Welcome to your first group project, you will be broken up into groups of either 5 or 6 people. Together over the next few days you will build a RESTful service for a banking application.

Task 1: Create the restful endpoint for an account object.

Account
- id: **Long** - type: **Enum** - nickname:**String** - rewards: **Integer** - balance:**Double** - customer:**Customer**

- **id** - is an auto generated value which should be assigned by the database
- **type** - is an enumerated value which when serialized should print out as a string
  - Savings
  - Checking
  - Credit
- **nickname** - is a string value assigned at creation to describe the account i.e “Leon’s savings account”
- **rewards** - is a integer value which represents the reward points assigned to an account
- **balance** - is the amount of money available in the account which can be applied to purchases.
- **customer-** customer object associated with account

Method: GET	URI: /accounts	Action: Get all accounts
Method: GET	URI: /accounts/{accountId}	Action: Get account by id
Method: GET	URI: /customers/{customerId}/accounts	Action: Get all accounts for customer
Method: POST	/customers/{customerId}/accounts	Create an account
Method: PUT	/accounts/{accountId}	Update a specific existing account
Method:DELETE	/accounts/{accountId}	Delete a specific existing account

Account: List all of the accounts available

Method: GET	URI: /accounts	Action: Get all accounts

Example Failed Response: Status (404)
```
{
   “code” : 404,
    “message”: “error fetching accounts”
}
```

Example Successful Response: Status (200)
```
{
   “code” : 200,
    “message”: “Success”,
   “data”:
           [
               {
                   "id": 1,
                   "type": "Savings",
                   "nickname": "Leon’s Account",
                   "rewards": 37605,
                   "balance": 19351,
                   "customer_id": 10
               },
               {
                   "id": 2,
                   "type": "Savings",
                   "nickname": "Zach Account",
                   "rewards": 37605,
                   "balance": 19351,
                   "customer_id": 11
               }
           ]
}
```

Account: Get one account by customer id

Method: GET	URI: /accounts/{customerId}	Action: Get account by id

Example Failed Response: Status (404)
```
{
   “code” : 404,
    “message”: “error fetching account”
}
```

Example Successful Response: Status (200)
```
{
   “code” : 200,
    “message”: “error fetching account”,
    “data”:
   {
       "id": 1,
       "type": "Savings",
       "nickname": "Leon’s Account",
       "rewards": 37605,
       "balance": 19351,
       "customer_id": 10
   }
}
```

Account : Update a account by account id

PUT	/accounts/{accountId}	Update a specific existing account

Example Failed Response: Status (404)
```
{
   “code” : 404,
    “message”: “Error ”,
}
```

Example Successful Response: Status (200)

```
{
  "code": 200,
  "message": "Customer account updated"
}
```

Account: Get all accounts belonging to one customer by customer ID

Method: GET	URI: /customers/{customerId}/accounts	Action: Get all accounts for a customer

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “Error fetching customers accounts”
}
```

Example Successful Response: Status (200)

```
{
   “code” : 200,
    “message”: “Success”,
   “Data”:
          [
            {
              "id": 1,
              "type": "Savings",
              "nickname": "Leon’s Saving Account",
              "rewards": 37605,
              "balance": 19351,
              "customer_id": 10
            },
           {
              "id": 32,
              "type": "Checking",
              "nickname": "Leon’s Checking Account",
              "rewards": 37605,
              "balance": 22,
              "customer_id": 10
            }
          ]
}
```

Account: Create a new account and assign it to a customer by the customer id		
POST	/customers/{customerId}/accounts	Create an account

Example Failed Response: Status (404)
{
    “code” : 404,
     “message”: “error fetching creating customers account”
}		

Example Successful Response: Status (201)
{
  "code": 201,
  "message": "Account created",
  "data": {
    "type": "Credit Card",
    "nickname": "string",
    "rewards": 0,
    "balance": 0,
    "customer_id": 10,
    "_id": 42
  }
}		

Account: Delete a account by account id		
DELETE	/accounts/{accountId}	Delete a specific existing account

Example Failed Response: Status (404)
{
    “code” : 404,
     “message”: “Account does not exist”
}		

Example Successful Response: Status (201)
 	
{
  "code": 202,
  "message": "Account successfully deleted"
}		

Task 2: Create the restful endpoint for a Customer object.

Customer
+ id:Long + first_name:String + last_name:String + address: Set<Address>

- **id** - is an auto generated value which should be assigned by the database
- **first_name -** first name of customer
- **last_name -** last name of customer
- **address -** A set of addresses associated with customer

Address
+ id:Long + street_number:String + street_name:String + city:String + state:String + zip: String

- **id** - is an auto generated value which should be assigned by the database
- **street_name -** name of street
- **street_number -** street number
- **city -** city of customer
- **state -** state customer lives in
- **zip -** zip code of customer

Method: GET	/accounts/{accountId}/customer	Get customer that owns the specified account
Method: GET	/customers	Get all customers
Method: GET	/customers/{id}	Get customer by id
Method: POST	/customers	Create a customer
Method: PUT	/customers/{id}	Update a specific existing customer

Customer: List all of the accounts for a specified customer

Method: GET	/accounts/{customerId}/customer	Get customer that owns the specified account

Example Failed Response: Status (404)

