ABSTRACT

The purpose of the project entitled as "HEALTH CARE MANAGEMENT SYSTEM" is to computerize the Front Office Management of Hospital to develop software which is user friendly simple, fast, and cost – effective. It deals with the collection of patient's information, policy details, etc. Traditionally, it was done manually. The main function of the system is register and store patient details and doctor details and retrieves these details as and when required, and also to manipulate these details meaningfully System input contains patient details, policy details, while system output is to get these details on to the screen. Only they can add data into the database. The data can be retrieved easily. The data are well protected for personal use and makes the data processing very fast.

Healthcare management systems, also known as healthcare information management systems, are designed to help healthcare providers collect, store, retrieve and exchange patient healthcare information more efficiently and enable better patient care.

These technology projects which are often compliance driven can require significant investments in time and resources. When selecting and implementing a healthcare management system, providers should keep three things in mind.

CONTENTS

CHAPTER 1: Introduction	1
CHAPTER 2: Requirements	2
2.1.1:Hardware Requirements	
2.1.2 : Software Requirements	
CHAPTER 3: Advantages	3
Disadvantages	
CHAPTER 4: Implementation	4
4.1.1: Spring Boot	
4.1.2: Postman	
4.1.3: MySql	
CHAPTER 5: Coding	10
5.1.1: Controller	
5.1.2: Entity	
5.1.3: Spring Output	
5.1.4: Post man Get method	
5.1.5: Post man Post method	
5.1.6: Post man Put method	
5.1.7: Post man Delete method	
5.1.8: Spring Application Properties	
5.1.9: MySql Tables	

CHAPTER 6: Conclusion 19

CHAPTER 7: Future Work 20

CHAPTER 1

1.1 INTRODUCTION:

Hospitals, clinic and community health agencies can be very different from other work environments. Healthcare systems are complex and there are many things you need to know about types of hospital systems, patient care, insurance, healthcare providers and legal issues

You need to know about the healthcare system so you can be effective on the job. The image below shows the different groups you will be working with. As you work with the healthcare team, you will need to know about hospital systems, types of care, and the roles of each member of the healthcare team. As you work with patients, you will need to understand different types of insurance, how to help uninsured patients and how to protect patient rights and privacy. You also need to know what community resources are available and how to access those services for patients.

Yet when looking at actual country examples of health insurance funds, a more complex picture emerges. There are various players at each stage of the process individuals and institutions, governmental and private. Insurance funds are managed by different types of third party institutions. In some systems the managing organization may also own or manage the service provider. There are variations in the range of care provided under insurance – it may be limited to treatment for serious illness only, or include

CHAPTER 2

2.1 REQUIREMENTS:

2.1.1: HARDWARE REQUIREMENTS :

- Hard disk - 4GB RAM
- 256 Internal memory
- Mouse – optical
- Key board – Multimedia

2.1.2 : SOFTWARE REQUIREMENTS :

- Spring Boot
- Post Man
- My SQL

CHAPTER 3

3.1 ADVANTAGES :

- The increasing prevalence of lifestyle diseases such as diabetes, hypertension, stroke and heart attack among the young as well as the elderly people in India is becoming a major cause of concern.
- Having no family history of these conditions does not ensure protection from related complications.
- In today's fast-paced life, you must be prepared for a medical contingency at all times.
- This is where a robust health insurance policy can help.
- One of the major health insurance benefits, in the face of a medical emergency, is that it allows you to take your mind off the stress related to healthcare costs and focus on the treatment instead.

3.2 DISADVANTAGES :

- Free, continuous and reliable information wasn’t always available.
- Lack of adequate information was a hindrance to arrive at a proper conclusion.
- The preference of consumers regarding various company policies may vary by their own analysis and observations from time to time.

CHAPTER 4

4.1 Implementation

4.1.1 SPRING BOOT :

- Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.
- Micro Service is an architecture that allows the developers to develop and deploy services independently. Each service running has its own process and this achieves the lightweight model to support business applications.

