Contents

[A. Home Page 3](#_Toc57392645)

[Service calls 4](#_Toc57392646)

[B. Dashboard 5](#_Toc57392647)

[Fund allocation to each scheme. 5](#_Toc57392648)

[Service calls 6](#_Toc57392649)

[Fund distribution to each charity house 6](#_Toc57392650)

[Service calls 7](#_Toc57392651)

[Each scheme wise performance 8](#_Toc57392652)

[Service calls 9](#_Toc57392653)

[C. Make donation 10](#_Toc57392654)

[Service calls 10](#_Toc57392655)

[D. Central Scheme Creation 15](#_Toc57392656)

[UI Wireframe 16](#_Toc57392657)

[Service calls 16](#_Toc57392658)

[E. Fund allocation 18](#_Toc57392659)

[UI Wireframe 19](#_Toc57392660)

[Service calls 19](#_Toc57392661)

[F. Fund disbursement 22](#_Toc57392662)

[Service calls 23](#_Toc57392663)

[G. Fund Allotment Report 25](#_Toc57392664)

[UI Framework 26](#_Toc57392665)

[Service calls 26](#_Toc57392666)

[H. Fund Disbursement Report 28](#_Toc57392667)

[UI Framework 29](#_Toc57392668)

[Service calls 29](#_Toc57392669)

[Table structures 31](#_Toc57392670)

[CentralSchemeMaster 31](#_Toc57392671)

[StateContractMaster 31](#_Toc57392672)

[DisbursementContractMaster 32](#_Toc57392673)

[Refers to StateContractMaster 32](#_Toc57392674)

[FFDC 33](#_Toc57392675)

[A. To get Access token 33](#_Toc57392676)

[B. Create Customer 33](#_Toc57392677)

[C. For Account creation 35](#_Toc57392678)

[D. Opening Savings account 36](#_Toc57392679)

[E. To transfer fund between accounts 37](#_Toc57392680)

[To convert the sprint boot application as Docker 38](#_Toc57392681)

# Home Page

This is the home page where the actors will login to the system. Donation to various schemes also made here.

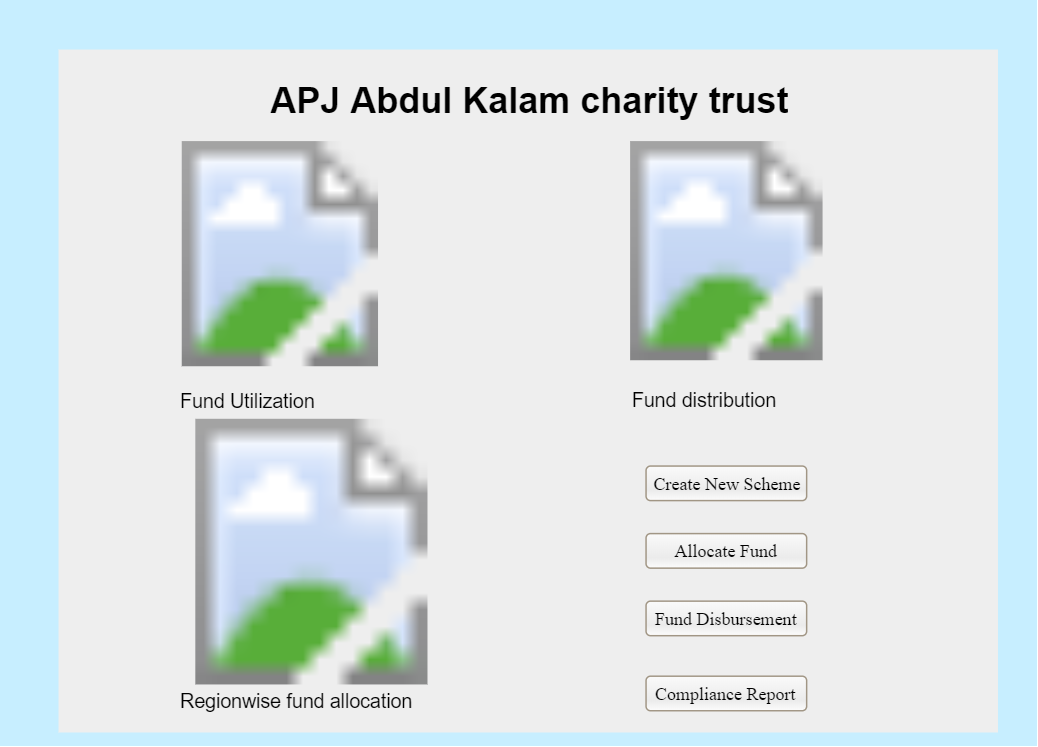


All fields in the page are static. No dynamic content. The two areas left and right to the login central area is the place holder to place any creative contents related to charity.

## Service calls

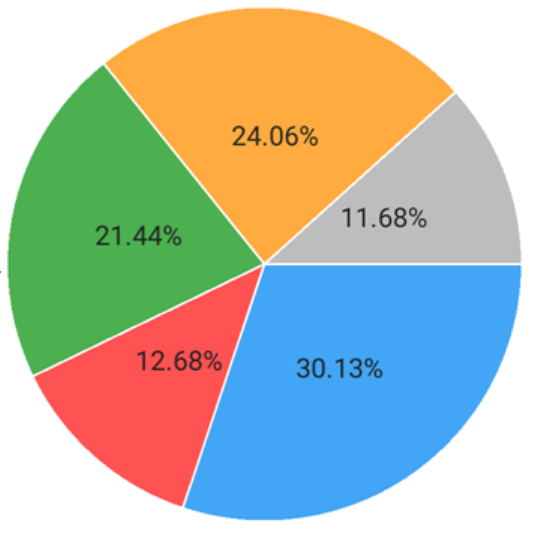
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Button/Field | Endpoint | RESTAPI Format | Host Controller endpoint | Remarks |
| 1 | Submit |  |  |  | Static flow takes to dashboard. Based on the type of user logging in we may need to configure different dash  Board   1. User from Charity HQ    1. All menu options will be shown    2. Dashboard the chart shows all charts created 2. User from state unit    1. Create scheme should not be shown.    2. Dashboard chart should be limited to the schemes allocated to the unit |
| 2 | Cancel |  |  | NA | Clear the screen |
| 3 | Make Donation |  |  |  | Takes the user to the make donation page |

# Dashboard



## Fund allocation to each scheme.

Each pie represents the amount allocated for the scheme.

