A PROJECT REPORT ON

Banking Management System

Submitted by

Mr. Prashanth S

Mr. Kushal sharma M.

Ms. Sreeja

Ms. Sudha R

Ms. Suma M R

.

Batch No. 6773

Under the Guidence of Trainer

Mrs. Indrakka Mali Mam

Sl No	Topic Name
1	Introduction
2	Objective
3	Software Requirements
4	System overview and snapshots
5	Conclusion

Introduction

The project "Banking Management System " is developed using spring boot framework, which mainly focuses on basic operations of Online Banking. Like Withdrawn Amount, Deposit Amount, Transfer Amount, Insert customer Details , Delete cutomer Details and getting records of Customer Details based AccountId.

Online Banking allows customer of financial institution to conduct financial Transaction on secure website Operated by the institution ,Which can be a retail or virtiual bank,credit union or building society.

Online Banking is practice of making bank Transaction or paying bills via the interne.people no longer leave the house to shop,Communicate,or even do their Banking.Online banking allows a customer to make deposite,withdrawls,and pay billsall with the by click.

Account Module:

- In the Account module the Bank can perform :
- Fetch all details of Account records.
- Fetch Account record by Particular Account id.
- Fetch customer record by Particular Account Id.
- .In the Account module Customer can Withdraw,Deposite ,Transfer Amount.
- In this Module Customer can delet their Account .

Customer Module:

- In the Course Module :
- We Can Fetch all details of Customer records.
- In This module Customer Can Update Their Records like PhoneNo,password,etc.
- We can get the customer details and delete the record by using customer id.

Objectives:

- The Main objective of study are
- To Understand the concept of Online Banking.
- To analyze the importance,function ,advantages and limitation of Online banking.
- To analyze the present e-banking scenario concerned with Atm,Internet banking,mobile Banking ,credit-debit card,Fund Transfer and other e-banking service.
- The Main objective is how e-banking provide efficient service to User.
- It provides “better and efficient” service”.
- Faster way to get information about the Customer And Account.
- Provide facility for proper monitoring and reduce paper work.
- All details will be available on a click.

System Overview:

- The Bank Management System will be automated the traditional system.
- There is no need to use paper and pen.
- Checking a Customer and Account details is very easy.
- Adding new Customer record is very easy.
- Deleting or updating a record of a particular Customer and Account is simple

Requirements:

Software Requirement:

- Database: MySQL
- API- Spring Data JPA, spring web, spring security
- Tools: Postman, IDE-Spring Tool Suit4
- Coding language-Java 1.8:

Hardware Requirements:

- RAM: 4GB or above
- Processor: 64bit
- Memory: 512 MB
- Disk Space: 100GB
- Windows:Windows 10 Or above

Spring Tool Suit : -

STS is an Eclipse-based development environment that is customized

for the development of spring applications.

It provides a ready-to-use environment to implement, debug, run and deploy your applications.

Postman :-

Postman is a standalone software testing API (Application ProgrammingInterface)

platform to build, test, design, modify, and document APIs. It is a simple Graphic User

Interface for sending and viewing HTTP requests and responses.

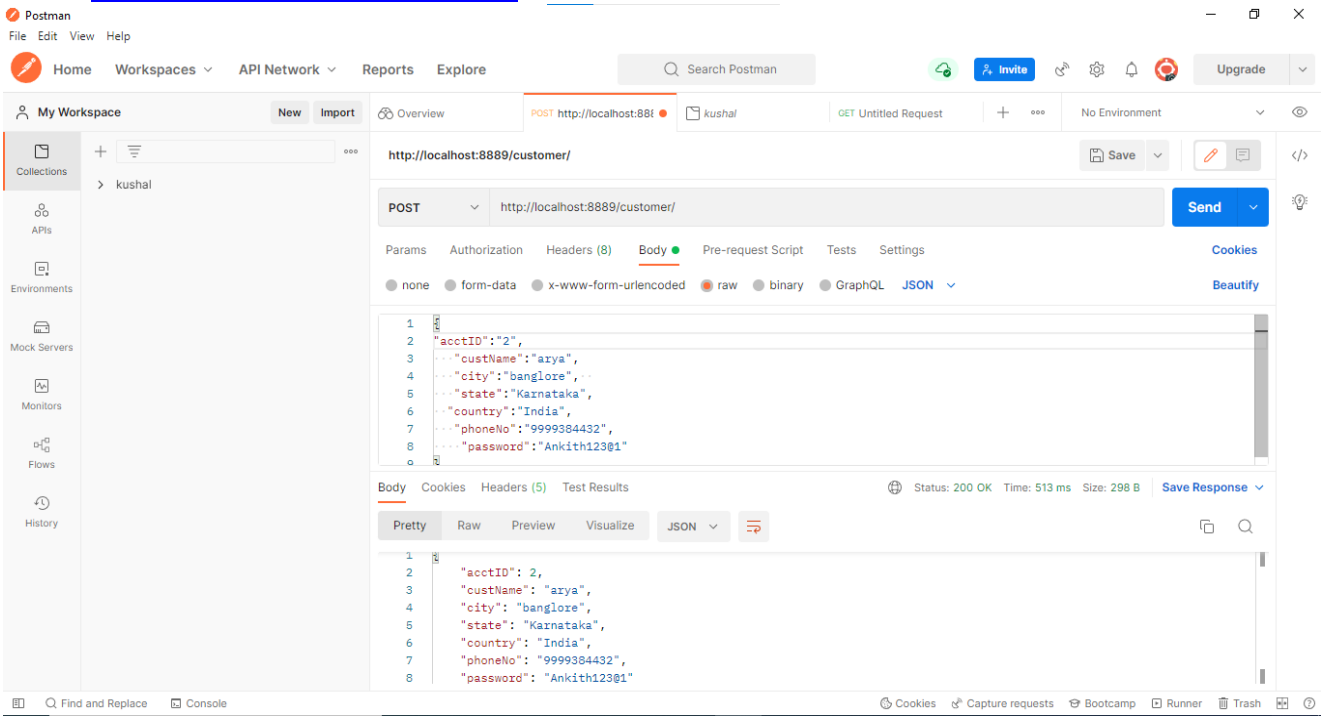
MySQL:

- MySQL is a relational database management system
- MySQL is open-source
- MySQL is free
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, scalable, and easy to use
- MySQL is cross-platform

Screenshots

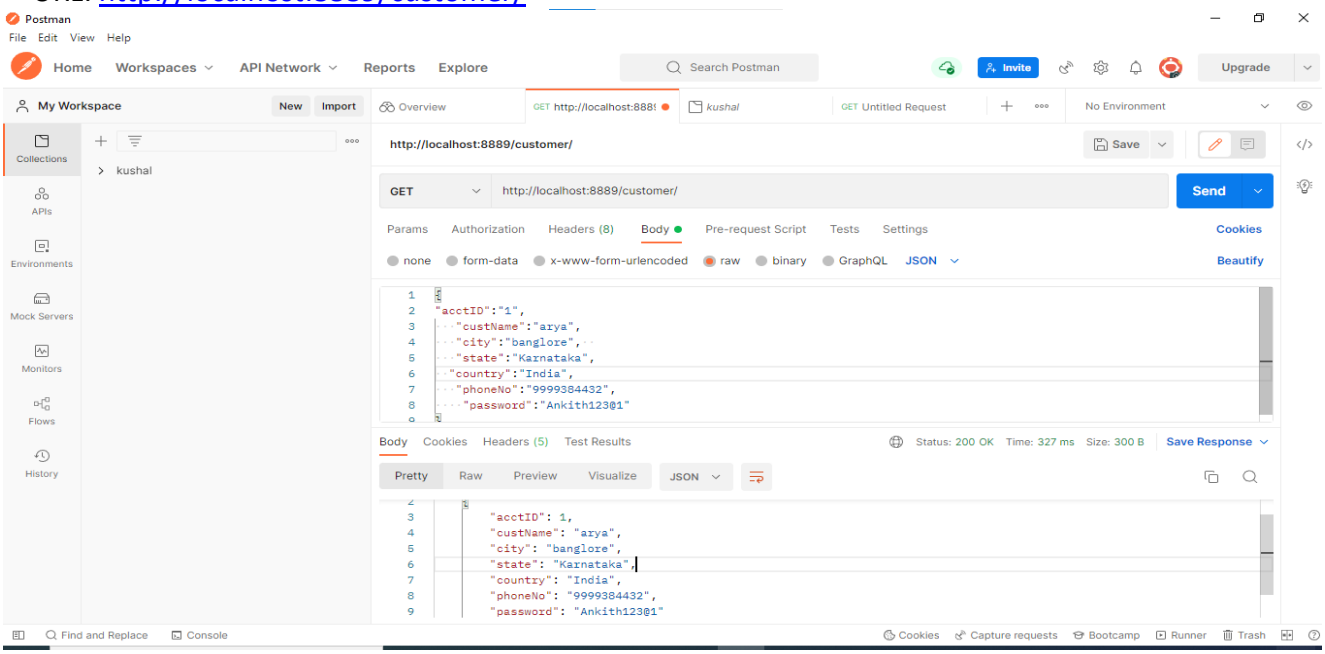
Step 1: we can Inserting the customer Details using post method.