Advantages :

- Simple scalability
- Compatible with Containers
- Minimum configuration
- Lesser production time
- Easy deployment

- Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. You can get started with minimum configurations without the need for an entire Spring configuration setup.

we can choose Spring Boot because of the features and benefits given here

- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides a powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto configured; no manual configurations are needed.
- It offers annotation-based spring application

Spring boots works as :

- Spring Boot automatically configures your application based on the dependencies you have added to the project by using @EnableAutoConfiguration annotation. For example, if MySQL database is on your classpath, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.
- The entry point of the spring boot application is the class contains @SpringBootApplication annotation and the main method.
- Spring Boot automatically scans all the components included in the project by using @ComponentScan annotation.
- Handling dependency management is a difficult task for big projects. Spring Boot resolves this problem by providing a set of dependencies for developers convenience. For example, if you want to use Spring and JPA for database access, it is sufficient if you include spring-boot-starter-data-jpa dependency in your project.
- Spring Boot Starter Actuator dependency is used to monitor and manage your application. Its code is shown below

```
<dependencies>

<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Spring Boot Starter Security dependency is used for Spring Security. Its code is shown below

```
<dependency>
<groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

4.1.2 POST MAN :

- Post man is an API platform for building and using API's .
- Post man simplifies each step of the API lifecycle and streamlines collaboration so you can create better.
- Post man is the place to find an official information on how to use post man in API projects.

Workflow in post man :

- **Making requests :-**

If your building an client app or to connect with APIs the essentials will be

- Connecting to APIs
- Authorization requests
- Grouping requests in collections
- Using variables
- Visualizing data

- **Building and managing APIs** :
  If your developing a back end , need to monitor API performance essentials will be
  - Defining an API from a schema
  - Monitoring test runs
  - Analyzing API performance
  - Mocking responses

- **Publishing APIs :**

  If your exposing an APIs for public use , post man can support developer onboarding :-

  - Publishing documentation
  - Using run in post man

- **Collaborating with your team** :
  If your using postman in your team guides to be followed as

  - Collaborating in post man
  - Creating workspaces
  - Managing post man for an organizations
  - Onboarding your team

- **Developing with post man :**

  If your developing post man APIs resources should be

  - Integrating with a development pipeline
  - Developing with post man utilities
  - Post man API

4.1.3 MY SQL :

- MySQL is a database system used for developing web-based software applications.
- MySQL used for both small and large applications.
- MySQL is a relational database management system (RDBMS).
- MySQL is fast, reliable, and flexible and easy to use.
- MySQL supports standard SQL (Structured Query Language)

- A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

- Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

- Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

Key points of my sql :-

- Database − A database is a collection of tables, with related data.
- Table − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- Primary Key − A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
- Foreign Key − A foreign key is the linking pin between two tables.

ADVANTAGES :

- MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons
- MySQL is released under an open-source license. So you have nothing to pay to use it.

- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.

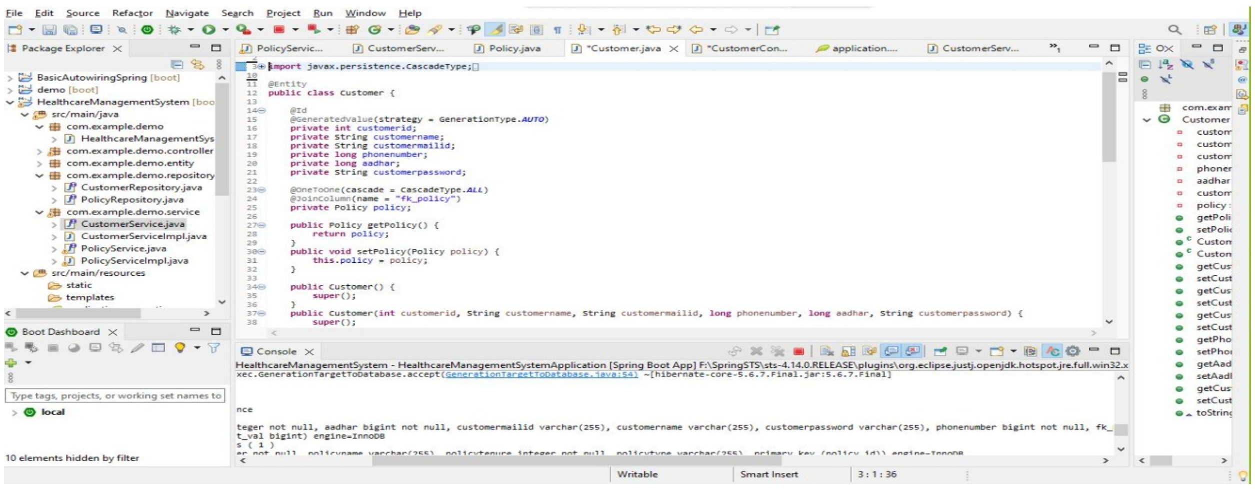
DISADVANTAGES :