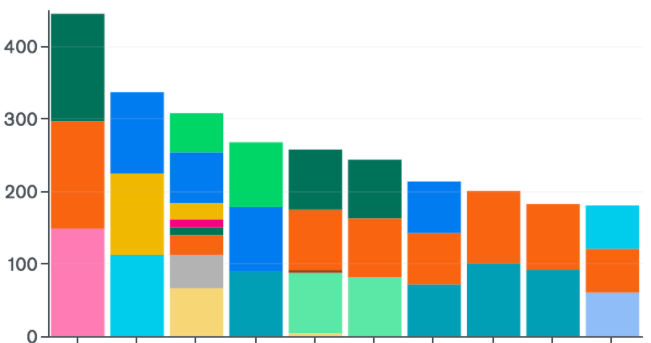


## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Button/Field | Endpoint and payload | Output REST API Format | Host Controller endpoint | Remarks |
| 1 | Scheme | <http://localhost:8091/scheme/> getSchemeSummary | {“NoOfSchemes”:”6”,”TotalAmount”:”5000”,”Scheme1”:”1000”, ”Scheme2”:”1000”, ”Scheme3”:”1000”, ”Scheme4”:”1000”, ”Scheme5”:”1000”} | ? |  |

## Fund distribution to each charity house

Each charity house will be taking care of multiple scheme. This view will give how many schemes allocated to each house and how much. Each colour represents one house.



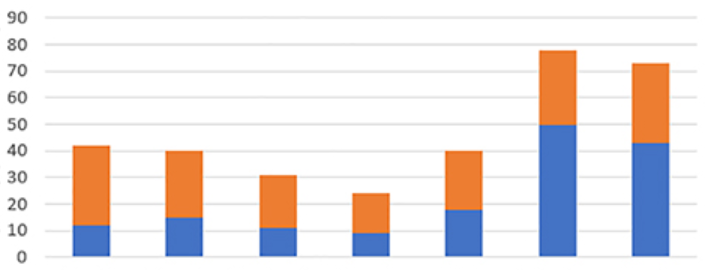
## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Button/Field | Endpoint and Input format | Output REST API Format | Host Controller endpoint | Remarks |
| 1 | Scheme | <http://localhost:8091/scheme/> getStateSummary | {  "KeralaHome": {  "scheme": [{“name”:”SchoolFee”,”SanctionAmount”:5000,”RemainingAmount”:1000}, {“name”:”SchoolFee”,”SanctionAmount”:5000,”RemainingAmount”:1000}]  },  "KarnatakaHome": {  "scheme": [{“name”:”SchoolStationaryDistribution”,”SanctionAmount”:5000,”RemainingAmount”:1000}, {“name”:”SchoolFee”,”SanctionAmount”:5000,”RemainingAmount”:1000}]  }  } | ? |  |

## Each scheme wise performance

This view gives each house wise fund consumed and the amount returned for a selected scheme

Scheme Name: 

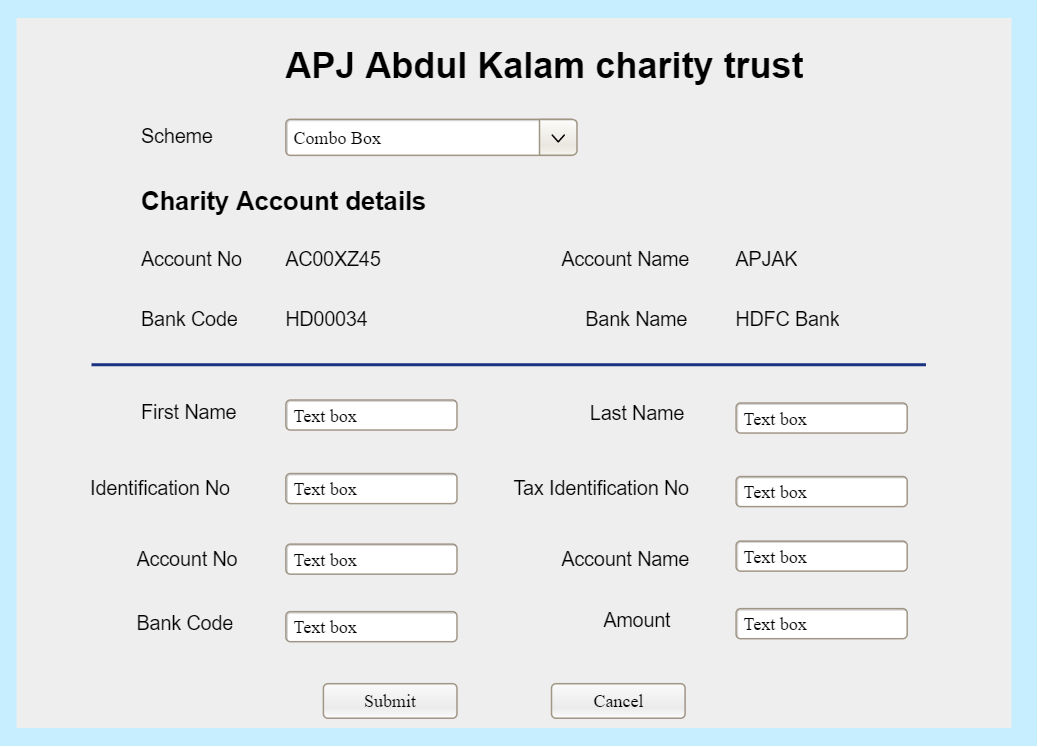


## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Button/Field | Endpoint and Input format | Output REST API Format | Host Controller endpoint | Remarks |
| 1 | Scheme Name | <http://localhost:8091/scheme/> getCentralSchemeList  {“centralSchemeAddress”:”0Xweerrw2424324sdfsdfds”} | {  "<SchemeName1>": {  "schemeDetails": [{“SchemeName”:”SchoolFee”,”SanctionAmount”:5000, ”BalanceAmount”:1000,"accountNumber":"sdfdsfs","accountName":"sfsdfdsf", "bankcode":"AS232 32SS","CentralAddress":"sdfsdfdsfd"}]},  "<SchemeName2>": {  "schemeDetails": [{“SchemeName”:”SchoolFee”,”SanctionAmount”:5000,”BalanceAmount”:1000, "accountNumber":"sdfdsfs","accountName":"sfsdfdsf","bankcode":"AS23232SS", "CentralAddress":"sdfsdfdsfd"}]}  } | ? | Returns the list of active schemes from local DB |
| 2 | For the graph | <http://localhost:8091/scheme/> getCentralStateSchemeDetails  {“centralSchemeAddress”:”0Xweerrw2424324sdfsdfds”} | {  "KeralaHome": {  "scheme": [{“name”:”SchoolFee”,”SanctionAmount”:5000,”RemainingAmount”:1000} ]},  "KarnatakaHome": {  "scheme": [{“name”:”SchoolStationaryDistribution”,”SanctionAmount”:5000,”RemainingAmount”:1000}  ]}  } |  |  |

