"response” : “Please enter a valid postcode.”

}

Sample Response Body if failed 4.

 {

"response” : “Error in updating location.”

}

newAPI : /get-merchant-screen – Saves userType in userMaster.

.

Returns nothing user type (the return value is irrelevant).

Sample Request Body 1.

 {

"userID”: 4, //required

"userType": "M",//required

}

Sample Request Body 2.

 {

"userID”: 4, //required

"userType": "S",//required

}

Sample Response Body if failed.

 {

“response”: “userID not valid”

} // checks if userId is a number, not does this user exist?

Sample Response Body if failed 2.

 {

“response”: “Invalid user type”

} // if user type input is neither M nor S

* newAPI : /get-merchant-details – gives details for the merchant screen after user selects ‘shopkeeper’ on Are you a shopkeeper/shopper?.

Sample Request Body.

 {

"userID”: 4, //required

"userType": "M",//required

}

Sample Response Body

 {

"Choose your type": [

{

"description": "Bar",

"shoptypeid": 1

},

{

"description": "Cafe",

"shoptypeid": 2

},

{

"description": "Cake Shop",

"shoptypeid": 3

},

{

"description": "Grocery",

"shoptypeid": 4

},

{

"description": "Hair Salon",

"shoptypeid": 5

},

{

"description": "Nails",

"shoptypeid": 6

},

{

"description": "Parlour",

"shoptypeid": 7

},

{

"description": "Newsagent",

"shoptypeid": 8

},

{

"description": "Flowerist",

"shoptypeid": 9

},

{

"description": "Stationery",

"shoptypeid": 10

},

{

"description": "Restaurant",

"shoptypeid": 11

},

{

"description": "Wine Shop",

"shoptypeid": 12

}

],

"Question": [

{

"questionid": 1,

"questiontext": "Are you a?",

"optionid": 1,

"optiontext": "Highstreet Shop",

"questiontype": "R"

},

{

"questionid": 1,

"questiontext": "Are you a?",

"optionid": 2,

"optiontext": "Corner Shop",

"questiontype": "R"

},

{

"questionid": 1,

"questiontext": "Are you a?",

"optionid": 3,

"optiontext": "Shop in a Shopping Mall",

"questiontype": "R"

},

{

"questionid": 1,

"questiontext": "Are you a?",

"optionid": 4,

"optiontext": "Shop at the Retail Park",

"questiontype": "R"

},

{

"questionid": 2,

"questiontext": "How many franchises do you have?",

"optionid": 5,

"optiontext": "None",

"questiontype": "R"

},

{

"questionid": 2,

"questiontext": "How many franchises do you have?",

"optionid": 6,

"optiontext": "1",

"questiontype": "R"

},

{

"questionid": 2,

"questiontext": "How many franchises do you have?",

"optionid": 7,

"optiontext": "2",

"questiontype": "R"

},

{

"questionid": 2,

"questiontext": "How many franchises do you have?",

"optionid": 8,

"optiontext": "3",

"questiontype": "R"

},

{

"questionid": 2,

"questiontext": "How many franchises do you have?",

"optionid": 9,

"optiontext": "4",

"questiontype": "R"

},

{

"questionid": 2,

"questiontext": "How many franchises do you have?",

"optionid": 10,

"optiontext": "5",

"questiontype": "R"

},

{

"questionid": 2,

"questiontext": "How many franchises do you have?",

"optionid": 11,

"optiontext": "More than 5",

"questiontype": "R"

}

]

}

Sample Response Body if failed.

 {

“response”: “userID not valid”

} // checks if userId is a number, not does this user exist?

Sample Response Body if failed 2.

 {

“response”: “Invalid user type”

} // if user type input is neither M nor S

Sample Response Body if failed 3.

 {

“response”: “User type for this user not found”

}

Sample Response Body if failed 4.

 {

“response”: “Unable to select shop type”

} // db error in selecting shop type

Sample Response Body if failed 5.

