**Account Service**

SQL Schema to store Account information:

createtable accounts(

account\_idINTPRIMARYKEYIDENTITY (1,1),

account\_nameVARCHAR(255)UNIQUENOTNULL,

firstnameVARCHAR(255)NOTNULL,

lastnameVARCHAR(255)NOTNULL,

email VARCHAR(255)UNIQUENOTNULL,

passwordVARCHAR(255)NOTNULL

);

**Rest APIs:**

1. Get all accounts:
   1. API method: getAllAccounts
   2. API endpoint: /accounts
   3. Request type: GET, No Body
   4. The API should return all the available accounts detail, empty if none is present
   5. Status codes:
      1. 200 – OK, if account data is been successfully retrieved
      2. 401 – Unauthorized, if user doesn’t have permission to view all account data
      3. 500 – Internal Server Error for any Database related issues
2. Get specific account by ID:
   1. API method: getAccountByID
   2. API endpoint: /accounts/{accountId}
   3. Request type: GET, no body
   4. The API should return the account by provided ID
   5. Status codes:
      1. 200 – OK, if account is present and retrieved successfully by provided ID.
      2. 401 - Unauthorized, if user doesn’t have permission to view the account specified by provided ID
      3. 400 – Bad Request, if the provided ID is invalid and account doesn’t exist with provided ID
      4. 500 – Internal Server Error, for any Database related issues
3. Get specific account by name, if present:
   1. API method: getAccountByName
   2. API endpoint: /accounts/withName/{accountName}
   3. Request type: GET, no body
   4. The API should return the account by provided account name
   5. Status codes:
      1. 200 – OK, if account is present and retrieved successfully by provided name.
      2. 401 - Unauthorized, if user doesn’t have permission to view the account specified by provided name
      3. 400 – Bad Request, if the provided name is invalid and account doesn’t exist with provided name
      4. 500 – Internal Server Error, for any Database related issues
4. Create an account:
   1. API method: createAccount
   2. API endpoint: /accounts/
   3. Request type: POST, with following JSON body

{

“name”: “<account name>”,

“firstname”: “<First name of user>”,

“lastname”: <Last name of user>,

“email”: <Email ID of user>,

“password”: <Password to be set for account>

}

* 1. The API should create an Account with provided User details
  2. Status codes:
     1. 200 – OK, if account information stored successfully
     2. 401 - Unauthorized, if user cannot perform the create account operation
     3. 400 – Bad Request, if JSON request isn’t valid or missing any of the value
     4. 409 – Conflict, if provided name already exists
     5. 500 – Internal Server error for any DB related errors

1. Update Account:
   1. API method: updateAccount
   2. API endpoint: /accounts/{accountId}
   3. Request type: PUT, with following JSON body

{

“name”: “<account name>”,

“firstname”: “<First name of user>”,

“lastname”: <Last name of user>,

“email”: <Email ID of user>,

“password”: <Password to be set for account>

}

* 1. The API should update the account information for provided account ID
  2. Status codes:
     1. 200 – OK, if account information stored successfully
     2. 401 - Unauthorized, if user cannot perform the create account operation
     3. 400 – Bad Request, if JSON request isn’t valid or missing any of the value
     4. 409 – Conflict, if provided name already exists
     5. 500 – Internal Server error for any DB related errors

1. Delete account:
   1. API method: deleteAccount
   2. API endpoint: /accounts/{accountId}
   3. Request type: DELETE, no body
   4. The API should delete the account information for provided account ID
   5. Status codes:
      1. 204 – No Content, if account is deleted successfully
      2. 401 - Unauthorized, if user cannot perform the delete account operation
      3. 400 – Bad Request, if provided accountId doesn’t exists
      4. 409 – Conflict, if provided name already exists
      5. 500 – Internal Server error for any DB related errors