```
{
   “code” : 0,
    “message”: “error fetching customers accounts”
}
```

Example Successful Response: Status (200)

```
 {
        "code": 200,
        "message": "Success",
        “Data”:[
          {
            "id": 1,
            "type": "Savings",
            "nickname": "Leon’s Savings Account",
            "rewards": 37605,
            "balance": 19351,
            "customer_id": 10
          },
          {
            "id": 2,
            "type": "Checking",
            "nickname": "Leon’s Checking Account",
            "rewards": 37605,
            "balance": 19351,
            "customer_id": 10
          }
        ]
}
```

Customer: List all customers		
Method: GET	/customers	Get all customers

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “error fetching accounts”
}
```

```
 {
            "code": 200,
            "message": "Success",
            “Data”:
            [
              {
                "id": 1,
                "first_name": "Leon",
                "last_name": "Hunter",
                "address": {
                  "street_number": "902",
                  "street_name": "Walker Road",
                  "city": "Clearfield",
                  "state": "Pennsylvania",
                  "zip": "16830"
                }
              },
              {
                "id": 2,
                "first_name": "David",
                "last_name": "Ginzburgh",
                "address": {
                  "street_number": "40",
                  "street_name": "Pitt Bridge",
                  "city": "Branchland",
                  "state": "West Virginia",
                  "zip": "25506"
                }
              }
            ]
}
```

Customer: List all customers		
Method: GET	/customers/{id}	Get customer by id

Example Failed Response: Status (404)
{
  “code” : 404,
  “message”: “error fetching account”
}		

{
"code": 200,
"message": "Success",
“data”:
{
"id": 1,
"first_name": "Leon",
"last_name": "Hunter",
"address": {
"street_number": "902",
"street_name": "Walker Road",
"city": "Clearfield",
"state": "Pennsylvania",
"zip": "16830"
}
}
}		

Customer : Create a new customer account

Method: POST	/customers	Create a customer

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “Error ”
}
```

Example Successful Response: Status (200)

```
 {
         "code": 200,
         "message": "Customer account updated",
         “Data”:
         {
                  "id": 3,
                  "firstName": "Tariq",
                  "lastName": "Hook",
                  "address":
                  [
                  {
                                   "id": 5,
                                   "streetNumber": "123",
                                   "streetName": "Sesame Street",
                                   "city": "Wilmington",
                                   "state": "DE",
                                   "zipcode": "19801"
                  }
                  ]
         }

}
```

Customer: Get all accounts belonging to one customer by customer ID		
Method: GET	URI: /customers/{customerId}/accounts	Action: Get account by id

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “Error fetching customers accounts”
}
```

Example Successful Response: Status (200)

```
 {
         "code": 200,
         "message": "Customer account updated",
         “Data”:
                  [
                    {
                      "id": 1,
                      "type": "Savings",
                      "nickname": "Leon’s Saving Account",
                      "rewards": 37605,
                      "balance": 19351,
                      "customer_id": 10
                    },
                   {
                      "id": 32,
                      "type": "Checking",
                      "nickname": "Leon’s Checking Account",
                      "rewards": 37605,
                      "balance": 22,
                      "customer_id": 10
                    }
                  ]

}
```

Customer: Create a new account and assign it to a customer by the customer id

POST	/customers/{customerId}/accounts	Create an account

Example Failed Response: Status (404)
```
{
    “code” : 404,
     “message”: “error fetching creating customers account”
}
```

Example Successful Response: Status (201)

```
{
  "code": 201,
  "message": "Account created",
  "data": {
    "type": "Credit Card",
    "nickname": "string",
    "rewards": 0,
    "balance": 0,
    "customer_id": 10,
    "_id": 42
  }
}
```