 {

“response”: “Unable to select question”

} // db error in selecting question

Sample Response Body if failed 6.

 {

“response”: “Incorrect user type”

} // User type input does not match the user type in the database for this user

* newAPI : /save-merchant-details – saves merchant response in db.

Sample Request Body.

 {

"userID" : "1",

"shopTypeID" : "122",

"cashierTills" : "5",

"answer" : [{"questionID" : 1, "optionID" : [35,36]}, {"questionID" : 2, "optionID" : [39]}]

}

// match each questionID to the optionID of the response the user gives

Sample Response Body.

 {

“response”: “Success”

}

Sample Response Body if failed.

 {

“response”: “userID not valid”

} // checks if userId is a number, not does this user exist?

Sample Response Body if failed 2.

 {

“response”: “shopTypeID not valid”

} // checks if shopTypeID is a number

Sample Response Body if failed 3.

 {

“response”: “cashierTills input not valid”

} // checks if cashierTills is a number

Sample Response Body if failed 4.

 {

“response”: “userID not found”

}

Sample Response Body if failed 5.

 {

“response”: “Error in inserting merchant details”

}

Sample Response Body if failed 6.

 {

“response”: “Error in inserting user options”

}

* newAPI : /create-loyalty-scheme

Sample Request Body.

{

“userID” : “4”

}

Sample Response Body.

{

"response": [

{

"Loyalty Scheme": "Cashback",

"id": 1,

"Redeem Frequency": [

"monthly",

"quarterly",

"yearly",

"any time"

],

"Points Conversion": [

"Amount Spent",

"Points Collected",

"Points Spent",

"Amount Collected"

],

"Stamps Conversion": ""

},

{

"Loyalty Scheme": "Vouchers",

"id": 2,

"Redeem Frequency": [

"monthly",

"quarterly",

"yearly",

"any time"

],

"Points Conversion": [

"Amount Spent",

"Points Collected",

"Points Spent",

"Amount Collected"

],

"Stamps Conversion": ""

},

{

"Loyalty Scheme": "Stamps",

"id": 3,

"Redeem Frequency": [

"monthly",

"quarterly",

"yearly",

"any time"

],

"Points Conversion": [

"Amount Spent",

"Points Collected",

"Points Spent",

"Amount Collected"

],

"Stamps Conversion": [

"Stamps Collected",

"Items Free"

]

}

],

"schemes": [

{

"Loyalty Scheme Type": "Ten Points One Monthly",

"Loyalty Scheme Type ID": 1,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "10",

"Amount from points": "1",

"name": "Ten Points One Monthly",

"redeem frequency": "monthly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Ten Points One Quarterly",

"Loyalty Scheme Type ID": 2,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "10",

"Amount from points": "1",

"name": "Ten Points One Quarterly",

"redeem frequency": "quarterly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Ten Points One Yearly",

"Loyalty Scheme Type ID": 3,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "10",

"Amount from points": "1",

"name": "Ten Points One Yearly",

"redeem frequency": "yearly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Ten Points One Any Time",

"Loyalty Scheme Type ID": 4,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "10",

"Amount from points": "1",

"name": "Ten Points One Any Time",

"redeem frequency": "any time",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Fifty Points One Monthly",

"Loyalty Scheme Type ID": 5,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "50",

"Amount from points": "1",

"name": "Fifty Points One Monthly",

"redeem frequency": "monthly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Fifty Points One Quarterly",

"Loyalty Scheme Type ID": 6,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "50",

"Amount from points": "1",

"name": "Fifty Points One Quarterly",

"redeem frequency": "quarterly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Fifty Points One Yearly",

"Loyalty Scheme Type ID": 7,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "50",

"Amount from points": "1",

"name": "Fifty Points One Yearly",

"redeem frequency": "yearly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Fifty Points One Any Time",

"Loyalty Scheme Type ID": 8,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "50",