# Make donation

This is the screen where the user will make a donation to a scheme



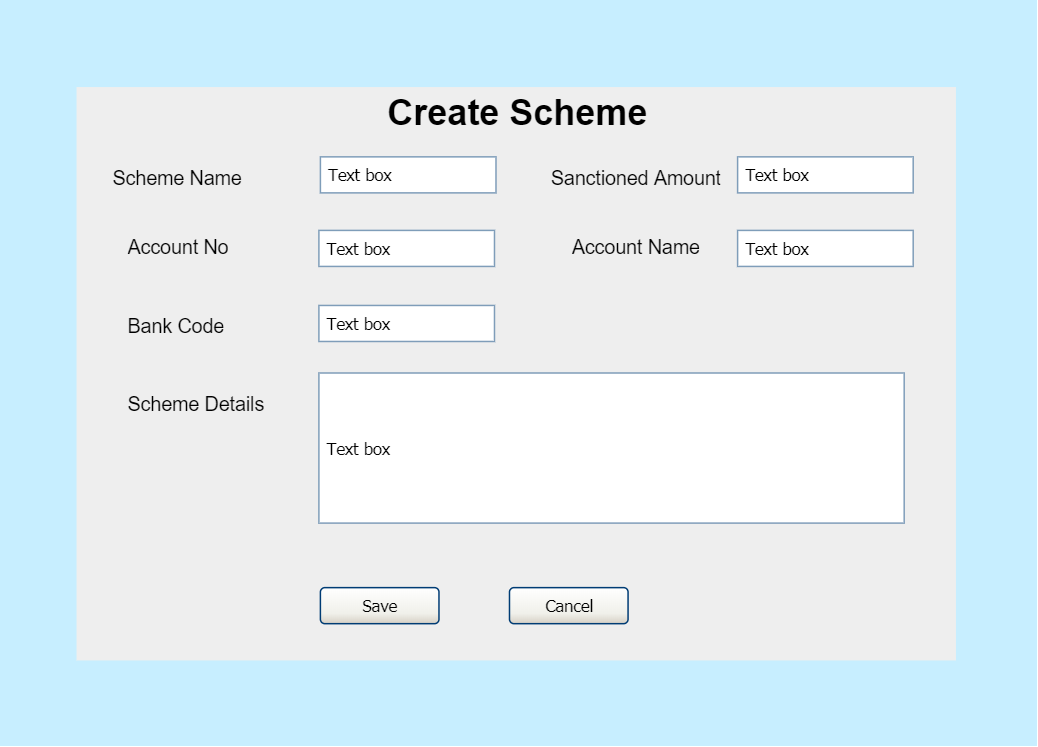
## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Button/Field | Endpoint | RESTAPI Format | Host Controller endpoint | Remarks |
| 1 | Scheme | <http://localhost:8091/scheme/> getCentralSchemeList | {  "<SchemeName1>": {  "schemeDetails": [{“SchemeName”:”SchoolFee”,”SanctionAmount”:5000, ”BalanceAmount”:1000,"accountNumber":"sdfdsfs","accountName":"sfsdfdsf", "bankcode":"AS232 32SS","CentralAddress":"sdfsdfdsfd"}]},  "<SchemeName2>": {  "schemeDetails": [{“SchemeName”:”SchoolFee”,”SanctionAmount”:5000,”BalanceAmount”:1000, "accountNumber":"sdfdsfs","accountName":"sfsdfdsf","bankcode":"AS23232SS", "CentralAddress":"sdfsdfdsfd"}]}  } | ? | List down all the active schemes. On selection of the scheme the details of the scheme is shown to the user. Display only field. Fetched from local DB |
| 2 | Account No | <http://localhost:8091/scheme/> getCentralSchemeDetails  {"centralContractAddress": "0x7df66a0328eb8046aa044e25705e267396d6c6b8"} | {      "schemeName": "TestScheme\_12",      "schemeAmount": 10001,      "balanceAmount": 0,      "centralContractAddress": "0x2b7e6285fcdc69ac15d76c982c08c780ff983cf8",      "accountNumber": "AC00EFV441",      "accountName": "SteveJobs1",      "bankcode": "HSBC00021",      "schemeDetails": "This the scheme details",      "schemeBalanceAmount": 0  } | /getCentralSchemeDetails | Displays the account number associated to the scheme. Will be stored in both the place. |
| 3 | Account Name | /getCentralSchemeDetails | Name of the account . Will be stored in both the place. |
| 4 | Bank Code | /getCentralSchemeDetails | Bank Code i.e) IFSC code, Sort code. Will be stored in both the place. |
| 5 | Bank Name | <http://localhost:8091/scheme/> getBankName  {“bankCode”:”ICICI00001”} | {  "BankCode":"ICICI0002","BankName":"ICICI Bank"  } | TBD | From local DB as master data look up. Or remove it. |
| 6 | First Name |  |  |  | First Name of the donor. Stored only in the local DB |
| 7 | Last Name |  |  |  | Last Name of the donor. Stored only in the local DB |
| 8 | Identification No |  |  |  | Unique identification number for the donor i.e) Adhaar No, Passport No etc. Stored in both the place. |
| 9 | Tax Identification No |  |  |  | Tax identification No for generating the tax certificate i.e) PAN. Stored in both the place. |
| 10 | Account No |  |  |  | Donor’s account number from where the amount will be taken. Stored in both the place. |
| 11 | Account Name |  |  |  | Donor’s account name from where the amount will be taken. Stored in both the place. |
| 12 | Bank Code |  |  |  | Donor’s account’s bank code from where the amount will be taken. Stored in both the place. |
| 13 | Amount |  |  |  | Amount to be transferred to the scheme account. . Stored in both the place. |
| 14 | Submit | <http://localhost:8091/scheme/makeDonation>  {      "centralAddress": "0x4b9310d4201d27d3e60fc42f886d0eda604dfc22",      "identificationNo": "ASCER234242",      "taxIdentificationNo": "SEFE53454",      "donorAccountNo": "AC00223e",      "donorAccountName": "Test Donor",      "donorBankCode": "HSBC00333",      "donationAmount": "400003"  } | {      "centralAddress": "0x2B7e6285fCdc69AC15D76c982C08C780ff983Cf8",      "donationAddress": "0xd865635a27b5555d83d10bad06d38e3a5623bb5f",      "identificationNo": "ASCER234242",      "taxIdentificationNo": "SEFE53454",      "donorAccountNo": "AC00223e",      "donorAccountName": "Test Donor",      "donorBankCode": "HSBC00333",      "donationAmount": 400003  } | /makeDonation | Will take it to payment gateway and once user authorize, the transaction will be completed.  DonationContract Address will be generated and stored in local DB |
| 15 | Cancel |  |  |  | Clears the UI |

# Central Scheme Creation

This is the screen in which the welfare scheme will be created.