- MySQL is not very efficient in handling very large databases.
- MySQL doesn’t have as good a developing and debugging tool as compared to paid databases.
- MySQL versions less than 5.0 do not support COMMIT, stored procedure and ROLE.
- MySQL is prone to data corruption as it inefficient in handling transactions.
- MySQL does not support SQL check constraints.

CHAPTER 5

5.1 CODING :

5.1.1 @ ENTITY



```
import javax.persistence.CascadeType;

@Entity
public class Customer {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int customerid;
    private String customername;
    private String customermailid;
    private long phonenumber;
    private long aadhar;
    private String customerpassword;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "fk_policy")
    private Policy policy;

    public Policy getPolicy() {
        return policy;
    }
    public void setPolicy(Policy policy) {
        this.policy = policy;
    }

    public Customer() {
        super();
    }
    public Customer(int customerid, String customername, String customermailid, long phonenumber, long aadhar, String customerpassword) {
        super();
```

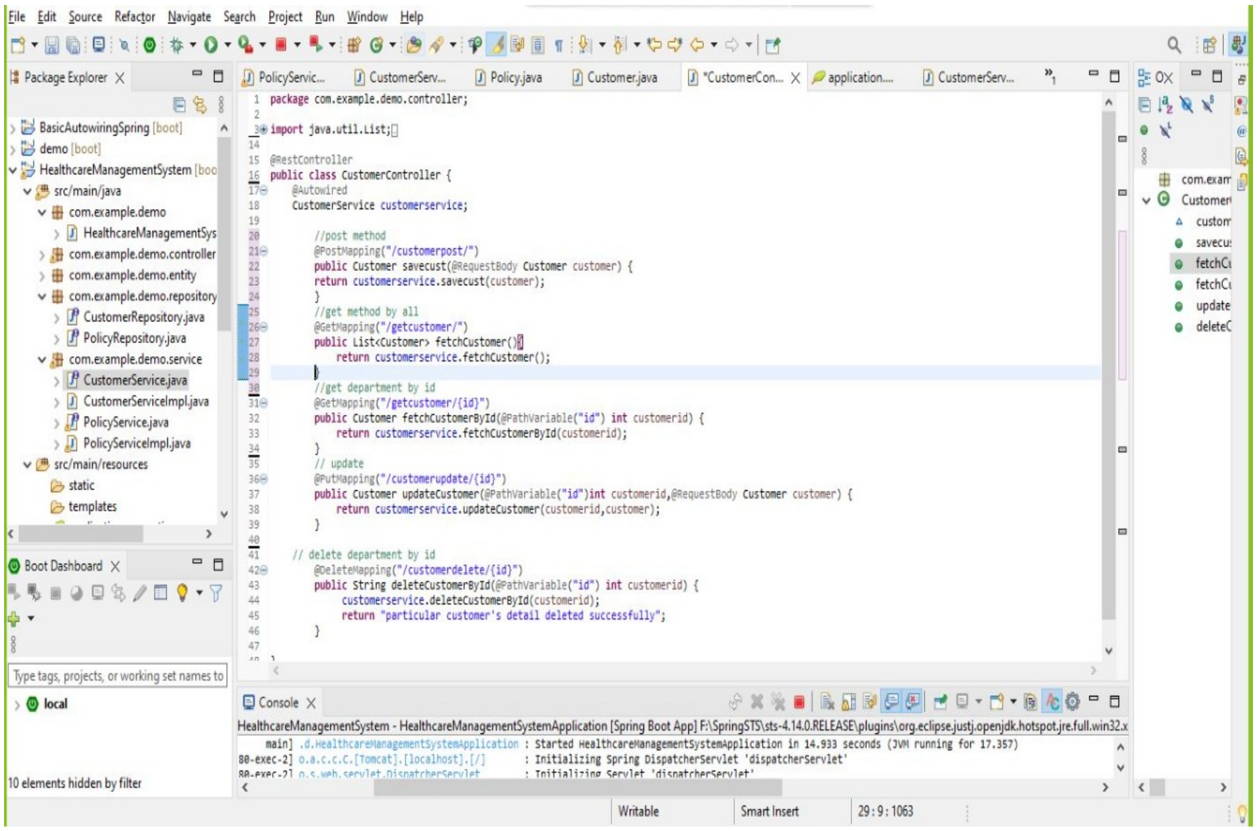
```
HealthcareManagementSystem - HealthcareManagementSystemApplication [Spring Boot App] F:\SpringSTS\sts-4.14.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x
xec.GenerationTargetToDatabase.accept(GenerationTargetToDatabase.java:54) ~[hibernate-core-5.6.7.Final.jar:5.6.7.Final]


nce

teger not null, aadhar bigint not null, customermailid varchar(255), customername varchar(255), customerpassword varchar(255), phonenumber bigint not null, fk_
t_val bigint) engine=InnoDB
s ( 1 )
er not null, policyname varchar(255), policytenure integer not null, policytype varchar(255), primary key (policy_id)) engine=InnoDB
```