"Amount from points": "1",

"name": "Fifty Points One Any Time",

"redeem frequency": "any time",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Hundred Points One Monthly",

"Loyalty Scheme Type ID": 9,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "100",

"Amount from points": "1",

"name": "Hundred Points One Monthly",

"redeem frequency": "monthly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Hundred Points One Quarterly",

"Loyalty Scheme Type ID": 10,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "100",

"Amount from points": "1",

"name": "Hundred Points One Quarterly",

"redeem frequency": "quarterly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Hundred Points One Yearly",

"Loyalty Scheme Type ID": 11,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "100",

"Amount from points": "1",

"name": "Hundred Points One Yearly",

"redeem frequency": "yearly",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Hundred Points One Any Time",

"Loyalty Scheme Type ID": 12,

"Amount Spend": "1",

"Points Collected": "1",

"Number of points": "100",

"Amount from points": "1",

"name": "Hundred Points One Any Time",

"redeem frequency": "any time",

"Stamps to Collect": null,

"Free Items": null,

"Months Expiry": 36

},

{

"Loyalty Scheme Type": "Collect 10 Stamps",

"Loyalty Scheme Type ID": 13,

"Amount Spend": null,

"Points Collected": null,

"Number of points": null,

"Amount from points": null,

"name": "Collect 10 Stamps",

"redeem frequency": null,

"Stamps to Collect": "10",

"Free Items": "1",

"Months Expiry": null

}

]

}

Sample Response Body if failed

  {  
    "response” : “Parameter Table Error”

}

Sample Response Body if failed 2

  {  
    "response” : “Incorrect Loyalty Scheme Type”

}

Sample Response Body if failed 3

  {  
    "response” : “userID not valid”

}

* newAPI : /create-offer : load information for create offer screen.

Sample Request Body.

 {

“userID” : “7”

}

Sample Response Body.

{

"response": [

{

"Offer Name": "Buy One Get One Free",

"OfferID": 17,

"OfferTypeID": 1,

"Number of Items Buying": 1,

"Number of Items Free": 1,

"% Discount": null,

"Cash Discount": null,

"Minimum Spend": null,

"Validity": "2023-09-30T23:00:00.000Z"

},

{

"Offer Name": "10% Off",

"OfferID": 18,

"OfferTypeID": 2,

"Number of Items Buying": null,

"Number of Items Free": null,

"% Discount": "10",

"Cash Discount": null,

"Minimum Spend": null,

"Validity": "2023-09-30T23:00:00.000Z"

},

{

"Offer Name": "£10 off",

"OfferID": 19,

"OfferTypeID": 3,

"Number of Items Buying": null,

"Number of Items Free": null,

"% Discount": null,

"Cash Discount": "10",

"Minimum Spend": null,

"Validity": "2023-09-30T23:00:00.000Z"

},

{

"Offer Name": "Minimum Spend 25 Get 10% off",

"OfferID": 20,

"OfferTypeID": 4,

"Number of Items Buying": null,

"Number of Items Free": null,

"% Discount": "10",

"Cash Discount": null,

"Minimum Spend": "25",

"Validity": "2023-09-30T23:00:00.000Z"

},

{

"Offer Name": "Minimum Spend 25 Get £5 off",

"OfferID": 21,

"OfferTypeID": 4,

"Number of Items Buying": null,

"Number of Items Free": null,

"% Discount": null,

"Cash Discount": "5",

"Minimum Spend": "25",

"Validity": "2023-09-30T23:00:00.000Z"

},

{

"Offer Name": "Buy 3 Get 1 Free",

"OfferID": 22,

"OfferTypeID": 5,

"Number of Items Buying": 3,

"Number of Items Free": 1,

"% Discount": null,

"Cash Discount": null,

"Minimum Spend": null,

"Validity": "2023-09-30T23:00:00.000Z"