## UI Wireframe



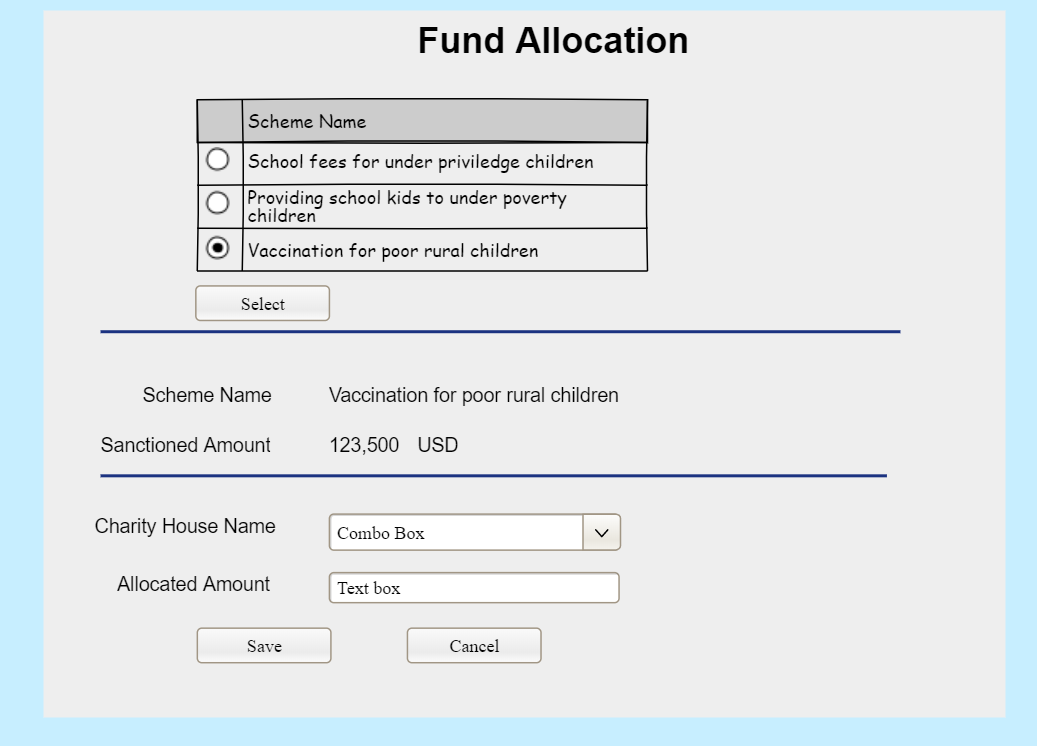
## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Button/Field | Endpoint | RESTAPI Format | Host Controller endpoint | Remarks |
| 1 | Scheme Name | NA | NA | NA | Data entry screen. Will be stored in both the place. |
| 2 | Sanctioned Amount | NA | NA | NA | Data entry screen. Will be stored in both the place. |
| 3 | Account No | NA | NA | NA | Data entry screen. Will be stored in both the place. |
| 4 | Account Name | NA | NA | NA | Data entry screen. Will be stored in both the place. |
| 5 | Bank Code | NA | NA | NA | Data entry screen. Will be stored in both the place. |
| 6 | Scheme Details | NA | NA | NA | This information will be stored only in local DB |
| 7 | Save | <http://localhost:8091/scheme/create>  {      "schemeName": "TestScheme\_12",      "schemeAmount": "10001",      "accountNumber": "AC00EFV441",      "accountName": "SteveJobs1",      "bankcode": "HSBC00021",      "schemeDetails":"This the scheme details"  } | {      "schemeName": "TestScheme\_13",      "schemeAmount": 10001,      "centralContractAddress": "0x60a2536b69e661ebd0b10586b8904a220c3cc688",      "accountNumber": "AC00EFV441",      "accountName": "SteveJobs1",      "bankcode": "HSBC00021",      "schemeDetails": "This the scheme details",      "schemeBalanceAmount": 10001  } | /create | To store the scheme details using central contract. Input for the call are@SchemeName and @SanctionedAmount. The contract address generated and Scheme details to be stored in local storage. It needs to be passed for state contract creation because the state contracts are created for a central contract |
| 8 | Cancel |  |  | NA | Clear the screen |

# Fund allocation

This is the screen in which the sanctioned amount will be allocated to individual states