URL: http://localhost:8889/customer/



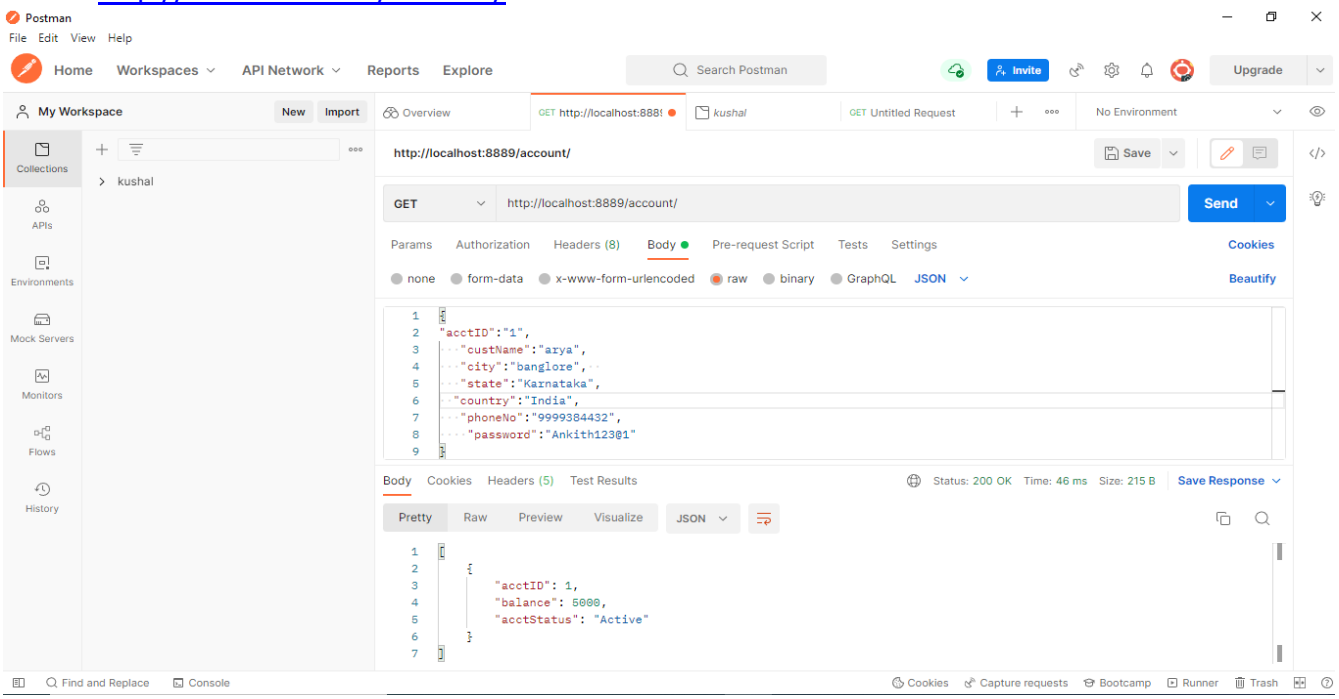
Step 2: Display All details of Customer Using Get Method.

URL: http://localhost:8889/customer/



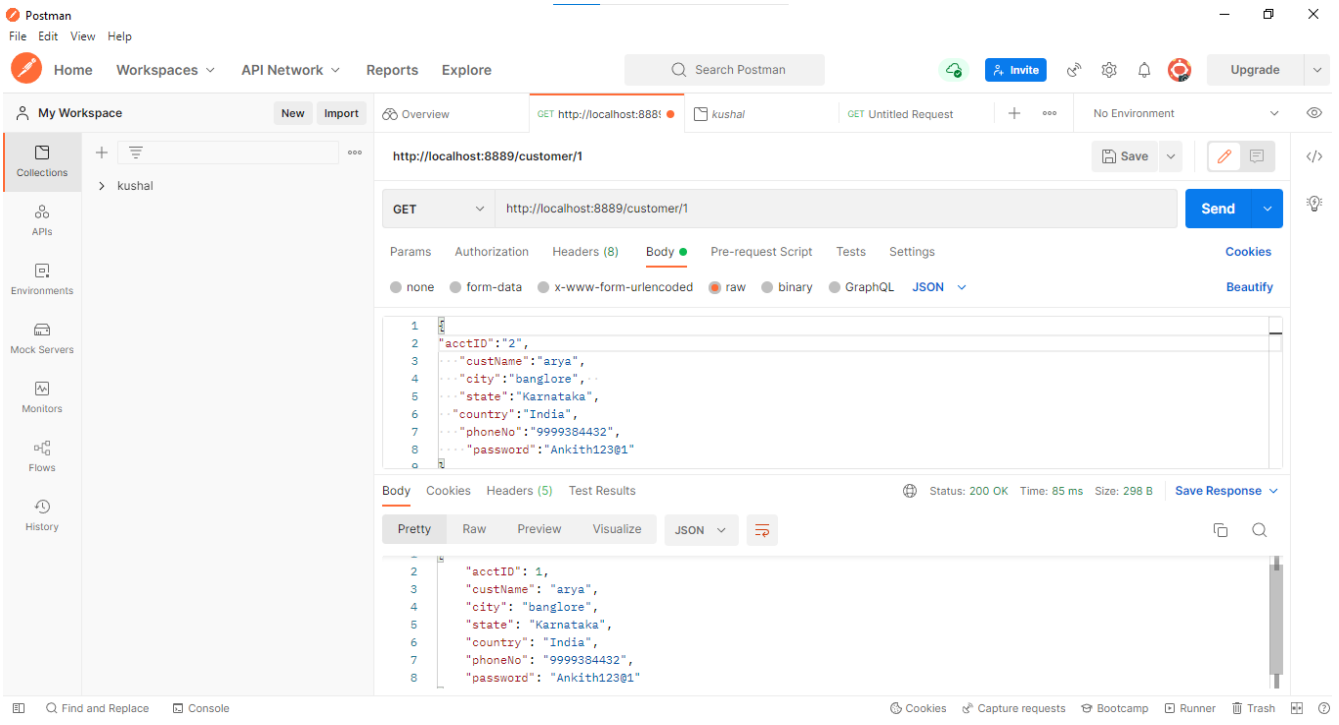
Step 3: Display All details of Account.