},

{

"Offer Name": "Buy 4 Get 10% off",

"OfferID": 23,

"OfferTypeID": 5,

"Number of Items Buying": 4,

"Number of Items Free": null,

"% Discount": "10",

"Cash Discount": null,

"Minimum Spend": null,

"Validity": "2023-09-30T23:00:00.000Z"

},

{

"Offer Name": "Buy 5 Get £5 off",

"OfferID": 24,

"OfferTypeID": 5,

"Number of Items Buying": 5,

"Number of Items Free": null,

"% Discount": null,

"Cash Discount": "5",

"Minimum Spend": null,

"Validity": "2023-09-30T23:00:00.000Z"

}

]

}

The field names required for the screen are the keys of each dictionary, and if the value is null then this field is not required for this offer.

Sample Response Body if failed

  {  
    "response” : “userID not valid”

} // checks if userId is a number, not does this user exist?

Sample Response Body if failed 2

  {  
    "response” : “userID not found”

}

Sample Response Body if failed 3

  {  
    "response” : “User is not a merchant”

}

* New API : /offers-near-you

location on, get offers from this location (location must be on). Offers from where shopper has been: get id of shop and shopper from user log.

Sample Request Body.

{"userID" : 4, "coords" : "52.437711, -1.844044", "radius" : "15", "page" : "1"}

Sample Response Body.

{

"response": [

{

"offertypeid": 5,

"offerid": 22,

"description": "Buy 3 Get 1 Free",

"buyitem": 3,

"freeitem": 1,

"percentagediscount": null,

"cashdiscount": null,

"minspend": null,

"buyproductid": null,

"offerpicture": null,

"validity": "2023-09-30T23:00:00.000Z",

"numberoffers": null,

"firstserveflag": null,

"virtualflag": null,

"validflag": true,

"sellproductid": null,

"predefined": true,

"businessname": "business1",

"businesslogo": null,

"userid": 1, // userid is that of the shopkeeper.

"distance": "3.77"

},

{

"offertypeid": 5,

"offerid": 29,

"description": null,

"buyitem": null,

"freeitem": null,

"percentagediscount": null,

"cashdiscount": null,

"minspend": null,

"buyproductid": null,

"offerpicture": null,

"validity": null,

"numberoffers": null,

"firstserveflag": null,

"virtualflag": null,

"validflag": true,

"sellproductid": null,

"predefined": false,

"businessname": "business1",

"businesslogo": null,

"userid": 1,

"distance": "3.77"

}

]

}

Sample Response Body if failed 1

  {  
    "response” : “userID not found.”  
}

Sample Response Body if failed 2 : incoming userID needs to be that of a shopper

  {  
    "response” : “user is not a shopper.”  
}

Sample Response Body if failed 3 : incoming userID needs to be that of a shopper

  {  
    "response” : “Coordinates invalid. Not a string.”  
}

Sample Response Body if failed 4 : db error.

  {  
    "response” : “Error in finding offers.”  
}

* newAPI : /get-shop-information (shoppers view)

Sample Request Body.

 {

"userID”: 4, //required – of the shop from /shop-near-you – not the shopper

}

Sample Response Body.

