ATM Transactions API

Create a RESTful Application which has following resources:

- **/createAccount**
- **/withDraw**
- **/deposit**
- **/checkBalance**

Pre-Requisite : Create an Amazon RDS MySQL Database Table with following details :

```
CREATE TABLE `Bank_Transactions` (
 `pk` int NOT NULL AUTO_INCREMENT,
 `custName` varchar(45) NOT NULL,
 `custAccNum` varchar(10) NOT NULL,
 `atmPin` varchar(4) NOT NULL,
 `bankName` varchar(10) NOT NULL,
 `accountType` varchar(10) NOT NULL,
 `ifscCode` varchar(45) NOT NULL,
 `branchName` varchar(45) NOT NULL,
 `totalBalance` decimal(19,9) NOT NULL DEFAULT '0.000000000',
 `transactionTimeStamp` varchar(45) NOT NULL,
 `accountStatus` varchar(10) NOT NULL DEFAULT 'Active',
 `wrongPin` int DEFAULT '0',
 `mailId` varchar(45) NOT NULL,
 `phoneNumber` varchar(45) NOT NULL,
 PRIMARY KEY (`pk`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

Before getting Started , Go through instructions carefully . Remember request field names and Database column names are different. All success and failure scenarios must be handled carefully. Updating balance, wrong pin , accountStatus must be taken care wherever needed

MuleSoft 3-D : Design , Develop, Deploy

Step1 : Design RAML for following resources (DESIGN):

/createAccount :

Method : POST

queryParams :

customerName - required : max length 45 , min length 4
bank - required: should be one of enum values ICICI , AXIS ,HDFC
type - required : should be on of enum values [savings , current]
branchName - required : min : 4 max : 10

Body : json

accountNum - required: should be string and min max length is 10
atmPin : required : type string (but pass numbers in string) min max ength = 4
ifscCode - required : min : 4 max : 10
depositAmount : not mandatory . If sent , it should be number
mailId : required : string
contact : required : string

Sample Response :

201 : {“status” : “Account Created Successfully with Account Number xxxxxxxx”}

/checkBalance :

Method : POST

queryParams :

bank - required: should be one of enum values ICICI , AXIS , HDFC
type - required : should be on of enum values [savings , current]

Body : json
accountNum - required: should be string and min max length is 10
atmPin : required : type string (but pass numbers in string) min max length = 4

Sample Response :
200 : {“status” : “Your total balance is xxxxxxxx as on 20-May-2020 is xxxxxx”}

/withDraw :

Method : POST

queryParams :

bank - required: should be one of enum values ICICI , AXIS ,HDFC
type - required : should be on of enum values [savings , current]

Body : json
accountNum - required: should be string and min max length is 10
atmPin : required : type string (but pass numbers in string) min max length = 4
amountToBeWithdraw : Required. it should be number

Sample Response :
200 : {“status” : “Amount xxxx is debited . Your total balance is xxxxxxxx”}

/unblock :

Method : PUT

Body : json

user with default value “admin”

password with default value “admin”

accountNum - required: should be string and min max length is 10

bank - required: should be one of enum values ICICI , AXIS , HDFC

Sample Response :

200 : {“status” : “Account xxxxx is unblocked”}

Note : Wherever you see xxxxxx , please use corresponding values. This is just for example

Once RAML is created, publish to exchange and share the Exchange URL

You should be creating Proxy APIs . Instead of manual appending of api id’s

Please follow the process to create Proxy APIs

Step 2 Development :

/createAccount :

Create account resource should be able to successfully insert a record into Database (creating an account) .