- @Entity annotation defines that a class can be mapped to a table. And that is it, it is just a marker, like for example Serializable interface
- @Entity Annotation This annotation specifies that the class is an entity:
- @Entity
  public class Customer {
  }
- The entity name defaults to the name of the class. We can change its name using the name element.

  @Entity(name="customer")
  Public class Customer {
  // fields, getters and setters
  }
- @Entity annotation defines that a class can be mapped to a table. And that is it, it is just a marker, like for example Serializable interface
- Each JPA entity must have a primary key that uniquely identifies it. The @Id annotation defines the primary key.
  Java.persistence.Id

- We can generate the identifiers in different ways which are specified by the @Generated value annotation.
- We can choose from four id generation strategies with the strategy element. The value can be **AUTO**, **TABLE**, **SEQUENCE**, or **IDENTITY**.
- The @OneToOne annotation is used to create the one-to-one relationship between the Student and Address entities

5.1.2 @ CONTROLLER



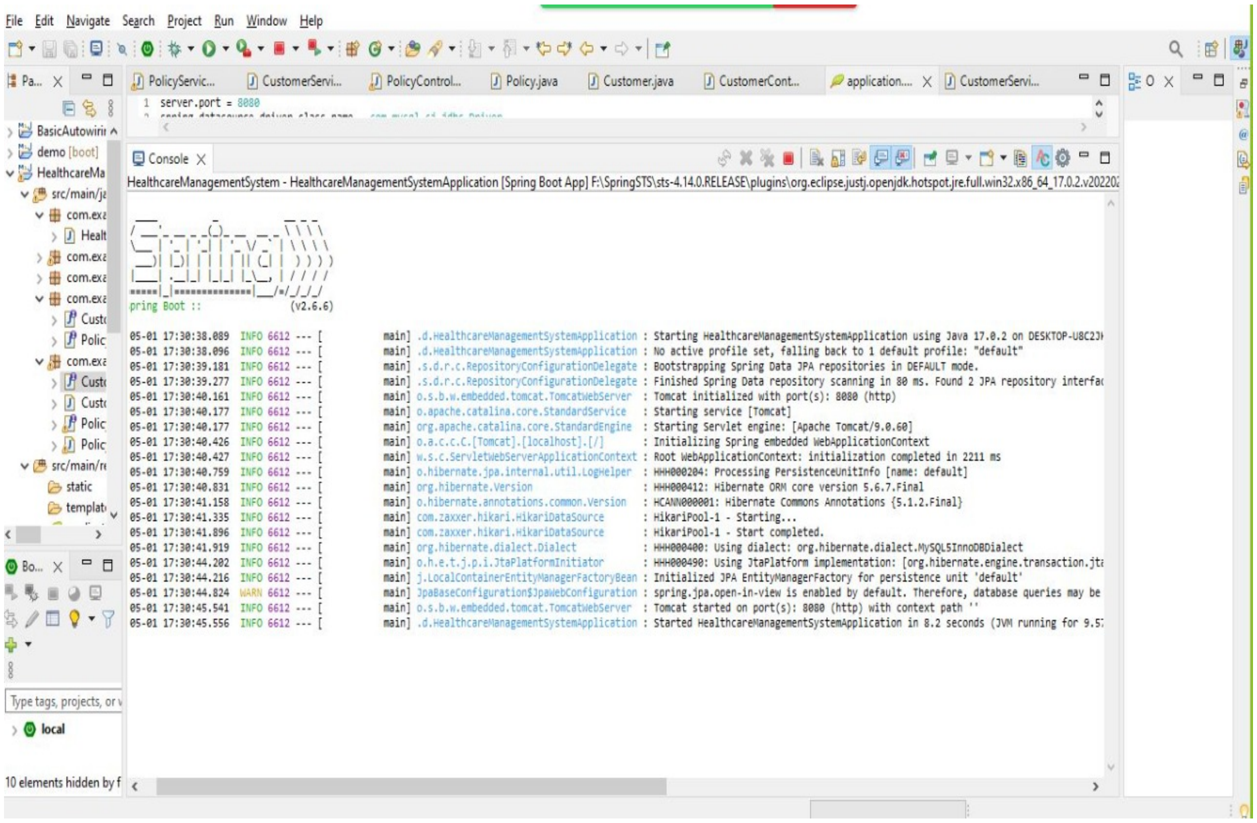
```java
package com.example.demo.controller;

import java.util.List;

@RestController
public class CustomerController {
    @Autowired
    CustomerService customerservice;

        //post method
        @PostMapping("/customerpost/")
        public Customer savecust(@RequestBody Customer customer) {
        return customerservice.savecust(customer);
        }
        //get method by all
        @GetMapping("/getcustomer/")
        public List<Customer> fetchCustomer(){
            return customerservice.fetchCustomer();
        }
        //get department by id
        @GetMapping("/getcustomer/{id}")