{

"response": [

{

"offertypeid": 1,

"offerid": 6,

"description": "new offer",

"buyitem": 2,

"freeitem": 1,

"percentagediscount": "0",

"cashdiscount": "0",

"minspend": "0",

"buyproductid": 3,

"offerpicture": "",

"validity": "2022-11-23T00:00:00.000Z",

"numberoffers": 0,

"firstserveflag": false,

"virtualflag": false,

"validflag": true,

"sellproductid": 3

},

{

"offertypeid": 4,

"offerid": 8,

"description": "new offer2",

"buyitem": 2,

"freeitem": 0,

"percentagediscount": "5",

"cashdiscount": "0",

"minspend": "0",

"buyproductid": 3,

"offerpicture": "",

"validity": "2022-11-23T00:00:00.000Z",

"numberoffers": 0,

"firstserveflag": false,

"virtualflag": false,

"validflag": true,

"sellproductid": 1

},

{

"offertypeid": 7,

"offerid": 11,

"description": "new offer3",

"buyitem": 0,

"freeitem": 0,

"percentagediscount": "0",

"cashdiscount": "5",

"minspend": "10",

"buyproductid": null,

"offerpicture": "",

"validity": "2022-11-23T00:00:00.000Z",

"numberoffers": 0,

"firstserveflag": false,

"virtualflag": false,

"validflag": true,

"sellproductid": null

},

{

"offertypeid": 5,

"offerid": 12,

"description": "new offer4",

"buyitem": 2,

"freeitem": 0,

"percentagediscount": "0",

"cashdiscount": "5",

"minspend": "0",

"buyproductid": 3,

"offerpicture": "",

"validity": "2022-11-23T00:00:00.000Z",

"numberoffers": 0,

"firstserveflag": false,

"virtualflag": false,

"validflag": true,

"sellproductid": 3

}

],

"shopInformation": [

{

"address1": "25 Totteridge Village, London N20 8PN, UK",

"mobilenumber": "458672369428",

"userid": 1

}

]

}

Sample Response Body if failed

  {“response” : “error in retrieving information.”}

* newAPI : /get-profile

Sample Request Body.

 {

"userID”: 4, //required

}

Sample Response Body 1.

{

“Postcode” : “KL41”,

“Address” : “address”,

“Logo” : “logo”,

“Name Of Business” : “business name”,

“Type Of Shop” : “shop type”,

“Number of Tills” : “5”

}

Sample Response Body 2.

{

“Postcode” : “KL41”,

“Address” : “address”,

“Interests” : “….”

}

Sample Response Body if failed

  {  
    "response” : “userID not valid”

} // checks if userId is a number, not does this user exist?

Sample Response Body if failed 2

  {  
    "response” : “userID not found”

}

Sample Response Body if failed 2

  {  
    "response” : “incorrect user type” // user types not matching

}

* newAPI : /save-profile

Sample Request Body 1. : for shopkeeper

 {

"userID”: 4, //required

"shopType": "groceries",//required

“nameOfBusiness” : “Tesco”, //required

“postcode” : “postcode”, //required

“logo” : “img”, //required

“cashierTills” : “3” //required

}

Sample Request Body 2. : for shopper

 {

"userID”: 4, //required

"interests” : “interests” //required

}

Sample Response Body.

 {

"response” : “success”

}

Sample Response Body if failed 1

  {  
    "response” : “User ID not found”  
}

Sample Response Body if failed 2

  {  
    "response” : “Shop Type ID not found”  
}

Sample Response Body if failed 3 : db error

  {  
    "response” : “Error in updating profile”  
}

Sample Response Body if failed 4 : at least one of the required fields is missing

  {  
    "response” : “Invalid fields”  
}

* newAPI : /save-offer

send notification.

Sample Request Body 1

 {

"userID”: 4, //required

“offerTypeID” : 3, //required

“offerID” : 6, // only required for predefined = true

“offerName” : “First Offer”,

“validity” : “10/06/22”,

“itemsBuy” : “3”,

“itemsFree” : “1”,

“percentageDiscount” : null,

“cashDiscount” : null,

“minSpend” : null,

“offerPicture” : “link”,

“predefined” : true // required

}

Sample Response Body.

{

"response” : “success”

}

Sample Response Body if failed 1

  {  
    "response” : “userID not found.”  
}

Sample Response Body if failed 2

  {  
    "response” : “offerTypeID not found.”  
}

Sample Response Body if failed 3 : If offer is predefined, need offerID with predefined error.

  {  
    "response” : “Need offerID with predefined offer.”  
}

Sample Response Body if failed 4 : when predefined offer, offerID and offerTypeID do not match.

  {  
    "response” : “OfferID and OfferTypeID do not match.”  
}

Sample Response Body if failed 5 : Predefined needs to be Boolean, not string.