Have below checks:

- **Before inserting a record, you have to check whether there is any account existing with same accountNumber and Bank Name. If yes, return the response saying " Account 1234567890 already exist "**
- **If depositAmount field is present : insert amount accordingly in totalBalance column. Else the table will have default value as 0.**
- **transactionTimeStamp column in Table is now() value in MuleSoft. It is mandatory field**
- **Note : By default when you insert a record accountStatus , wrongPin values are taken as Active and 0 respectively**

- **Upon successful insertion , send out a mail using GMAIl:**
    - **Subject : Congratulations ! Account created**
    - **To : the email id sent in Body**
    - **From : your gmail id**
    - **Body : Congratulations ! Your account is created Successfully with Account Number "1234567890" with ICICI Bank.**

- **Create a JSON file containing all details of the Account Holder and save file name with <AccountNumber_AccountHolderName>.json in Amazon S3**
- **Once the account is created , send out the response as mentioned in RAML to the client.**
- **Connectivity issues and any other DB related issues should be handled in Error Handling**

/checkBalance :

checkBalance resource is used to check the total Balance of Valid account

- **Before fetching total balance, you have to check whether there is any account existing with accountNumber and Bank Name.**
  - **If no, return the response saying "Account 1234567890 does not exist . Enter Valid Details"**
  - **If yes , check whether "accountStatus" = "Active" .If not, return response saying "Account 1234567890 temporarily blocked. Please visit nearest Branch"**

- **If account exists with accountNumber and Bank Name and status is "Active". Check for atmPin whether it is valid or not**

  - **If atmPin is incorrect , and "wrongPin" value is already 3 then update accountstatus column in Database to "Blocked" and return response : "Maximum Attempts reached .Your Account 1234567890 is temporarily blocked. Please reach nearest Branch"**

  - **If "wrongPin" value is less than 3 , then increment by 1 for every wrong attempt and return response saying "Login Attempt Failed .Attempts left : 2"**

- **If atmPin is correct , the display the response as described in RAML**

Note :

- **Upon failure attempts , send out a mail using GMAIl:**
  - **Subject : Failed Attempt !**
  - **To : the email id sent in Body**
  - **From : your gmail id**
  - **Body : This is to inform you that there's a failed attempt happened while transaction . Your Account will be blocked after 3 attempts**

- **Upon All attempts failed, send out a mail using GMAIl:**
  - **Subject : Account Blocked !**
  - **To : the email id sent in Body**
  - **From : your gmail id**
  - **Body : This is to inform you that Your Account has been be blocked due to 3 failed attempts. Please reach out nearest branch to unblock**

- **No Mail should be sent on successful transaction**

/withDraw :

withDraw resource is used to withdraw the amount from Valid account
All validations done for checkBalance should be done here as well (make use of sub-flows and flows so that you can re-use the logic)

- **Before fetching total balance, you have to check whether there is any account existing with accountNumber and Bank Name.**
  - **If no, return the response saying "Account 1234567890 does not exist . Enter Valid Details"**
  - **If yes , check whether "accountStatus" = "Active" .If not, return response saying "Account 1234567890 temporarily blocked. Please visit nearest Branch"**

- **If account exists with accountNumber and Bank Name and status is "Active". Check for atmPin whether it is valid or not**
  - **If atmPin is incorrect , and "wrongPin" value is already 3 then update accountstatus column in Database to "Blocked" and return response :**
    "Maximum Attempts reached .Your Account 1234567890 is temporarily blocked. Please reach nearest Branch"
  - **If "wrongPin" value is less than 3 , then increment by 1 for every wrong attempt and return response saying "Login Attempt Failed .Attempts left : 2"**

- **If atmPin is correct :**
  - **Check if total Balance is less than amountTobeWithdrawn , then send out a response saying "Insufficient Funds"**
  - **if total Balance is greater than amountTobeWithdrawn . Then deduct amount and update the Database with final balance after deduction**
  - **Send out a response as defined in RAML**

Note :

- **Upon failure attempts , send out a mail using GMAIl:**
  - **Subject : Failed Attempt !**
  - **To : the email id sent in Body**
  - **From : your gmail id**