URL : http://localhost:8889/account/



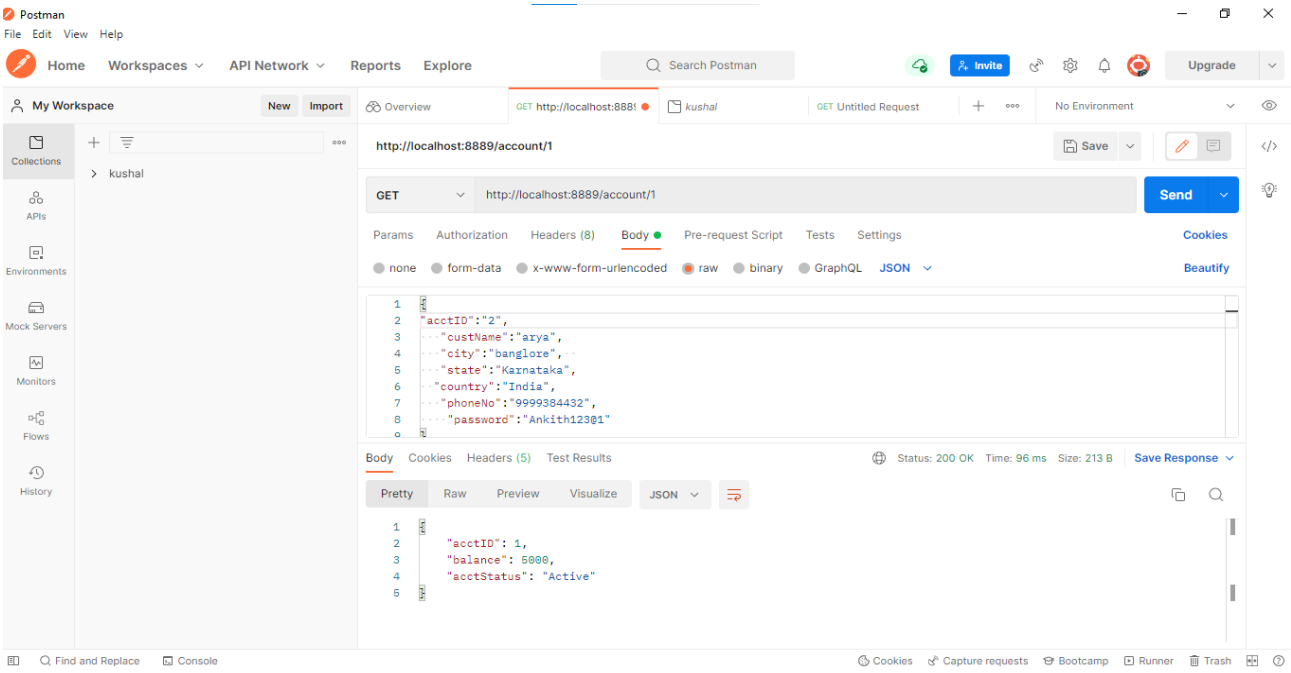
Step 4: Get Particular details of customer based on id.

URL : http://localhost:8889/customer/1



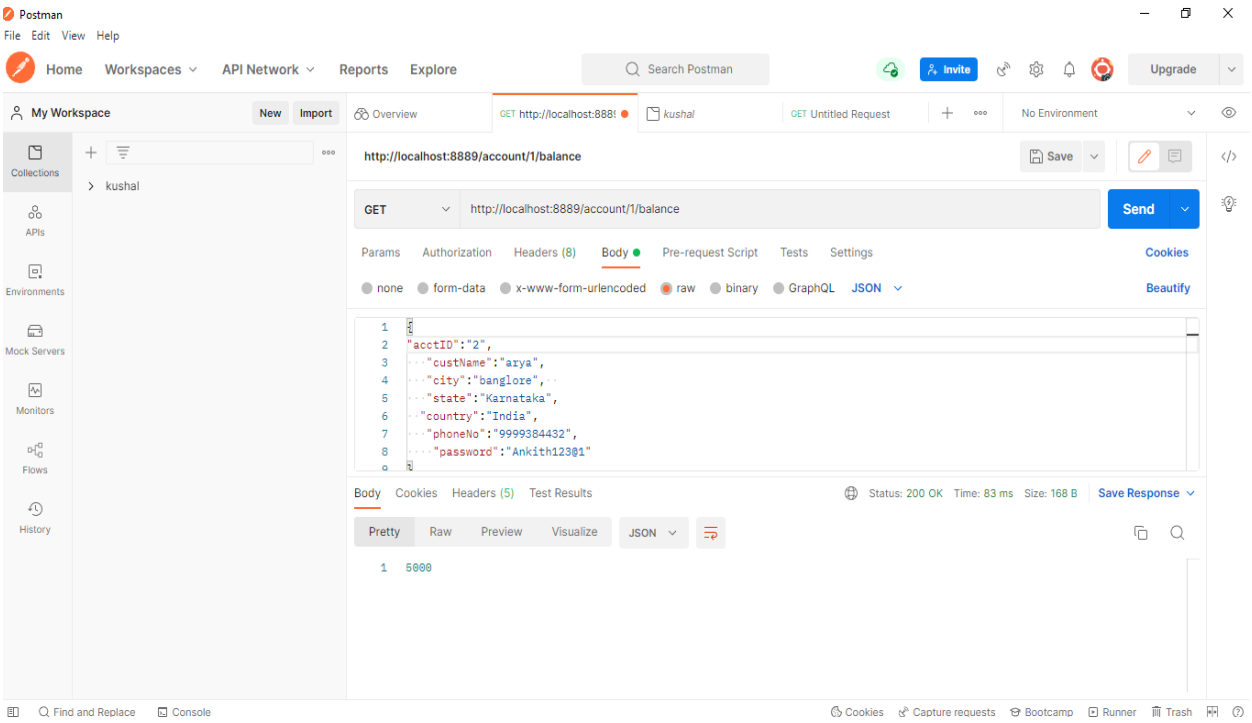
Step 5:Get Particular details of customer Account based on id Using Get Method.

URL : http://localhost:8889/account/1



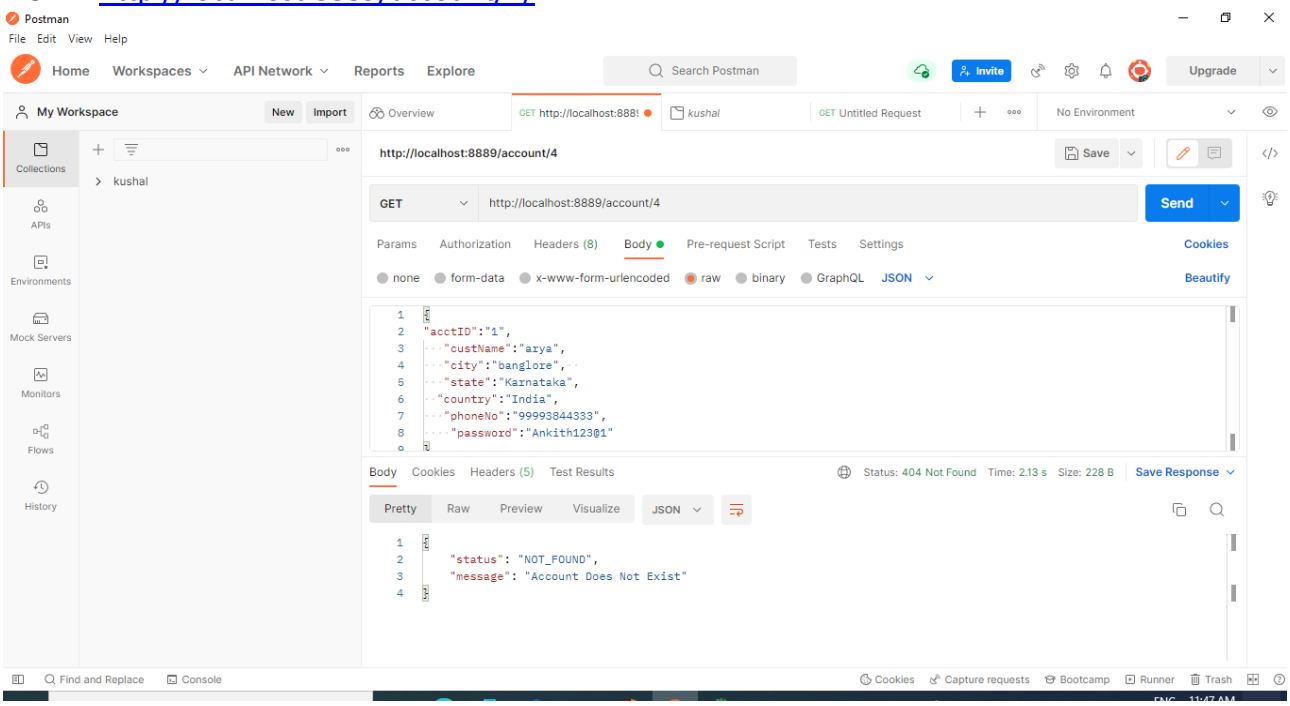
Step 6: GET the balance of customer based on id