  {  
    "response” : “Predefined needs to be boolean.”  
}

Sample Response Body if failed 6

  {  
    "response” : “Error saving offer.”  
}

* newAPI : /save-loyalty-scheme

Sample Request Body 1.

 {

"userID”: 4, //required

“loyaltySchemeTypeID” : “6”, // required

“loyaltySchemeID” : “5”, // required unless user is creating new custom scheme

“loyaltySchemeName” : “name”

“amountSpend” : “10”

“pointsCollected” : “5”

“pointsRedeem” : “15”

“amountFromPoints” : “1”

“redeemFrequency” : “weekly”

“stampsCollect” : null,

“freeItems” : null,

“monthsExpire” : “36”,

“validFromDate” : “2023-10-25”,

“predefined” : true // required

“createFlag” : true //required, createFlag = true for create screen, false for modify screen. Are you creating (true) or modifying (false)?

}

Sample Response Body if failed 1

  {  
    "response” : “userID not found.”  
}

Sample Response Body if failed 2

  {  
    "response” : “offerTypeID not found.”  
}

Sample Response Body if failed 3 : If loyalty scheme is predefined, need loyaltySchemeID with predefined error.

  {  
    "response” : “Need loyaltySchemeID with predefined.”  
}

Sample Response Body if failed 4 : when predefined offer, loyaltySchemeID and loyaltySchemeTypeID do not match.

  {  
    "response” : “loyaltySchemeID and loyaltySchemeTypeID do not match.”  
}

Sample Response Body if failed 5 : Predefined needs to be Boolean, not string.

  {  
    "response” : “Predefined needs to be boolean.”  
}

Sample Response Body if failed 6

  {  
    "response” : “Error saving offer.”  
}

* newAPI : /view-loyalty-scheme

output is loyaltySchemeMaster. Gives all loyalty schemes for this userID (shopkeeper). ViewFlag is true when on the view loyalty screen, but when it’s the modify screen and you need loyalty scheme data to display, viewFlag is false.

Sample Request Body.

 {

“viewFlag” : true

}

Sample Response Body.

 {

"response": [

{

"Redeem Frequency": "quarterly",

"Loyalty Scheme Type ID": 10,

"Predefined": true,

"Loyalty Scheme Type": "Cashback",

"Valid From": "2023-09-30T23:00:00.000Z",

"Loyalty Scheme Name": "Hundred Points One Quarterly",

"Expire Flag": false,

"Points Collected": "3",

"Number of Customers": 1,

"Points Redeemed": null,

"Points To Be Redeemed": null,

"Stamps To Collect": null,

"Stamps Collected": null,

"Free Items Given": null

},

{

"Redeem Frequency": null,

"Loyalty Scheme Type ID": 15,

"Predefined": false,

"Loyalty Scheme Type": "Stamps",

"Valid From": null,

"Loyalty Scheme Name": null,

"Expire Flag": false,

"Points Collected": "4",

"Number of Customers": 1,

"Points Redeemed": null,

"Points To Be Redeemed": null,

"Stamps To Collect": null,

"Stamps Collected": null,

"Free Items Given": null

}

]

}

Sample Response Body if failed 1

  {  
    "response” : “userID not found.”  
}

Sample Response Body if failed 2

  {  
    "response” : “No Loyalty Schemes Found for user.”  
}

Sample Response Body if failed 3 // db error

  {  
    "response” : “Error in fetching data.”  
}

* newAPI : /shop-near-you

Gets shops near user.

CHANGE RADIUS ON SCREEN.

{"userID" : 4, "coords" : "52.437711, -1.844044", "radius" : "15", "page" : "1"}

Sample Request Body.

 {

"userID”: 4, //required

“coords”: “0.19028,-1.183491” //required

“radius”: “15”, //required. Radius is in miles.

“page” : “1”// page = 1 returns first 10 results, page = 2 returns 11-20 etc.

}

Sample Response Body. (could be up to 10 responses). Offercount is how many offers are available on this shop. Userid is of the shopkeeper.