        public Customer fetchCustomerById(@PathVariable("id") int customerid) {
            return customerservice.fetchCustomerById(customerid);
        }
        // update
        @PutMapping("/customerupdate/{id}")
        public Customer updateCustomer(@PathVariable("id")int customerid,@RequestBody Customer customer) {
            return customerservice.updateCustomer(customerid,customer);
        }

    // delete department by id
        @DeleteMapping("/customerdelete/{id}")
        public String deleteCustomerById(@PathVariable("id") int customerid) {
             customerservice.deleteCustomerById(customerid);
             return "particular customer's detail deleted successfully";
        }

}
```

```
HealthcareManagementSystem - HealthcareManagementSystemApplication [Spring Boot App] F:\SpringSTS\sts-4.14.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x
     main] .d.HealthcareManagementSystemApplication : Started HealthcareManagementSystemApplication in 14.933 seconds (JVM running for 17.357)
80-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
80-exec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
```

- RestController
  @RestController is a convenience annotation for creating Restful controllers. It is a specialization of @Component and is autodetected through classpath scanning. It adds the @Controller and @ResponseBody annotations. It converts the response to JSON or XML. It does not work with the view technology, so the methods cannot return ModelAndView. It is typically used in combination with annotated handler methods based on the @RequestMapping annotation.

- @Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.
- Autowiring can't be used to inject primitive and string values. It works with reference only.