## UI Wireframe

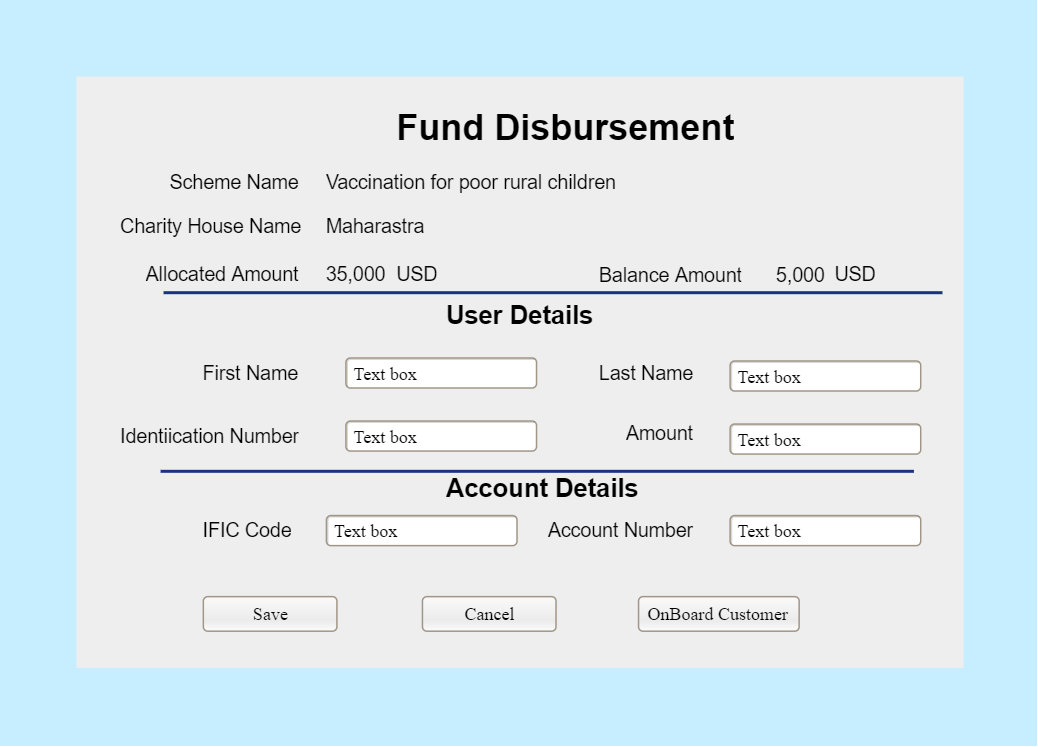


## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Field / Button | EndPoint | Rest API Format | HostCall | Remarks |
| 1 | Scheme List | <http://localhost:8091/scheme/> getCentralSchemeList | {  "<SchemeName1>": {  "schemeDetails": [{“SchemeName”:”SchoolFee”,”SanctionAmount”:5000, ”BalanceAmount”:1000,"accountNumber":"sdfdsfs","accountName":"sfsdfdsf", "bankcode":"AS232 32SS","CentralAddress":"sdfsdfdsfd","StateAddress":"sdfsfsdfd"}]},  "<SchemeName2>": {  "schemeDetails": [{“SchemeName”:”SchoolFee”,”SanctionAmount”:5000,”BalanceAmount”:1000, "accountNumber":"sdfdsfs","accountName":"sfsdfdsf","bankcode":"AS23232SS", "CentralAddress":"sdfsdfdsfd","StateAddress":"sdfsfsdfd"}]}  } | ? | Will be fetched from local DB along with the addres |
| 2 | Scheme Name | <http://localhost:8091/scheme/> getCentralSchemeDetails  {"centralContractAddress": "0x7df66a0328eb8046aa044e25705e267396d6c6b8"} | {      "schemeName": "TestScheme\_13",      "schemeAmount": 10001,      "centralContractAddress": "0x60A2536B69e661eBd0b10586b8904a220c3cc688",      "accountNumber": "AC00EFV441",      "accountName": "SteveJobs1",      "bankcode": "HSBC00021",      "schemeDetails": **null**,      "schemeBalanceAmount": 10001  } | /getCentralSchemeDetails | Fetched from |
| 3 | Sanctioned Amount | /getCentralSchemeDetails | Display only field from previous drop down selection |
| 4 | Charity House Name | <http://localhost:8091/scheme/> getCharityHouseList | {  "<HomeName1>": {  "HomeDetails": [{“HomeName”:”Kerala”,”StateId”:35}]},  "<HomeName1>": {  "HomeDetails": [{“HomeName”:”Karnataka”,”StateId”:36}]}  } | ? | Should fetch from local DB. |
| 5 | Allocated Amount | NA | NA | NA | Data entry field |
| 6 | Save | <http://localhost:8091/scheme/> disburseAmountToState  {"centralAddress":"0x60A2536B69e661eBd0b10586b8904a220c3cc688","stateId":"34","sanctionedAmount":"1000"} | {      "centralAddress": "0x60A2536B69e661eBd0b10586b8904a220c3cc688",      "stateId": 34,      "sanctionedAmount": 1000,      "returnedAmount": 0,      "stateContractAddress": "0xdaa798713ac6607b1e7c74c18a055e2c3d5c4349",      "schemeName": **null**  } | /disburseAmountToState | To allocate the funds to individual states. The input for this call @CenralContractaddess, @StateId and @AlloatedAmount. The State contract address and details to be stored in local storage. It needs to be passed for disbursement contract as the disbursement is for the specific state |
| 7 | Cancel |  |  | NA | Clear the screen |

# Fund disbursement

This the screen where the amount is finally reaches the beneficiary. The beneficiary details and his account details are captured.



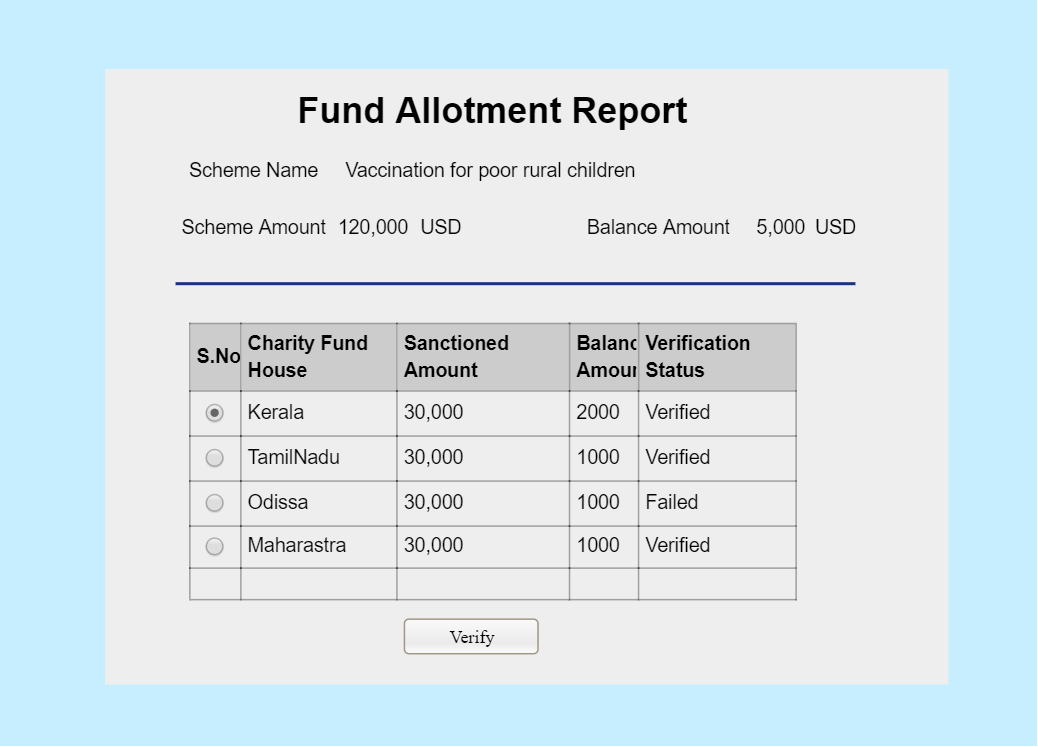
## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Field / Button | EndPoint | Rest API format | HostCall | Remarks |
| 1 | Scheme Name | <http://localhost:8091/scheme/> getCentralSchemeDetails  {"centralContractAddress": "0x7df66a0328eb8046aa044e25705e267396d6c6b8"} | {      "schemeName": "TestScheme\_13",      "schemeAmount": 10001,      "centralContractAddress": "0x60A2536B69e661eBd0b10586b8904a220c3cc688",      "accountNumber": "AC00EFV441",      "accountName": "SteveJobs1",      "bankcode": "HSBC00021",      "schemeDetails": **null**,      "schemeBalanceAmount": 10001  } | /getCentralSchemeDetails | The input for the call is @CentralContractaddress |
| 2 | Charity House Name | <http://localhost:8091/scheme/getStateSchemeDetails>  {"stateContractAddress":"0xd67694cf0768600f62b4e8d807d0292265b94801"} | {      "centralAddress": "0x60A2536B69e661eBd0b10586b8904a220c3cc688",      "stateId": 34,      "sanctionedAmount": 1000,      "returnedAmount": 1000,      "stateContractAddress": "0xdaa798713ac6607b1e7c74c18a055e2c3d5c4349",      "schemeName": "TestScheme\_13"  } | /getStateSchemeDetails | The input for the call is state contract address |
| 3 | Allocated Amount | /getStateSchemeDetails | The input for the call is state contract address |
| 4 | Balance Amount | /getStateSchemeDetails | The input for the call is state contract address |
| 5 | First Name |  |  |  | Data entry field. Stored only in Local DB |
| 6 | Last Name |  |  |  | Data entry field . Stored only in Local DB |
| 7 | Identification Number |  |  |  | Data entry field. Stored In both the place |
| 8 | Amount |  |  |  | Date entry field. Stored In both the place |
| 9 | IFIC Code |  |  |  | Data entry field. Stored In both the place |
| 10 | Account Number |  |  |  | Data entry field. Stored In both the place |
| 11 | Save | <http://localhost:8091/scheme/disburseAmountToIndividual>  {"stateContractAddress":"0xDAa798713ac6607B1E7c74C18a055E2C3D5C4349","identificationNumber":"12334534","bankCode":"450054","accountNumber":"AC00223","disbursementAmount":"10000"} | {      "identificationNumber": "12334534",      "bankCode": "450054",      "accountNumber": "AC00223",      "disbursementAmount": 10000,      "stateId": 0,      "disbursementAddress": "0x21a690ddaa31d008004694c540ff88ef1702e1d8",      "stateContractAddress": "0xDAa798713ac6607B1E7c74C18a055E2C3D5C4349"  } | /disburseAmountToIndividual | Input to this call is "stateContractAddress":"0x97881094B2", "identificationNumber":"12334534", "bankCode":"450054","accountNumber":"AC00223", "disbursementAmount":"10000" |
| 12 | Cancel |  |  | NA | Clear the screen |
| 13 | OnBoard Customer |  |  | NA | Take the user to customer onboard screen and call customer onboard API in FFDC. Post account creation take him to Fund disbursement screen |