Task 3: Create the restful endpoint for bills

Bill
- id: **Long** - status: **String** - payee: **String** - nickname: **String** - creation_date: **String** - payment_date: **String** - recurring_date: **Integer** - upcoming_payment_date: **String** - payment_amount: **Double** - account_id: **String**

- **id** - Bill unique identifier
- **Status** - An enumerated value representing the Status of the bill that serializes to a string
    - Pending
    - Cancelled
    - Completed
    - Recurring
- **payee** - The entity the bill will be paid to. Example: Verizon, ComEd
- **nickname** - A nickname for the bill to help you identify it
- **creation_date** - Date the bill was created
- **payment_date** - Date when bill is going to be paid or was paid. e.g. 4/31/2015
- **recurring_date** - Day of month bill will recur, e.g. 15th of every month. It can only be from 1 to 31
- **upcoming_payment_date** - Next bill payment date, calculated from recurring date. e.g. 1/3/2014
- **payment_amount** - Bill amount
- **account_id** - ID of account that this bill is associated with

Method: GET	URI: /accounts/{accountId}/bills	Action: Get all bills for a specific account
Method: GET	URI: /bills/{billId}	Action: Get bill by id
Method: GET	URI: /customers/{customerId}/bills	Action: Get all bills for customer
Method: POST	/accounts/{accountId}/bills	Create an bill
Method: PUT	/bills/{billId}	Update a specific existing bill
Method:DELETE	/bills/{billId}	Delete a specific existing bill

Bills: List all bills that are associated with the specified account		
Method: GET	URI: /accounts/{accountId}/bills	Action: Get all bills for a specific account

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “error fetching bills”
}
```

Example Successful Response: Status (200)

```
{
  “code”: 200,
  “data”:  [
   {
     "_id": 0201,
     "status": "pending",
     "payee": "COMMONWEALTHEDISONBILLINGDEPT",
     "nickname": "ComEd Electric Bill",
     "creation_date": "2017-07-25",
     "payment_date": "2017-07-25",
     "recurring_date": 25,
     "upcoming_payment_date": "2017-07-25",
     "Account_id": 3
  },
   {
     "_id": 0202,
     "status": "recurring",
     "payee": "VERIZONWIRELESSRECEIVABLES",
     "nickname": "Cellphone Bill",
     "creation_date": "2017-06-15",
     "payment_date": "2017-07-15",
     "recurring_date": 15,
     "upcoming_payment_date": "2017-08-15",
     "Account_id": 3
  }
]
}
```

Bills: List the bill with the specified bill ID		
Method: GET	URI: /bills/{billId}	Action: Get bill by id

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “error fetching bill with id:”
}
```

Example Successful Response: Status (200)

```
{
 “code”: 200,
 “data”:
  {
    "_id": 0201,
    "status": "pending",
    "payee": "COMMONWEALTHEDISONBILLINGDEPT",
    "nickname": "ComEd Electric Bill",
    "creation_date": "2017-07-25",
    "payment_date": "2017-07-25",
    "recurring_date": 25,
    "upcoming_payment_date": "2017-07-25",
    "Account_id": 3
 }
}
```

Bills: List all bills associated with the given customer ID		
Method: GET	URI: /customers/{customerId}/bills	Action: Get all bills for customer

Example Failed Response: Status (404)
```
{
   “code” : 404,
    “message”: “error fetching bills”
}
```

Example Successful Response: Status (200)

```
{
  “code”: 200,
  “data”:  [
   {
     "_id": 40201,
     "status": "pending",
     "payee": "COMMONWEALTHEDISONBILLINGDEPT",
     "nickname": "ComEd Electric Bill",
     "creation_date": "2017-07-25",
     "payment_date": "2017-07-25",
     "recurring_date": 25,
     "upcoming_payment_date": "2017-07-25",
     "Account_id": 1
  },
   {
     "_id": 40202,
     "status": "recurring",
     "payee": "ATT Communications Mobile Div",
     "nickname": "Cellphone Bill",
     "creation_date": "2017-06-15",
     "payment_date": "2017-07-15",
     "recurring_date": 15,
     "upcoming_payment_date": "2017-08-15",
     "Account_id": 2
  }
]
}
```

Bills: Create a new bill and assign it to an account by the account id		
Method: POST	/accounts/{accountId}/bills	Create an bill