URL :http://localhost:8889/account/1/balance



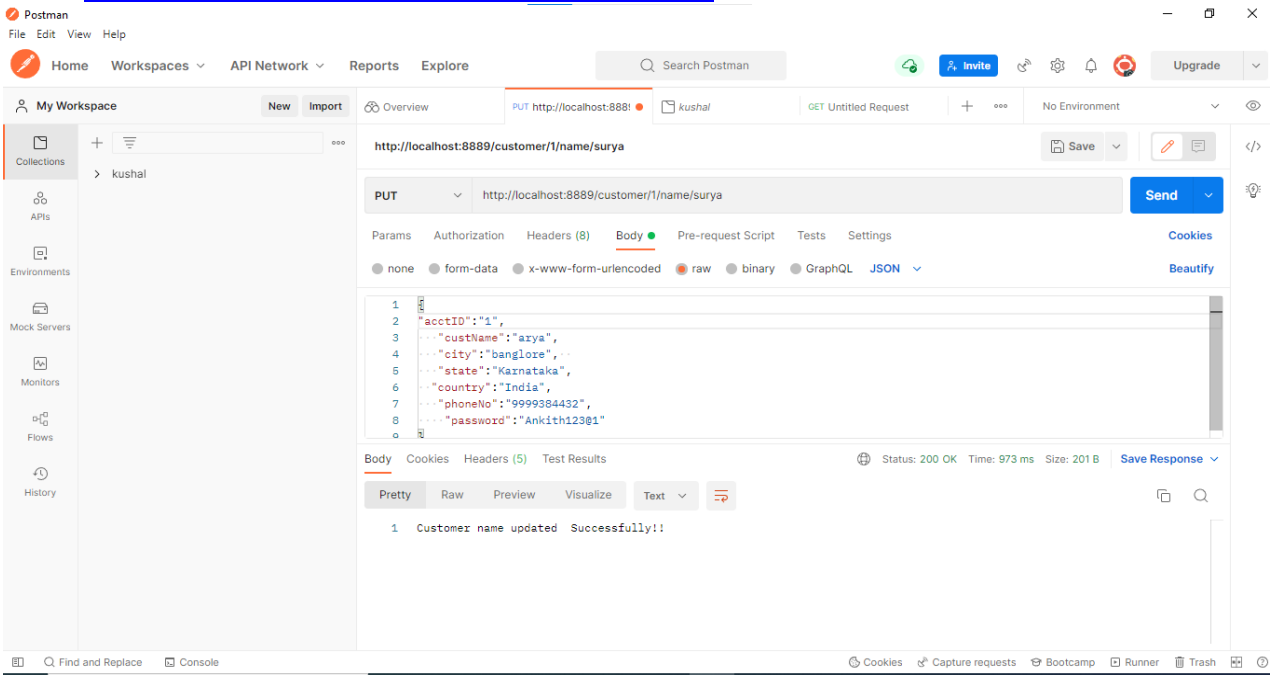
Step 7: If Customer doesn,t have Account Or Their AccountID Then it shows exception like "Account does not Found"

URL : http://localhost:8889/account/4/



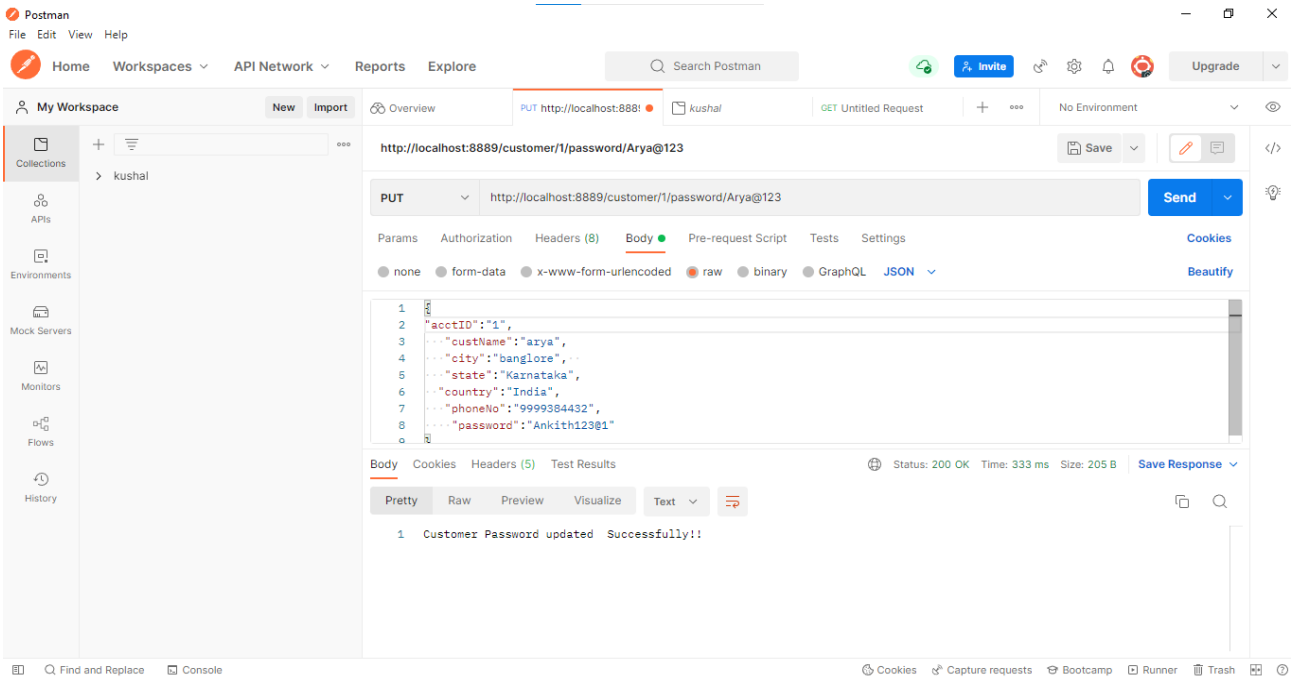
Step 8: If Customer wants to Change His Details like Name .We can update customer Details Based on Id using PUT Method.

URL : http://localhost:8889/customer/2/name/Surya



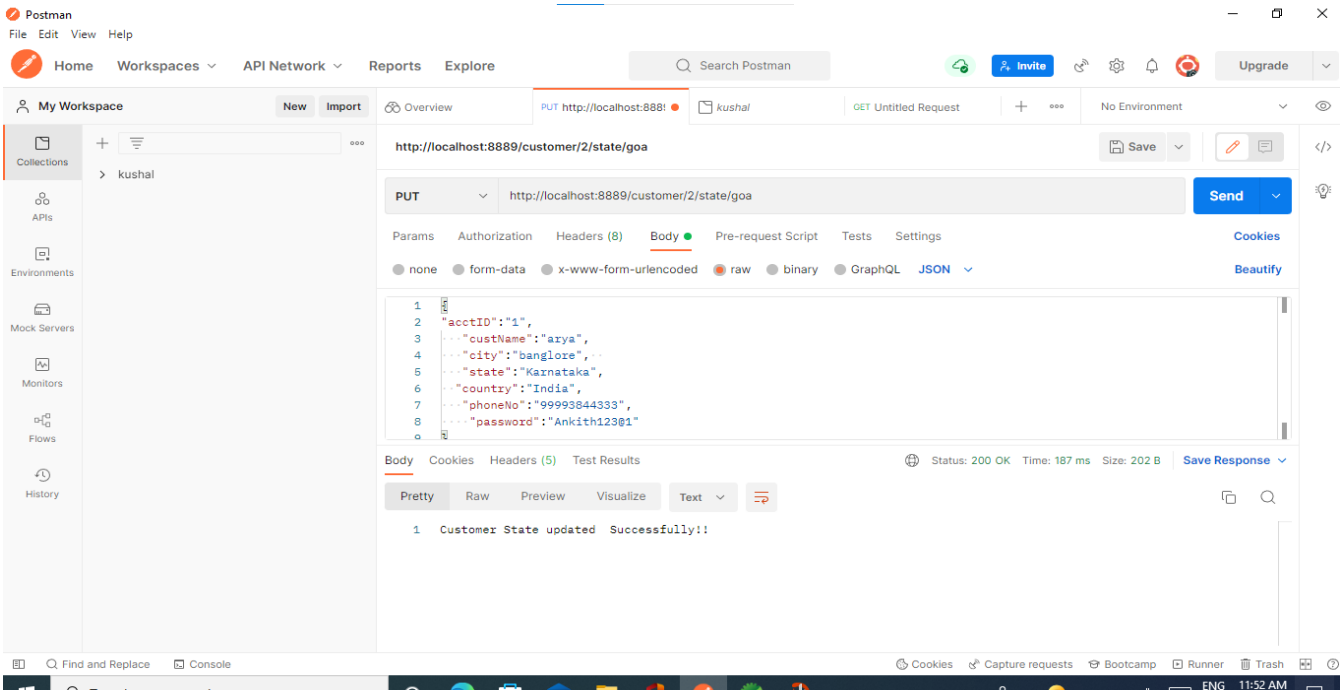
Step 9: IF Customer wants to Change His City based on id Using PUT Method.