  - **Body : This is to inform you that there's a failed attempt happened while transaction . Your Account will be blocked after 3 attempts**

- **Upon All attempts failed, send out a mail using GMAIl:**
    - **Subject : Account Blocked !**
    - **To : the email id sent in Body**
    - **From : your gmail id**
    - **Body : This is to inform you that Your Account has been be blocked due to 3 failed attempts. Please reach out nearest branch to unblock**

- **Upon Successful withdrawal , send out a mail using GMAIl:**
    - **Subject : Transaction Alert!**
    - **To : the email id sent in Body**
    - **From : your gmail id**
    - **Body : This is to inform you that Your Account has been debited with xxxx amount and your Total Balance is xxxx**

/unblock :

Unblock resource is used to unblock the blocked account
Check if user and password = admin
Check if valid combination of bank and Account number is given :

- **If no, return the response saying "Account 1234567890 does not exist . Enter Valid Details"**
- **If yes , check whether "accountStatus" = "Active" .If yes, return response saying "Account 1234567890 is Active."**
- **If "accountStatus" = "Blocked" .Update accountStatus to "Active" and wrongPin column to 0, return response saying "Account 1234567890 is Unblocked."**

Note :
- **Upon Successful unblock , send out a mail using GMAIl:**
    - **Subject : Account Unblocked!**
    - **To : the email id sent in Body**
    - **From : your gmail id**
        - **Body : This is to inform you that Your Account has been unblocked**

Step 3 Scheduler flow :

Have a scheduler in your application which will trigger a flow daily at 7pm where it should fetch the current total balance of every account and send out a mail to respective mail ids.

- **send out a mail using GMAIl:**
  - **Subject : Todays Total Balance!**
  - **To : the email id sent in Body**
  - **From : your gmail id**
    - **Body : Your total Balance as on Today is xxxxxxxx**

Use foreach and no record should stop sending out mail if there is any error occurred for any account. Hope you know what to use !

Step 4 Unit Testing :

Test All success and failure scenarios for all resources . If everything looks good. You are good to deploy.

Step 5 MUnits (optional) :

Always good to write MUnits . But its optional for now! But once you deploy your application successfully , then try to write them to have good hands-on practice . Make use of record units option.

Step 6 Deploy to Cloud Hub:

After Step 4, deploy your application to cloud hub. Give name of your application as "mulesoft-atm-transactions-yourName"

Step 7 Manage by appending Policies:

Use Any 2 policies from below to all resources . Its good if you can apply all:

1. **Basic Auth**
2. **Client ID enforcement**
3. **IP WhiteListing**
4. **Rate Limiting**

Upon Successful Deployment Share the Postman collection!

Best Practices:

- **Give RAML name , Project name and Deployable app name as "mulesoft-atm-transaction-yourName"**
- **Give proper naming to flows, private flows and sub-flows**
- **Use common functionality wherever required and refer them using flow-refs**
- **Store input Data in variables . Don't use multiple set variables , instead use Transform message to store vars**
- **Instead of storing many variables. You can save payload in one variable and then access fields individually like vars.inputPayload.custName, vars.inputPayload.atmPin etc**
- **Use camel case**
- **Have loggers wherever necessary especially when you are connecting to external systems like DB , AWS, SMPT**
- **Use Dataweave as first option , if not go with Choice router**
- **Use for-each for scheduler job requirement . Use cron expression to run everyday 7pm**
- **Set appropriate Status codes by setting vars.httpStatus**
- **Use AWS DB and S3 as local DB will not work when deployed in cloud hub**
- **Handle error handling wherever required. Use On-error Propagate mostly as On-error continue will continue the flow . So check twice before deploying.**
- **Write MUnits if you have time. Try to make use of Record Munits feature . Remember use 4.3 version of Runtime , 7.5 version of studio and Munit plugin with 2.2.5**
- **If you want to add extra functionality (only after successful completion of actual API), you can make use of sending text message to user whenever a transaction like checkBalance, withdraw , block of account happens. Make use of twilio Connector (google it :)**
- **Don't forget to share the cloud hub Proxy urls . Send postman collection of it**

I wish you All the success ahead!

Yours ,

Sravan Lingam.

Please connect and give feedback @ https://www.linkedin.com/in/sravanlingam/