# Fund Allotment Report

This report shows the subunit wise sanctioned and balance amount. This will be generated from local store. User will be able to validate the details by selecting and clicking validate.

## UI Framework



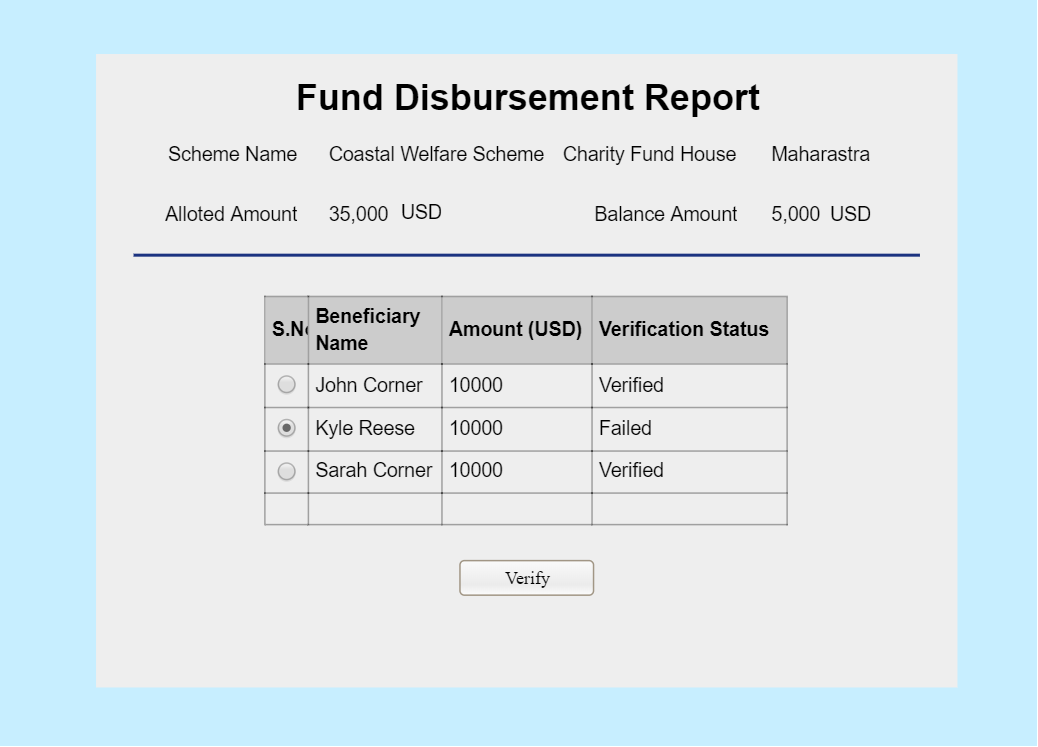
## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Field / Button | End Point | Rest API Format | HostCall | Remarks |
| 1 | SchemeName | <http://localhost:8091/scheme/> getCentralSchemeDetails  {"centralContractAddress": "0x7df66a0328eb8046aa044e25705e267396d6c6b8"} | {      "schemeName": "TestScheme\_13",      "schemeAmount": 10001,      "centralContractAddress": "0x60A2536B69e661eBd0b10586b8904a220c3cc688",      "accountNumber": "AC00EFV441",      "accountName": "SteveJobs1",      "bankcode": "HSBC00021",      "schemeDetails": **null**,      "schemeBalanceAmount": 10001  } | /getCentralSchemeDetails | The input for the call is @CentralContractaddress |
| 2 | Scheme Amount | /getCentralSchemeDetails | The input for the call is @CentralContractaddress |
| 3 | Balance Amount | /getCentralSchemeDetails | Central scheme’s balance amount |
| 4 | Charity house | <http://localhost:8091/scheme/> getCentralSchemeReport  {"centralContractAddress": "0x7df66a0328eb8046aa044e25705e267396d6c6b8"} | {  "<HomeName1>": {  "schemeDetails": [{“SchemeName”:”SchoolFee”,”SanctionAmount”:5000, ”BalanceAmount”:1000,"accountNumber":"sdfdsfs","accountName":"sfsdfdsf", "bankcode":"AS232 32SS","CentralAddress":"sdfsdfdsfd","StateAddress":"sdfsfsdfd",”VerificationStatus”:”True”}]},  "<HomeName2>": {  "schemeDetails": [{“SchemeName”:”SchoolFee”,”SanctionAmount”:5000,”BalanceAmount”:1000, "accountNumber":"sdfdsfs","accountName":"sfsdfdsf","bankcode":"AS23232SS", "CentralAddress":"sdfsdfdsfd","StateAddress":"sdfsfsdfd",”VerificationStatus”:”False”}]}  } | NA | From local data store |
| 5 | Sanctioned Amount | NA | From local data store |
| 6 | Balance Amount | NA | From local data store |
| 7 | Verification Status | NA | From local data store |
| 8 | Verify | <http://localhost:8091/scheme/getStateSchemeDetails>  {"stateContractAddress":"0xd67694cf0768600f62b4e8d807d0292265b94801"} | {      "centralAddress": "0x60A2536B69e661eBd0b10586b8904a220c3cc688",      "stateId": 34,      "sanctionedAmount": 1000,      "returnedAmount": 1000,      "stateContractAddress": "0xdaa798713ac6607b1e7c74c18a055e2c3d5c4349",      "schemeName": "TestScheme\_13"  } | /getStateSchemeDetails | Input to the call is selected state’s contract address |
| 9 | Cancel |  |  | NA | Clear the screen |

# Fund Disbursement Report

This report list down all the actual beneficiary for a specific state for the scheme.

