**Customer Details Assessment:**

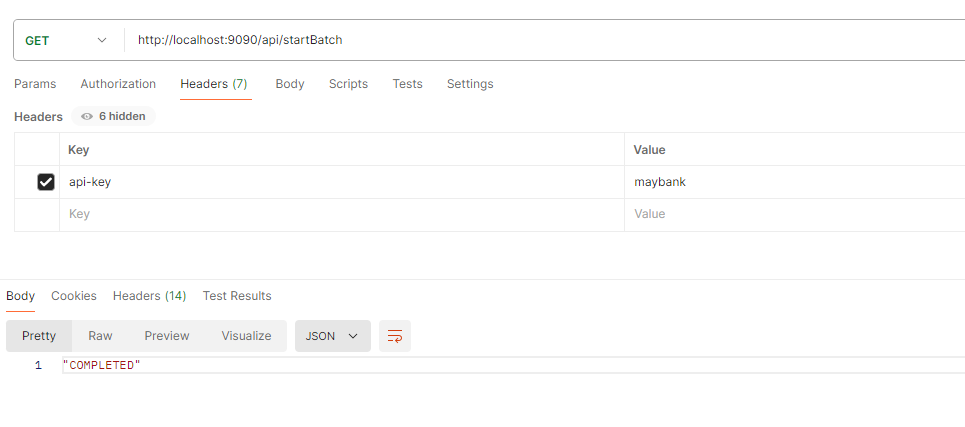
* Developed using the **Spring boot(3.3.0)**, **Java17** & **Angular18 and MySQL Database**.
* Source Code available in GitHub [phani238/Customer-Details (github.com)](https://github.com/phani238/Customer-Details)

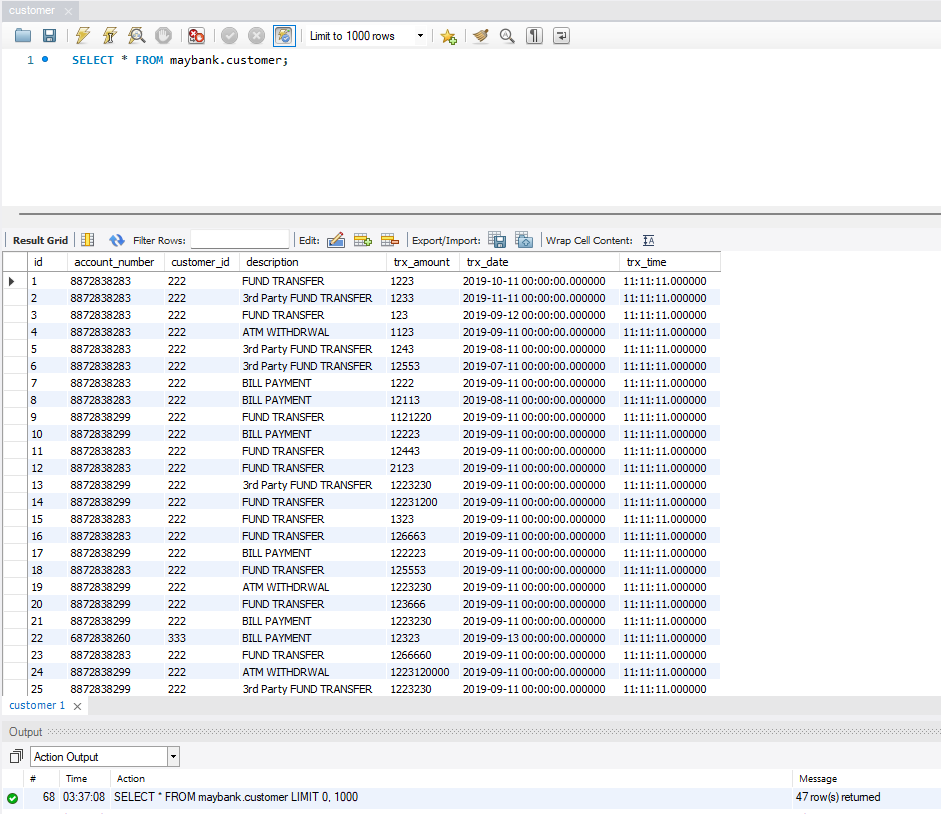
**Task 1: Batch Job for Consuming Text File**

We used a Spring Batch Job to consume the text file, with the following setup:

1. **Batch Job Architecture:**
   1. The job structure follows the sequence: Job → Step → 1. Reader, 2. Processor, 3. Writer.
2. **Configuration**:
   1. **The `BatchConfig.java`** file manages the entire batch job lifecycle.
3. **API Integration**:
   1. The `**CustomerDetailsController.java`** enables API calls for the batch job.
   2. URL: `http://localhost:9090/api/startBatch`
   3. Spring Security is implemented. If security headers are not provided, API calls will be blocked.
4. The following header parameters are required for API calls
   1. Key: `api-key`
   2. Value: `maybank`
5. **Database Integration**:
   1. All values are successfully stored in the database.