 {

"response": [

{

"businessname": "business1",

"businesslogo": null,

"userid": 1,

"offercount": "2",

"distance": "3.77"

},

{

"businessname": "business2",

"businesslogo": null,

"userid": 2,

"offercount": "0",

"distance": "14.29"

}

]

}

Sample Response Body if failed 1

  {  
    "response” : “userID not found.”  
}

Sample Response Body if failed 2 : incoming userID needs to be that of a shopper

  {  
    "response” : “user is not a shopper.”  
}

Sample Response Body if failed 3 : incoming userID needs to be that of a shopper

  {  
    "response” : “Coordinates invalid. Not a string.”  
}

Sample Response Body if failed 4 : db error.

  {  
    "response” : “Error in finding shops.”  
}

THE APIs BELOW ARE INCOMPLETE.

* New API : /scan-and-go

Click on this and it will open camera, ask you to scan the shop QR Code, will download the whole inventory locally in your phone. When download is completed, it will say Ready to Go (Wait before that). It will have a camera open on the screen to scan objects, add to basket, show picture, price/discount etc. as well.

Sample Request Body.

 {

"userID”: 4, //required – of the merchant.

}

Sample Response Body.

 {

“product” : “[“banana”, “apple”, …]”

“picture” : “[“bananaPicture”, “applePicture”, …]”

“discount”: “[“15% off”, “none”]”

}

Sample Response Body if failed

  {  
    "status": "OK",  
    "message": "Failed",  
    "errors": [  
        ""  
    ]  
}

* New API : /collect-points

Click on it and it will generate a QR Code on the screen (App Function). Shopper is in offline mode (deliberately) to make sure that his function is smooth. Shopkeeper will scan this Code via approve-points. It will open the camera on shopkeeper phone, scan code and it will give you option to enter price/points and approve. This phone will update Shoppers data as well as Shopkeepers data in the backend. The API should notify Shopper with a message on how many points allocated to the Shopper. All transaction tables to be updated as well.

Sample Request Body.

 {

"userID”: 4, //required – of the merchant.

}

Sample Response Body.

 {

“product” : “[“banana”, “apple”, …]”

“picture” : “[“bananaPicture”, “applePicture”, …]”

“discount”: “[“15% off”, “none”]”

}

Sample Response Body if failed

  {  
    "status": "OK",  
    "message": "Failed",  
    "errors": [  
        ""  
    ]  
}

/collect-points

Click on it and it will generate a QR Code on the screen (App Function). Shopper is in offline mode (deliberately) to make sure that his function is smooth. Shopkeeper will scan this Code via approve-points. It will open the camera on shopkeeper phone, scan code and it will give you option to enter price/points and approve. This phone will update Shoppers data as well as Shopkeepers data in the backend. The API should notify Shopper with a message on how many points allocated to the Shopper. All transaction tables to be updated as well.

/approve-points

It will open the camera on shopkeeper phone, scan code and it will give you option to enter price/points and approve. This phone will update Shoppers data as well as Shopkeepers data in the backend. The API should notify Shopper with a message on how many points allocated to the Shopper. All transaction tables to be updated as well.

/scan-avail

Same as Scan-and-Go except

-       Only offers are produced and not the whole inventory

-       AR mode can be enabled to make sure that users are able to hunt offers around

-       In both cases, it’s a MUST that Shopper scans the Shopkeeper’s QR code using right button.

-       Additional button on screen to “Add Offer to Wallet”

/get-offers-available

Same as Scan-and-Go except

-       Only offers are produced and not the whole inventory

-       AR mode can be enabled to make sure that users are able to hunt offers around

-       In both cases, it’s a MUST that Shopper scans the Shopkeeper’s QR code using right button.

/offers-applied : shows details for offers that a user has applied showing how much money they’re saving

/add-wallet : Adding offers to cart/wallet

/view-wallet : seeing the offers you have added.