URL : http://localhost:8889/customer/4/city/Bangalore



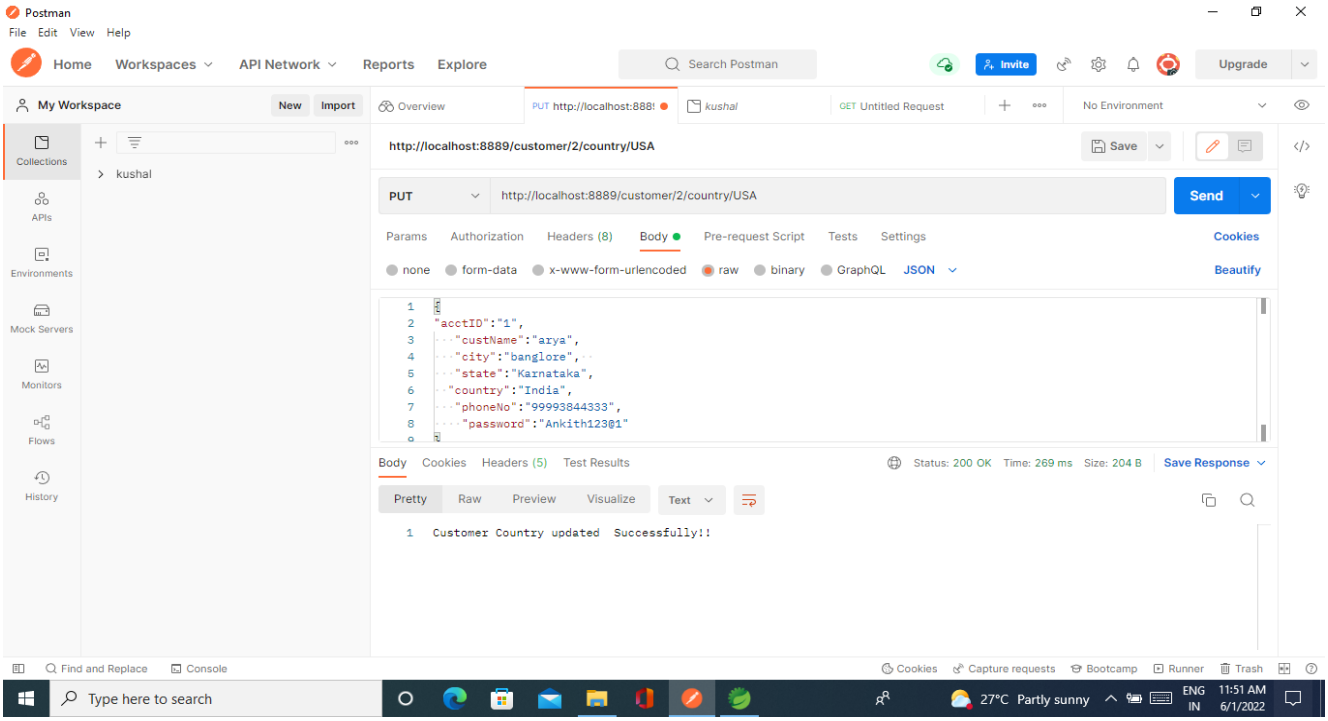
Step 10: IF Customer wants to Change His State based on id Using PUT Method.

URL : http://localhost:8889/customer/2/city/AndraPradesh



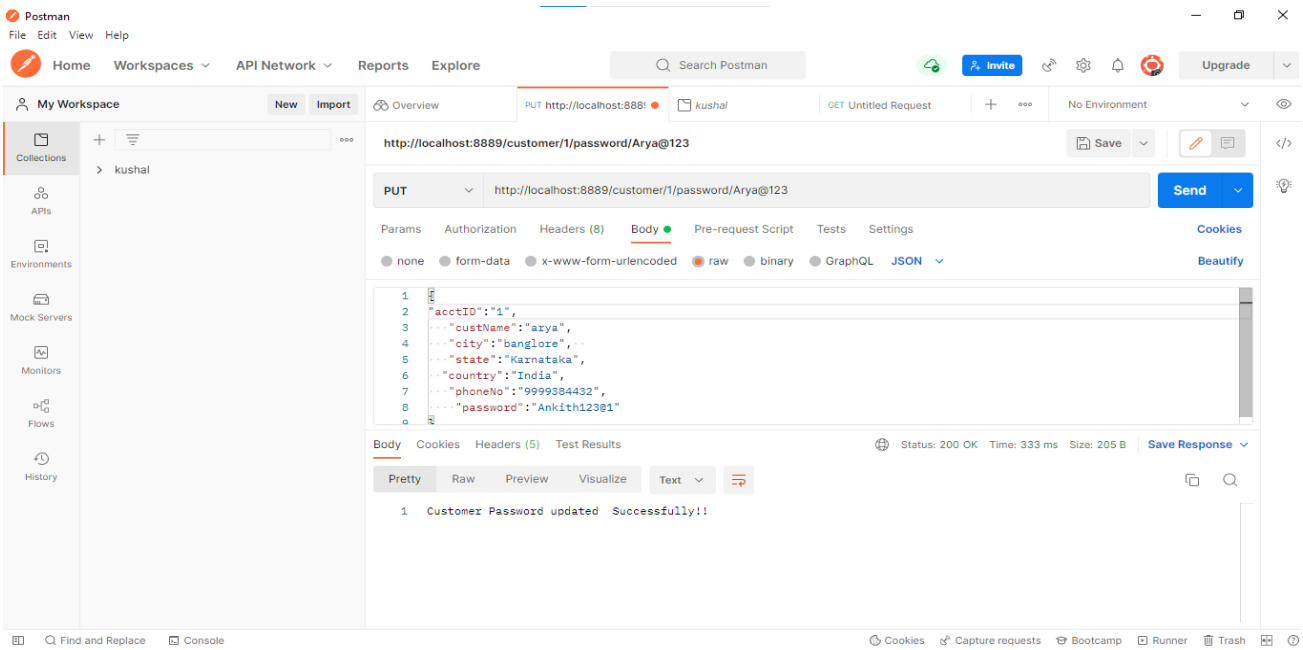
Step 11: IF Customer wants to Change His Country based on id Using PUT Method.