Example Failed Response: Status (404)

```
{
   "code" : 404,
    "message": "Error creating bill: Account not found"
}
```

Example Successful Response: Status (201)

```
{
  "code": 201,
  "message": "Created bill and added it to the account",
  "data": {
    "status": "pending",
    "payee": "ComEd Receivables",
    "nickname": "Electric Bill",
    "payment_date": "2017-07-25",
    "recurring_date": 25,
    "payment_amount": 27.5,
    "creation_date": "2017-07-25",
    "account_id": 3,
    "_id": 40203
  }
}
```

Bills: Update an existing bill by ID		
Method: PUT	/bills/{billId}	Update a specific existing bill

Example Failed Response: Status (404)
{
    “code” : 404,
     “message”: “Bill ID does not exist”
}		

Example Successful Response: Status (202)
 	
{
  "code": 202,
  "message": "Accepted bill modification",
}		

Bills: Delete a specific bill by ID number		
Method:DELETE	/bills/{billId}	Delete a specific existing bill

Example Failed Response: Status (404)
{
    “code” : 404,
     “message”: “This id does not exist in bills”
}		

Example Successful Response: Status (204) <No content in response body>		

Task 4: Create the restful endpoint for Deposits

Deposit
- id: **Long** - type: **String** - transaction_date: **String** - status: **String** - payee_id: **Long** - medium: **String** - amount: **Double** - description: **String**

- **id** - Deposits unique identifier
- **type** - Type of transaction. Enumeration that serializes to a String
    - P2p
    - Deposit
    - Withdrawal
- **transaction_date** - Timestamp of deposit execution
- **status** - Status of the deposit. Enumeration that serializes to a String
    - Pending
    - Cancelled
    - Completed
- **payee_id** - id of account receiving the deposit
- **medium** - Type of deposit. Enumeration that serializes to a String
    - Balance
    - Rewards
- **amount** - Deposit amount
- **description** - Description of the deposit

Method: GET	URI: /accounts/{accountId}/deposits	Action: Get all deposits for a specific account
Method: GET	URI: /deposits/{depositId}	Action: Get deposit by id
Method: POST	/accounts/{accountId}/deposits	Create an deposit
Method: PUT	/deposits/{depositId}	Update a specific existing deposit
Method:DELETE	/deposits/{depositId}	Delete a specific existing deposit

Deposits: List all deposits that are associated with the specified account		
Method: GET	URI: /accounts/{accountId}/deposits	Action: Get all deposits for a specific account

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “Account not found”
}
```

Example Successful Response: Status (200)

```
{
  “code”: 200,
  “data”:  [
   {
     "_id": 12345,
     "medium": "balance",
     "transaction_date": "2017-07-25",
     "amount": 0.01,
     "description": "penny saved",
     "status": "executed",
     "payee_id": 3,
     "type": "deposit"
  },
       {
     "_id": 12346,
     "medium": "rewards",
     "transaction_date": "2017-07-25",
     "amount": 60.00,
     "description": "cashback bonus",
     "status": "executed",
     "payee_id": 3,
     "type": "deposit"
   }
]
}
```

Deposits: List the deposit with the specified ID		
Method: GET	URI: /deposits/{depositId}	Action: Get deposit by id

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “error fetching deposit with id:”
}
```

Example Successful Response: Status (200)

```
{
 “code”: 200,
 “data”:
  {
    "_id": 12345,
    "medium": "balance",
    "transaction_date": "2017-07-25",
    "amount": 0.01,
    "description": "penny saved",
    "status": "executed",
    "payee_id": 3,
    "type": "deposit"
  }
}
```

Deposits: Create a new deposit and credit the specified account with the given amount

Method: POST	/accounts/{accountId}/deposits	Create an deposit

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “Error creating deposit: Account not found”
}
```

Example Successful Response: Status (201)

```
{
  "message": "Created deposit and added it to the account",
  "code": 201,
  "data": {
    "medium": "rewards",
    "transaction_date": "2017-07-25",
    "amount": 75,
    "description": "Cashback Bonus",
    "status": "pending",
    "payee_id": 2,
    "type": "deposit",
    "_id": 12347
  }
}
```

Deposits: Update an existing deposit by ID

Method: PUT	/deposits/{depositId}	Update a specific existing deposit

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “Deposit ID does not exist”
}
```

Example Successful Response: Status (202)

```
{
  "code": 202,
  "message": "Accepted deposit modification",
}
```