## UI Framework



## Service calls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Field / Button | End Point | Rest API Format | HostCall | Remarks |
| 1 | Scheme Name | [http://localhost:8091/scheme/ getStateSchemeDetails](http://localhost:8091/scheme/%20getStateSchemeDetails)  {"stateContractAddress":"0xd67694cf0768600f62b4e8d807d0292265b94801"} | {      "centralAddress": "0x60A2536B69e661eBd0b10586b8904a220c3cc688",      "stateId": 34,      "sanctionedAmount": 1000,      "returnedAmount": 1000,      "stateContractAddress": "0xdaa798713ac6607b1e7c74c18a055e2c3d5c4349",      "schemeName": "TestScheme\_13"  } | /getStateSchemeDetails | Input for this call is state’s contract address |
| 1 | State | /getStateSchemeDetails | Input for this call is state’s contract address |
| 2 | Allotted Amount | /getStateSchemeDetails | Input for this call is state’s contract address |
| 3 | Balance Amount | /getStateSchemeDetails | Input for this call is state’s contract address |
| 4 | Beneficiary Name | [http://localhost:8091/scheme/ getDisbursementReport](http://localhost:8091/scheme/%20getDisbursementReport)  {"stateContractAddress":"0xd67694cf0768600f62b4e8d807d0292265b94801"} | {  "BeneficiaryID": {  "BeneficiaryDetails": [{“FirstName”:”David”, “LastName”:”John”,”DisbursedAmount”:1000,”IdentifierID”:”sdfdfddsfs”,"CentralAddress":"sdfsdfdsfd","StateAddress":"sdfsfsdfd",”DisbursementContractAddress”:”sdfsfdfs”,”VerificationStatus”:”True”}]},  "BeneficiaryID": {  "BeneficiaryDetails": [{“FirstName”:”David”, “LastName”:”John”,”DisbursedAmount”:1000,”IdentifierID”:”sdfdfddsfs”,"CentralAddress":"sdfsdfdsfd","StateAddress":"sdfsfsdfd",”DisbursementContractAddress”:”sdfsfdfs”,”VerificationStatus”:”True”}]}  } | NA | From Local DB. Hidden value of beneficiary ID to be stored |
| 5 | Amount | NA | From Local DB |
| 6 | Verification Status | NA | From local DB |
| 7 | Verify | <http://localhost:8091/scheme/getDisbursementDetails>  {"disbursementAddress":"0xedcb3a0fe27ec125ae4b3c373a5b7506f3f4c28e"} | {      "identificationNumber": "12334534",      "bankCode": "450054",      "accountNumber": "AC00223",      "disbursementAmount": 10000,      "stateId": 0,      "disbursementAddress": "0xc2142fa5d03896e7a33cdda4e719edb850410cbd",      "stateContractAddress": "0xdaa798713ac6607b1e7c74c18a055e2c3d5c4349"  } | /getDisbursementDetails | Should call two APIs to get the User ID and amount verification |
| 8 | Cancel |  |  | NA | Clear the screen |

# Table structures

## CentralSchemeMaster

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Field Name | Data Type | Remarks |
| 1 | SchemeID | Number(10) | Auto generated sequence number |
| 2 | Scheme Name | Varchar2(250) | Mandatory field |
| 3 | Scheme Amount | Number(15,2) | Mandatory field |
| 4 | ContractAddress | Varchar2(250) | Mandatory field |
| 5 | BalanceAmount | Number(15,2) | The amount that not consumed |
| 6 | Currency | Varchar(5) | Mandatory field |

## StateContractMaster

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Field Name | Data Type | Remarks |
| 1 | SchemeAllotmentID | Number(15) | Auto generated sequence number  Mandatory field |
| 2 | CentralSchemeID | Number(10) | Refers the CentralSchemeMaster.SchemeID  Mandatory field |
| 3 | StateId | Number(5) | Mandatory field |
| 4 | AllotedAmount | Number(15,2) | The original allocated amount for each state  Mandatory field |
| 5 | Currency | Varchar(5) | Mandatory field |
| 6 | BalanceAmount | Number(15,2) | The amount not yet consumed by the state. |
| 7 | StateContractAddress | Varchar2(250) | Mandatory field |

## DisbursementContractMaster

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Field Name | Data Type | Remarks |
| 1 | DisbursementID | Number(20) | Auto generated sequence number  Mandatory field |
| 2 | CentralSchemeID | Number(10) | Refers to CentralSchemeMaster.SchemeID  Mandatory field |
|  | SchemeAllotmentID | Number(15) | Refers to StateContractMaster . SchemeAllotmentID  Mandatory field |
| 3 | First Name | Varchar2(30) | Mandatory field |
| 4 | Last Name | Varchar2(30) | Mandatory field |
|  | Identification Number | Varchar2(50) | Mandatory field |
| 4 | DisbursedAmount | Number(10,2) | The original allocated amount for each state  Mandatory field |
| 5 | Currency | Varchar(5) | Mandatory field |
| 7 | DisbursementContractAddress | Varchar2(250) | Mandatory field |

POC sources are in GitHub

<https://github.com/pushparajk/Chai4SoWell/>

# FFDC

Preprod env:

<https://developer.preprod.fusionfabric.cloud/>

Application Id: 2ebd0a3a-3d10-4ae3-85bf-826b00930b4a

Secret Key: 04d00400-27f3-4d92-a1f4-16cc809e174f

## To get Access token

<https://api.preprod.fusionfabric.cloud/login/v1/sandbox/oidc/token>

grant\_type=client\_credentials

resource = [http://blr2devlp0366:8080/\*](http://blr2devlp0366:8080/*)

client\_secret = 04d00400-27f3-4d92-a1f4-16cc809e174f

client\_id = 2ebd0a3a-3d10-4ae3-85bf-826b00930b4a

Branch code as : 00000001

The returned token needs to be added in subsequent calls header as Authorization. The token needs to be prefixed with Bearer.

## Create Customer

https://api.preprod.fusionfabric.cloud/retail-banking/customers/v1/personal-customers/

Created John Walker Customer ID: 024340

Created John Walker Jr Customer ID: 024762

**Sample payload for customer onboarding**

{

  "branchCode": "00000001",

  "title": "Doctor",

  "firstName": "John",

  "lastName": "Walker Jr",

  "dateOfBirth": "1975-05-01",

  "gender": "MALE",

  "countryOfResidency": "US",

  "identification": {

    "type": "CCPT",

    "id": "string"

  },

  "kycCheckRequired": "CORE-DEFINED",

  "addresses": [

    {

      "addressType": "BUSINESS",

      "country": "US",

      "line1": "Starbucks Branch",

      "line2": "201 Powell Street",

      "line3": "",

      "line4": "San Francisco",

      "line5": "CA",

      "postalCode": "94102",

      "buildingNumber": "string"

    }

  ],

  "phoneNumbers": [

    {

      "type": "MOBILE",

      "number": "0044 01753 573244"

    }

  ],

  "emailAddresses": [

    {

      "type": "HOME",

      "address": "OfficeAdmin@OfficeAddress.com"

    }

  ],

  "fatcaDetails": {

    "isUSResident": **true**,

    "isUSTaxResident": **true**,

    "tin": "234581"

  }

}

Account with amount is customer / account id : PTYBILL3 / 01020YBILL300 [It has more amount.]