5.1.3 SPRING OUTPUT



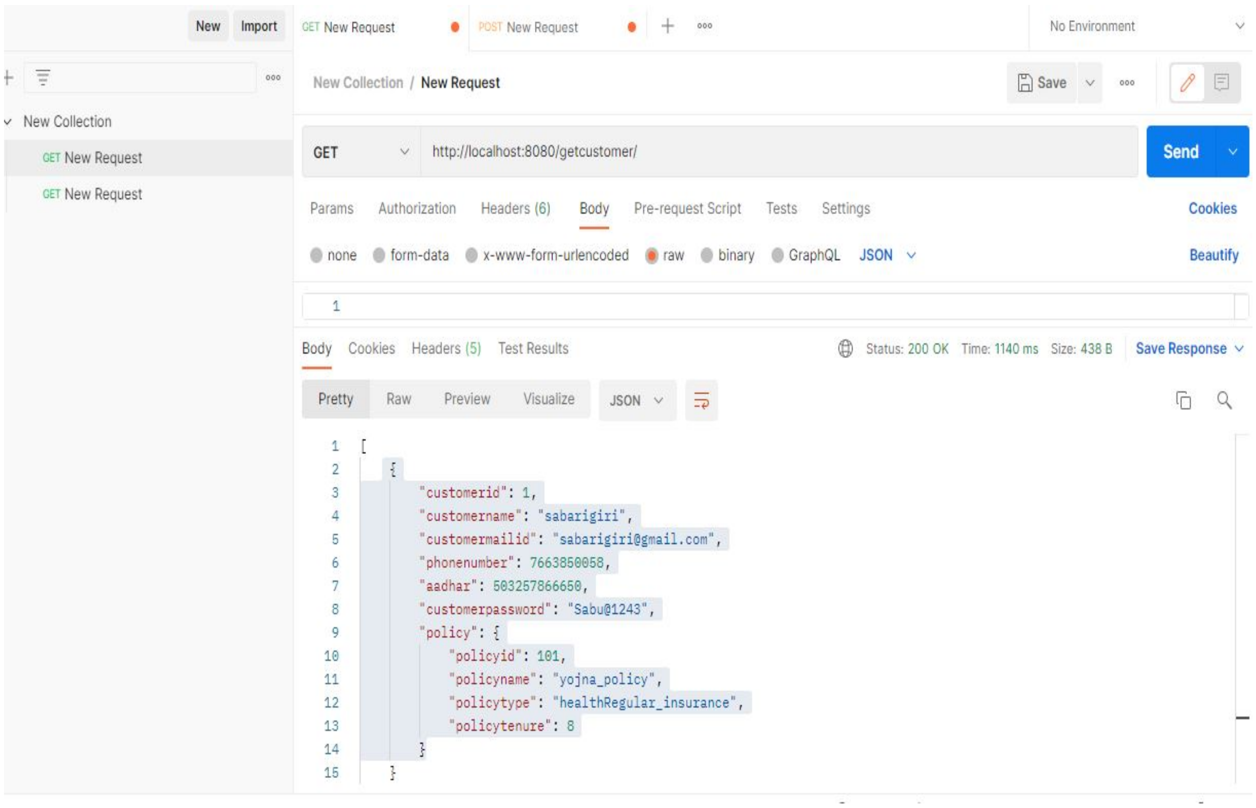
```
server.port = 8080
```

HealthcareManagementSystem - HealthcareManagementSystemApplication [Spring Boot App] F:\SpringSTS\sts-4.14.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220

```
pring Boot ::                (v2.6.6)

05-01 17:30:38.089  INFO 6612 --- [           main] .d.HealthcareManagementSystemApplication : Starting HealthcareManagementSystemApplication using Java 17.0.2 on DESKTOP-U8C2J
05-01 17:30:38.096  INFO 6612 --- [           main] .d.HealthcareManagementSystemApplication : No active profile set, falling back to 1 default profile: "default"
05-01 17:30:39.181  INFO 6612 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
05-01 17:30:39.277  INFO 6612 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 80 ms. Found 2 JPA repository interfa
05-01 17:30:40.161  INFO 6612 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
05-01 17:30:40.177  INFO 6612 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
05-01 17:30:40.177  INFO 6612 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.60]
05-01 17:30:40.426  INFO 6612 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
05-01 17:30:40.427  INFO 6612 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2211 ms
05-01 17:30:40.759  INFO 6612 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
05-01 17:30:40.831  INFO 6612 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.6.7.Final
05-01 17:30:41.158  INFO 6612 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
05-01 17:30:41.335  INFO 6612 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
05-01 17:30:41.896  INFO 6612 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
05-01 17:30:41.919  INFO 6612 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
05-01 17:30:44.202  INFO 6612 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jt
05-01 17:30:44.216  INFO 6612 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
05-01 17:30:44.824  WARN 6612 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be
05-01 17:30:45.541  INFO 6612 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
05-01 17:30:45.556  INFO 6612 --- [           main] .d.HealthcareManagementSystemApplication : Started HealthcareManagementSystemApplication in 8.2 seconds (JVM running for 9.5
```