**Note:** The batch file is located at “src/main/resources/static/dataSource.txt”.  
  
Screenshots





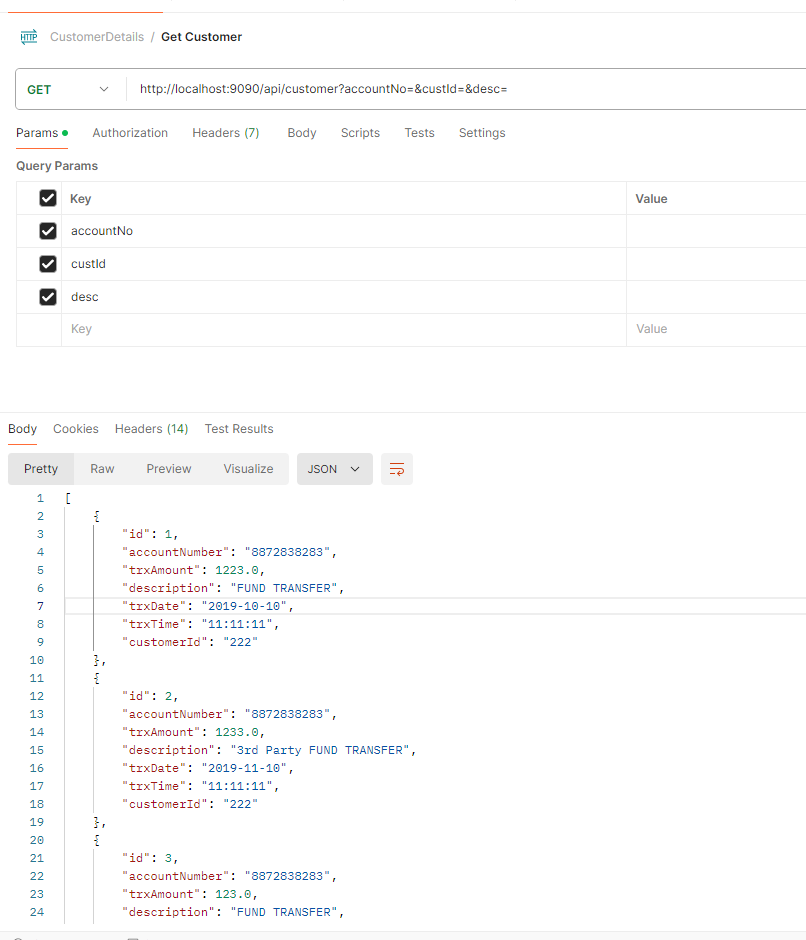
**Task 2: Retrieve and Update API’s**

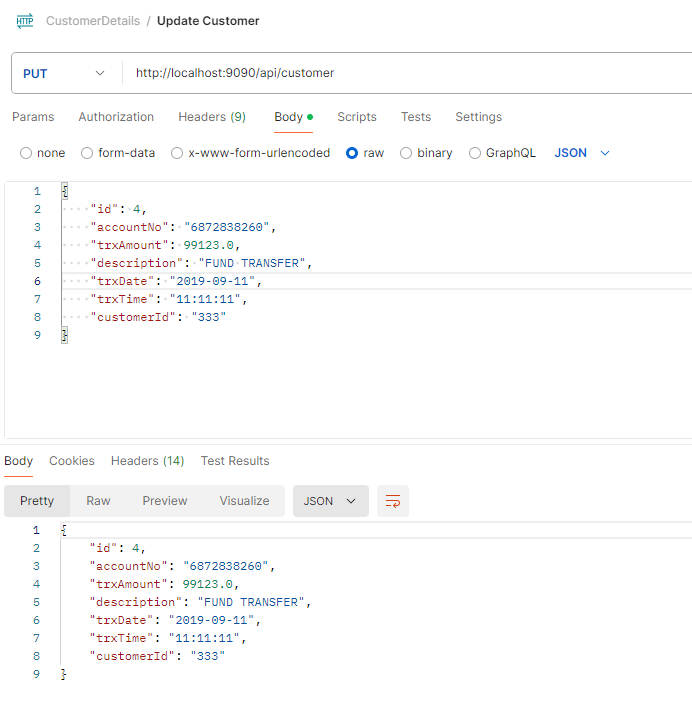
1. **Framework and Security**:
   1. We used the Spring Boot framework and Spring Security for API calls.
2. **API Flow**:
   1. The flow for all APIs is as follows: Controller → Service → Repository → Entity.
3. **Controller:**
   1. ` CustomerDetailsController.java` contains the information about the API endpoints.
4. **Service:**
   1. ` CustomerDetailsService.java` performs the business logic and connects to the repositories, which have all JPA connections by default.
5. **Repository:**
   1. ` CustomerDetailsRepository.java` is an interface where we created custom functions for retrieving data based on search criteria.
6. **Entity:**
   1. `Customer.java` is the entity class used to create the database table.
7. **APIs Created**:
   1. `getCusotmersSearch` using the **@GetMapping**
   2. `updateCustomer` using the **@PutMapping**
8. The following header parameters are required for API calls

**Note:** The following header parameters are required for API calls

* 1. Key: `api-key`
  2. Value: `maybank`

Screenshots





**Task 3: Developed front-end screens for the API’s**

1. **Framework:**
   1. Used Angular 18 for developing the screens.
2. **Components Created**:
   1. `CustomerSearchComponent.ts`
   2. `EditCustomerComponent.ts`
3. **CustomerSearchComponent**:
   1. Handles search criteria and displays results in a tabular format.
4. **EditCustomerComponent**:
   1. A pop-up component that appears when the user wants to update a specific record.

**Note:** Unable to complete pagination for the screens within the given timeframe.

Screenshots