URL : http://localhost:8889/customer/2/country/USA



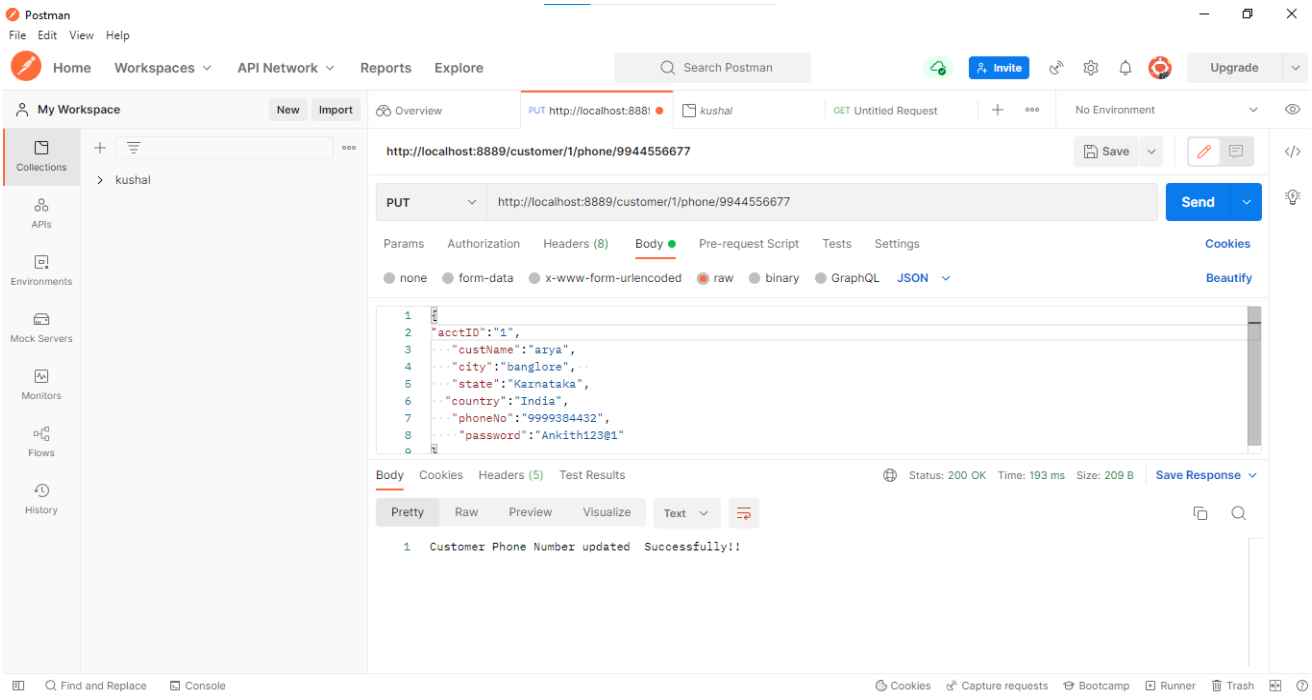
Step 12: IF Customer wants to Change His Password Details.

URL : http://localhost:8889/customer/4/password/prash123



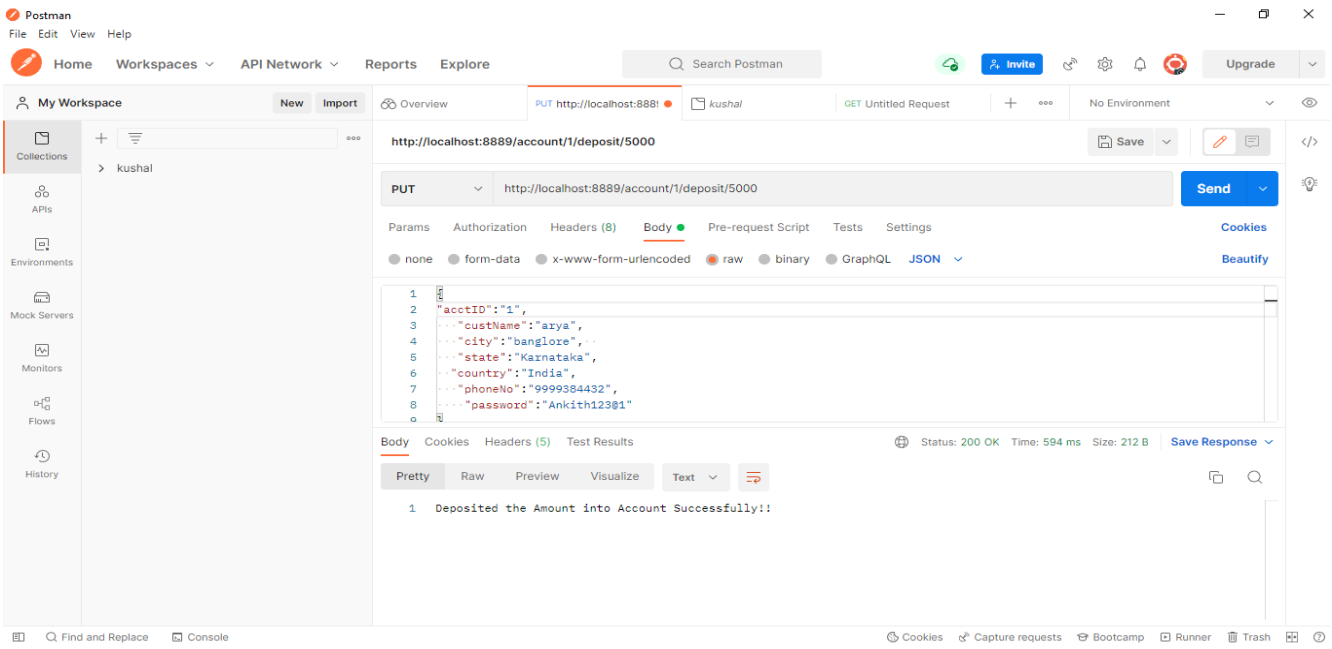
Step 13: If Customer wants to Change His Phone Number details.

URL :http://localhost:8889/customer/4/phone/9980765432



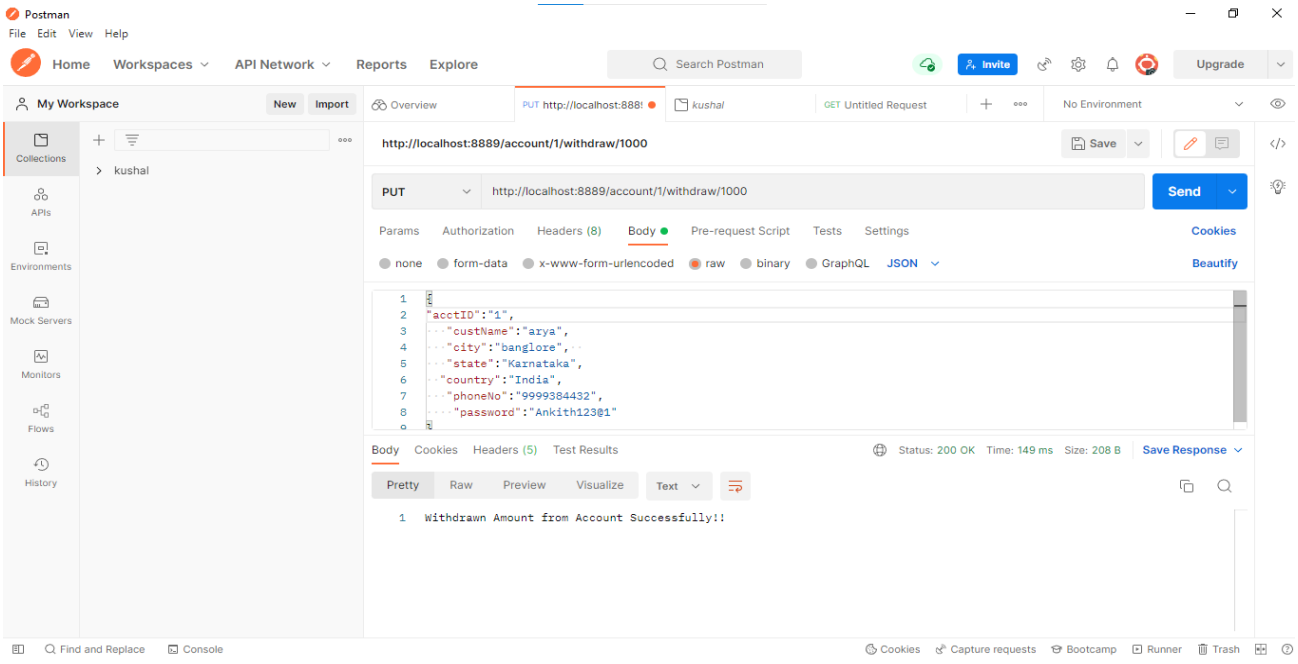
Step 14: Account Holder Can to Deposit there some money into his Account Based on Account Id Using Put Method.