Deposits: Delete a specific deposit by ID number		
Method:DELETE	/deposits/{depositId}	Delete a specific existing deposit

Example Failed Response: Status (404)
{
    “code” : 404,
     “message”: “This id does not exist in deposits”
}		

Example Successful Response: Status (204) <No content in response body>		

Task 5: Create the restful endpoint for Withdrawals

Deposit
- id: **Long** - type: **String** - transaction_date: **String** - status: **String** - payer_id: **Long** - medium: **String** - amount: **Double** - description: **String**

- **id** - Withdrawals unique identifier
- **type** - Type of transaction. Enumeration that serializes to a String
    - P2p
    - Deposit
    - Withdrawal
- **transaction_date** - Timestamp of deposit execution
- **status** - Status of the deposit. Enumeration that serializes to a String
    - Pending
    - Cancelled
    - Completed
- **payer_id** - id of account receiving the deposit
- **medium** - Type of withdrawal. Enumeration that serializes to a String
    - Balance
    - Rewards
- **amount** - Withdrawal amount
- **description** - Description of the withdrawal

Method: GET	URI: /accounts/{accountId}/withrdawals	Action: Get all withdrawals for a specific account
Method: GET	URI: /withdrawals/{withdrawalId}	Action: Get withdrawal by id
Method: POST	/accounts/{accountId}/withdrawals	Create an withdrawal
Method: PUT	/withdrawals/{withdrawalId}	Update a specific existing withdrawal
Method:DELETE	/withdrawals/{withdrawalId}	Delete a specific existing withdrawal

Withdrawals: List all withdrawals that are associated with the specified account

Method: GET	URI: /accounts/{accountId}/withrdawals	Action: Get all withdrawals for a specific account

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “Account not found”
}
```

Example Successful Response: Status (200)

```
{
  “code”: 200,
  “data”:  [
   {
    "_id": 13345,
    "medium": "balance",
    "transaction_date": "2017-07-25",
    "amount": 0.01,
    "description": "penny spent",
    "status": "executed",
    "payer_id": 3,
    "type": "withdrawal"
  },
      {
    "_id": 13346,
    "medium": "balance",
    "transaction_date": "2017-07-25",
    "amount": 100.00,
    "description": "ATM Withdrawal",
    "status": "executed",
    "payer_id": 3,
    "type": "withdrawal"
   }
]
}
```

Withdrawals: List the withdrawal with the specified ID

Method: GET	URI: /withdrawals/{withdrawalId}	Action: Get withdrawal by id

Example Failed Response: Status (404)

```
{
   “code” : 404,
    “message”: “error fetching withdrawal with id:”
}
```

Example Successful Response: Status (200)

```
{
  “code”: 200,
  “data”:
   {
     "_id": 13345,
     "medium": "balance",
     "transaction_date": "2017-07-25",
     "amount": 0.01,
     "description": "penny spent",
     "status": "executed",
     "payer_id": 3,
     "type": "withdrawal"
  }
}
```

Withdrawals: Create a new withdrawal and deduct the specified account by the given amount

Method: POST	/accounts/{accountId}/withdrawals	Create an withdrawal

Example Failed Response: Status (404)
```
{
   “code” : 404,
    “message”: “Error creating withdrawal: Account not found”
}
```

Example Successful Response: Status (201)

```
{
  "message": "Created withdrawal and deducted it from the account",
  "code": 201,
  "data": {
    "medium": "rewards",
    "transaction_date": "2017-07-25",
    "amount": 100,
    "description": "ATM Withdrawal",
    "status": "pending",
    "payer_id": 2,
    "type": "withdrawal",
    "_id": 13347
  }
}
```

Withdrawals: Update an existing deposit by ID

Method: PUT	/withdrawals/{withdrawalId}	Update a specific existing withdrawal

Example Failed Response: Status (404)
```
{
   “code” : 404,
    “message”: “Withdrawal ID does not exist”
}
```

Example Successful Response: Status (202)

```
{
  "code": 202,
  "message": "Accepted withdrawal modification",
}
```

Withdrawals: Delete a specific withdrawal by ID number		
Method:DELETE	/withdrawals/{withdrawalId}	Delete a specific existing withdrawal

Example Failed Response: Status (404)
{
    “code” : 404,
     “message”: “This id does not exist in withdrawals”
}		

Example Successful Response: Status (204) <No content in response body>		