Sample customer

024762

024340

## For Account creation

<https://api.preprod.fusionfabric.cloud/retail-banking/current-and-savings-account/onboarding/v1/accounts/currentsavings>

**Sample payload**

{

    "accountDetails": {

        "customerId": "024762",

        "customerName": "John Walker Jr",

        "branchCode": "00000001",

        "productId": "01010DEFAULTUSD",

        "productDescription": "Regular Current Account - Individual - USD",

        "accountName": "JOHN Jr",

        "accountOwnership": "SOLE",

        "modeOfOperation": "SOLE",

        "postingRestriction": "DEBITCREDIT"

    },

    "accountId": "010100247620000",

    "createDateTime": "2020-11-18T07:09:42.384+0000"

}

## Opening Savings account

https://api.preprod.fusionfabric.cloud/retail-banking/current-and-savings-account/onboarding/v1/accounts/currentsavings

{

    "accountDetails": {

        "customerId": "024340",

        "customerName": "John Walker",

        "branchCode": "00000001",

        "productId": "01010DEFAULTUSD",

        "productDescription": "Regular Current Account - Individual - USD",

        "accountName": "JOHN",

        "accountOwnership": "SOLE",

        "modeOfOperation": "SOLE",

        "postingRestriction": "DEBITCREDIT"

    },

    "accountId": "010100243400000",

    "createDateTime": "2020-11-18T07:07:34.904+0000"

}

Sample accounts created

024762 / 010100247620001 🡪 100 USD credited from 01020YBILL300

024340 / 010100243400001

## To transfer fund between accounts

https://api.preprod.fusionfabric.cloud/retail-banking/transactions/v1/financial-postings

Payload

{

    "forcePostIndicator": **true**,

    "postingEntries": [

        {

            "accountId": "010100243400001",

            "amount": {

                "currency": "USD",

                "amount": 1

            },

            "baseEquivalent": {

                "currency": "USD",

                "amount": 1

            },

            "exchangeRate": 1,

            "blockReference": "",

            "chequeNumber": "",

            "creditDebitIndicator": "CREDIT",

            "narrative": "Fund Transfer",

            "postingType": "C01"

        },

        {

            "accountId": "010100247620001",

            "amount": {

                "currency": "USD",

                "amount": 1

            },

            "baseEquivalent": {

                "currency": "USD",

                "amount": 1

                },

            "exchangeRate": 1,

            "blockReference": "",

            "chequeNumber": "",

            "creditDebitIndicator": "DEBIT",

            "narrative": "Fund Transfer",

            "postingType": "C00"

        }

    ],

    "sourceBranchCode": "00000001",

    "sourceId": "FFDC",

    "transactionReference": "c35bc8cf-5aad-427e-82d9-44dd9977e77c",

    "valueDate": "2020-11-18"

}

# To convert the sprint boot application as Docker

1. Generate the jar file from the sprint boot application. It can be done by selecting Maven 🡪 Maven build … 🡪 And specify the “package”. The jar will be generated under target folder.
2. Go to the folder above the target folder and keep the below Dockerfile

FROM openjdk:11-jre-slim

MAINTAINER "Catena"

WORKDIR /app

COPY ./target/\*.jar ./app.jar

ENTRYPOINT ["java", "-jar", "/app/app.jar"]

EXPOSE 8080

1. Create the docker for mongo DB using the below command

docker network create catenabend

docker run -d -p 27017:27017 --name <name of the docker image> --network <network name> -t mongo:latest

eg)

docker run -d -p 27017:27017 --name mongodb1 --network catenabend -t mongo:latest

To create without network use the below command

docker run -p 27017:27017 --name mongodb -d mongo:latest

1. Now run the docker command for Ethereum

docker run -d --name ethereum --network catenabend -p 8545:8545 ethereum/client-go:v1.8.6 --rpc --rpcaddr "0.0.0.0" --rpcapi="db,eth,net,web3,personal" --rpccorsdomain "\*" --dev

1. Now inspect the network catenabend

* Docker network inspect catenabend

"Containers": {

"7974d3910da061ff881f17f7b4e1bd6d23f94caee226b21480e0d63ea09fc17e": {

"Name": "mongodb1",

"EndpointID": "c4b4b8c98c89c79d40601d9ad79c4396786cb6f699ebbfdf27a225daca506a4e",

"MacAddress": "02:42:ac:13:00:02",

"IPv4Address": "172.19.0.2/16",

"IPv6Address": ""

},

"7a1668decf84e62f6da7233df5dff2add5022503e8a6362b7728e61dbfeae16c": {

"Name": "ethereum",

"EndpointID": "6fbdd1313ffd77cb190d7cea76f5ebb28f34d90bb5834d2e4115a78c50694c89",

"MacAddress": "02:42:ac:13:00:03",

"IPv4Address": "172.19.0.3/16",

"IPv6Address": ""

},

"e0678709e1c03a63a82ad432a7bfbbf01ad4b76fd939e0a457d91454f53744a8": {

"Name": "catena",

"EndpointID": "696313a67777a7998e60799662f5ee49449727836e5dbfc8bddbf6187d398b3a",

"MacAddress": "02:42:ac:13:00:04",

"IPv4Address": "172.19.0.4/16",

"IPv6Address": ""

}

},

1. Take the highlighted IP address for the Ethereum image i.e) 172.19.0.3
2. Mention the IP address in the **private** String ethereumURL="http://172.19.0.3:8545"; in CentralSchemeService. Run Maven build package
3. Type the following command from the folder above the target folder

docker image build -t catena .

1. Now run the docker image for spring boot application

docker run --rm -p 8080:8080 --name <name\_of\_app\_container> --net backend --link <mongo\_container\_name>:mongo <name\_of\_the\_\_app\_image>

docker run -d -p 8091:8091 --name catena --network catenabend catena

That’s it now you will be able to access the spring boot application in localhost:8091

Azure Ethereum

<https://docs.nethereum.com/en/latest/azure/set-up-blockchain-on-azure/>

We may need a scheme list screen from where the scheme will be selected and operations like Allocate fund, fund disbursement can be called.

Each report should have the option to select the scheme name

When transfer to state the central contract balance amount to be updated

<https://logomakr.com/7x5hac>

logomakr.com/47Gqry

logomakr.com/1mK8ol