URL :http://localhost:8889/account/3/deposit/60000



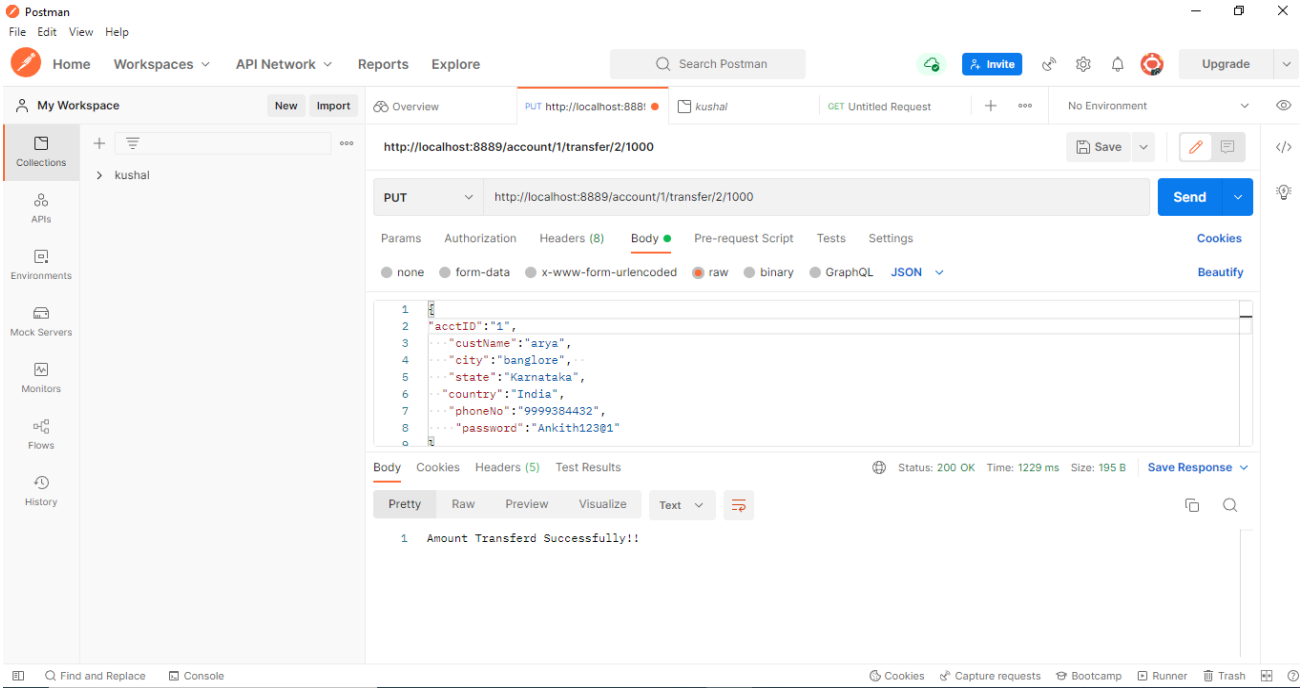
Step 15:Account Holder Can Withdraw there some money into his Account Based onAccount Id.

URL :http://localhost:8889/account/1/withdraw/60000



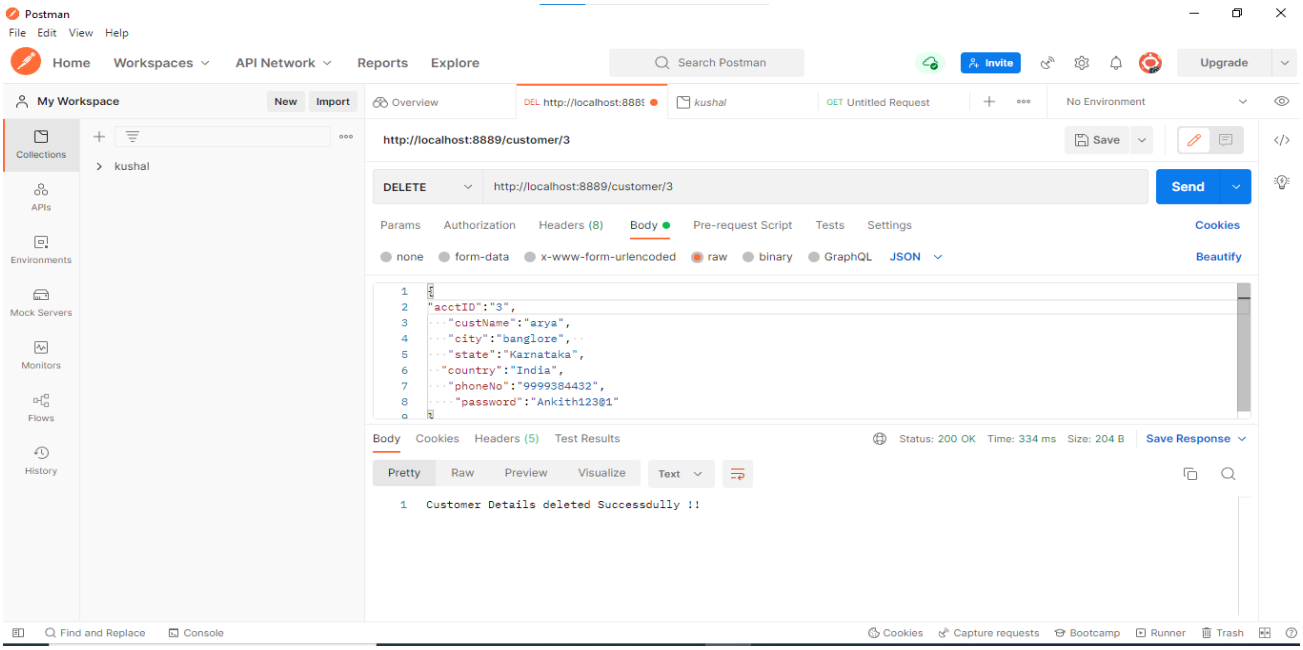
Step 16: Account Holder Can Transfer money from one Account into Another Account Based on ID Using Put Method.

URL :http://localhost:8889/account/1/transfer/3/10000



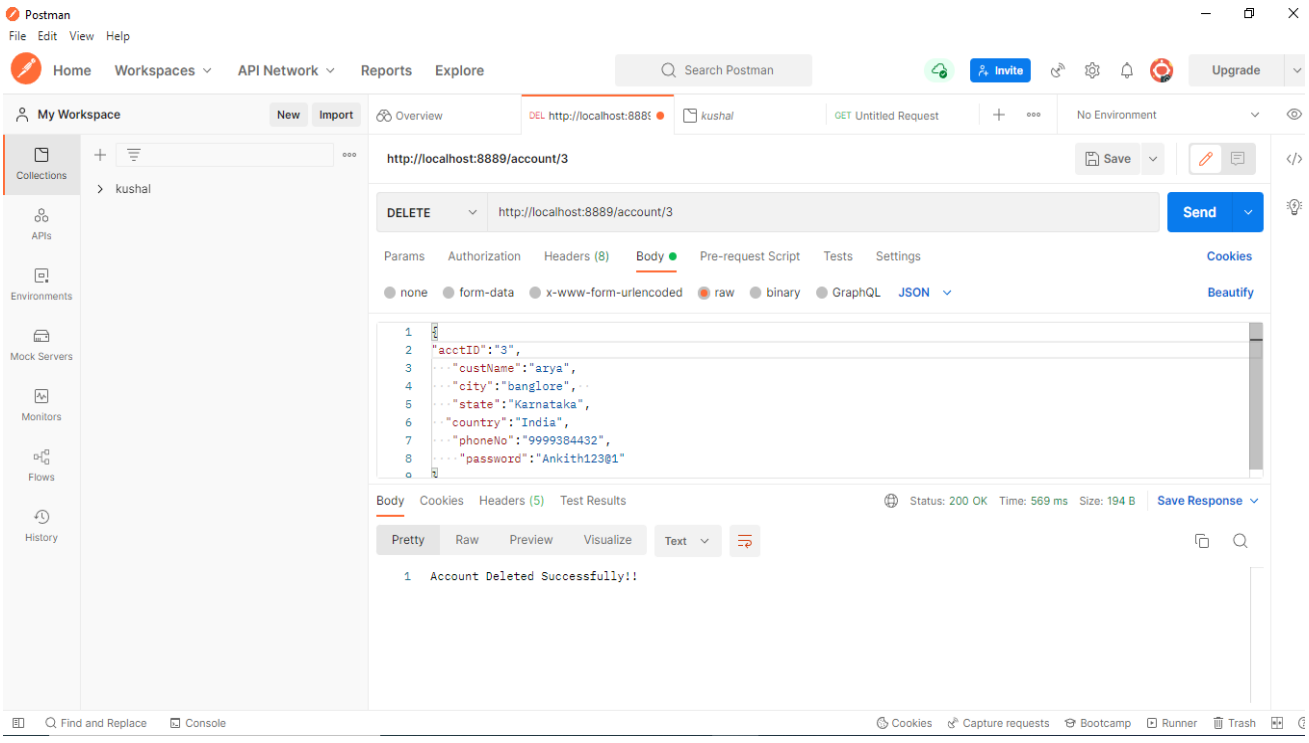
Step 17:Bank can delete thier customer details based on id Using Delete Method.

URL : http://localhost:8889/customer/2



Step 18:Bank delete their account details based on id Using Delete Method.

URL : http://localhost:8889/account/2



Annotations:

1. @Entity:

The @Entity annotation specifies that the class is an entity and is mapped to a database table. The @Table annotation specifies the name of the database table to be used for mapping

2.@ID

The @Id annotation **specifies the primary key of an entity** and th @GeneratedValue provides for the specification of generation strategies for the values of primary keys.

3. @Generated Value:

Marking a field with the @GeneratedValue annotation specifies that a value will be automatically generated for that field. This is primarily intended for primary key fields but Object DB also supports this annotation for non-key numeric persistent fields as well

4.@Rest Controller.

@RestController is used for making restful web services with the help of the @RestController annotation.This annotation is used at the class level and allows the class to handle the requests. The RestController allows to handle all REST APIs such as GET, POST, Delete, PUT requests.

5. @Autowired:

The spring framework enables automatic dependency injection. In other words, by declaring all the bean dependencies in a spring configuration file , Spring container can autowire relationships between collaborating beans. This is called spring bean autowiring.

6. @GetMapping:

The @GetMapping annotation is a specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET). It maps the HTTP GET requests on the specific handler method. It is used to create a web service endpoint that fetches

7.@putMapping:

It maps the **HTTP PUT** requests on the specific handler method. It is used to create a web service endpoint that **creates** or **updates**

8.@Delete Mapping.

@DeleteMapping is a specialized version that acts as a shortcut for Request Mapping The DELETE HTTP method is used to delete the resource and @DeleteMapping annotation maps the HTTP DELETE requests onto specific handler methods of a Spring controller. You can use this annotation only at the method level. You can use only the @RequestMapping annotation at the class level.

9. @PostMapping:

The @PostMapping is specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.POST). The @PostMapping annotated methods in the @Controller annotated classes handle the HTTP POST requests matched with given URI expression.

10.@Repository:

@Repository's job is to catch persistence-specific exceptions and re-throw them as one of spring's unified unchecked exceptions.

11.@Query

The @Query annotation can only be used to annotate repository interface methods. The call of the annotated methods will trigger the execution of the statement found in it, and their usage is pretty straightforward. The @Query annotation supports both native SQL and JPQL.

12.@Transactional:

The @Transactional annotation is that specifies the semantics of the transactions on a method.
We have two ways to rollback a transaction: declarative and programmatic.
The @Transactional annotation makes use of the
attributes noRollbackFor or noRollbackForClassName to avoid rollback on listed exceptions.
The default rollback behavior in the declarative approach will rollback on runtime exceptions.

13.@Modifying:

The @Modifying annotation is used to enhance the @query annotation so that we can execute not only select queries,but also Insert,Update,Delete,and even DDl queries.

14. @Service:

We mark beans with @Service to indicate that they're holding the business logic. Besides being used in the service layer, there isn't any other special use for this annotation.

15.@RequestBody

.It is used to bind HTTP request with an object in a method parameter.

16. @BeanAnnotation:

Spring @Bean annotation tells that a method produces a bean to be managed by the spring container. It is a method-level annotation. During Java configuration (@Configuration), the method is executed and its return value is registered as a bean within a BeanFactory.

17. @ControllerAdvice:

@ControllerAdvice is a specialization of the @Component annotation which allows to handle exceptions across the whole application in one global handling component. It can be viewed as an interceptor of exceptions thrown by methods annotated with @RequestMapping and similar.

18. @ Exception Handler:

The @ExceptionHandler is an annotation used to handle the specific exceptions and sending the custom responses to the client.

Database Table Design:

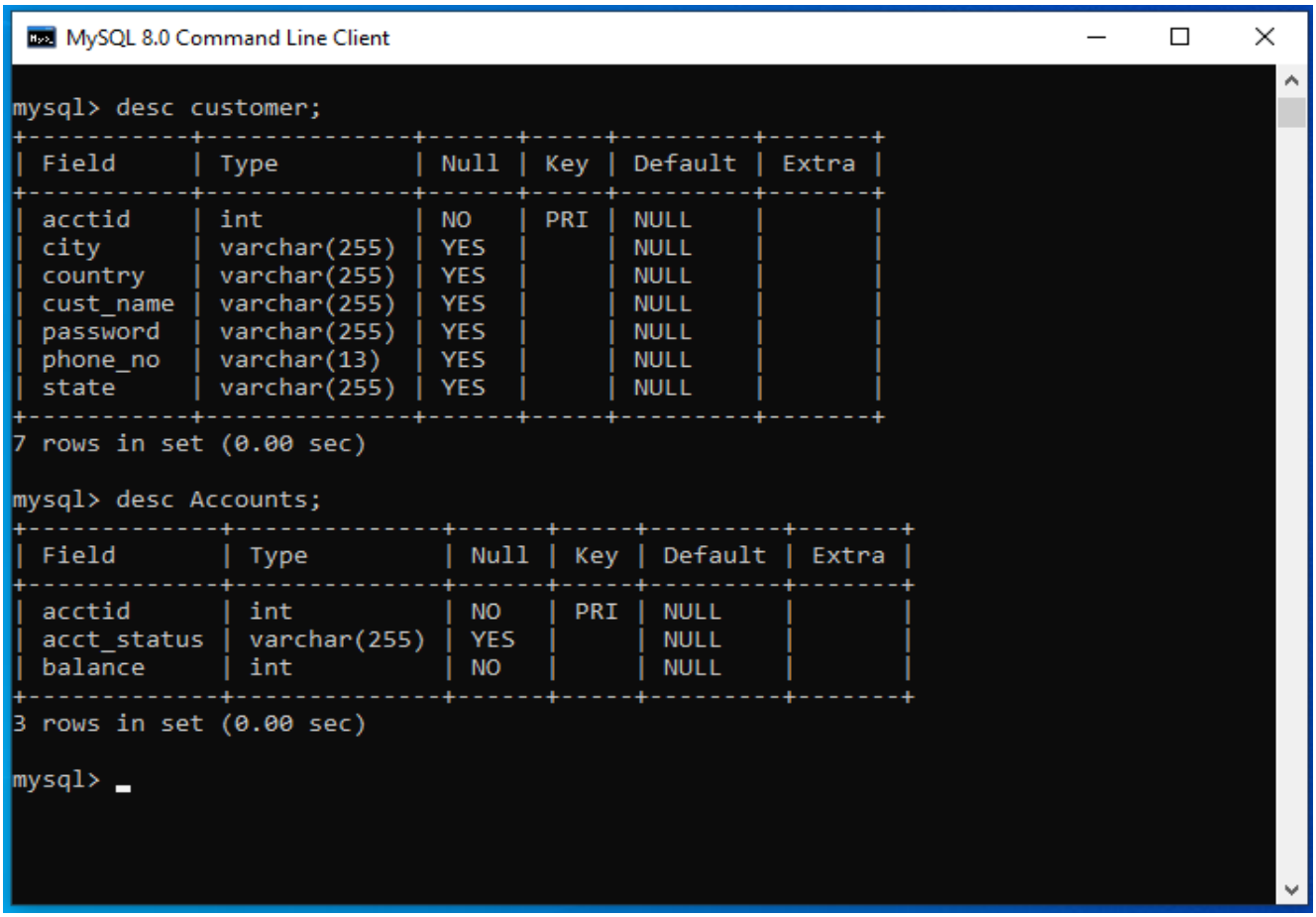
Customer Table:

SL NO	Field Name	Data Type
1	**ID**	**Int**
2	**Name**	**String**
3	**City**	**String**
4	**State**	**String**
5	**Country**	**String**
6	**Phone Number**	**String**
7	**Password**	**String**

Account Table:

SL NO	Field Name	Data Type
1	**ID**	**Int**
2	**Balance**	**Int**
3	**Account Status**	**String**

My SQL Database Table Design:



Conclusion:

This Project is developed to nature the needs of user in a banking Sector by embedding all the tasks of transaction take place in the bank

Bank Management System make online banking Service more accessible t o custome By giving them an Easy palce to find and sort Information like get their Account Inforamtion details,Online ,modify their details online and Transfer their Money to another Account etc..

.