- The running process of spring boot application in an console.

5.1.4 POST MAN (GET METHOD)



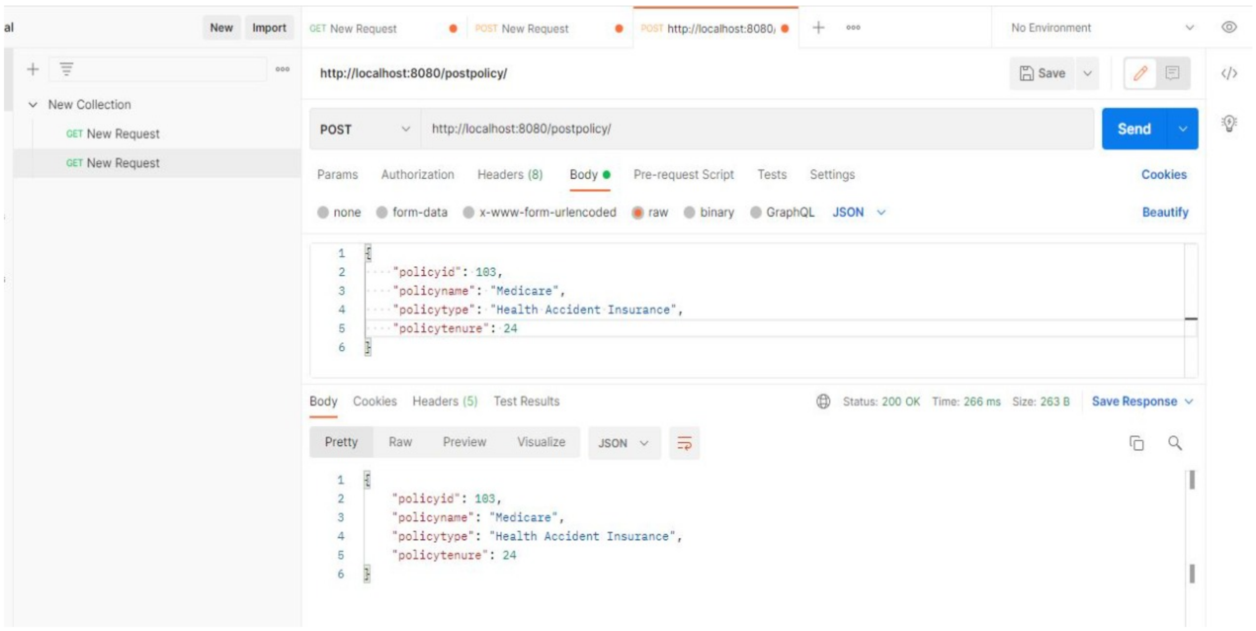
GET http://localhost:8080/getcustomer/

Status: 200 OK Time: 1140 ms Size: 438 B

```
[
    {
        "customerid": 1,
        "customername": "sabarigiri",
        "customermailid": "sabarigiri@gmail.com",
        "phonenumber": 7663850058,
        "aadhar": 503257866650,
        "customerpassword": "Sabu@1243",
        "policy": {
            "policyid": 101,
            "policyname": "yojna_policy",
            "policytype": "healthRegular_insurance",
            "policytenure": 8
        }
    }
```

- The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
- It helps to fetch all the customer details as well as specified customer details also.

5.1.5 POST MAN (POST METHOD)



http://localhost:8080/postpolicy/

POST http://localhost:8080/postpolicy/

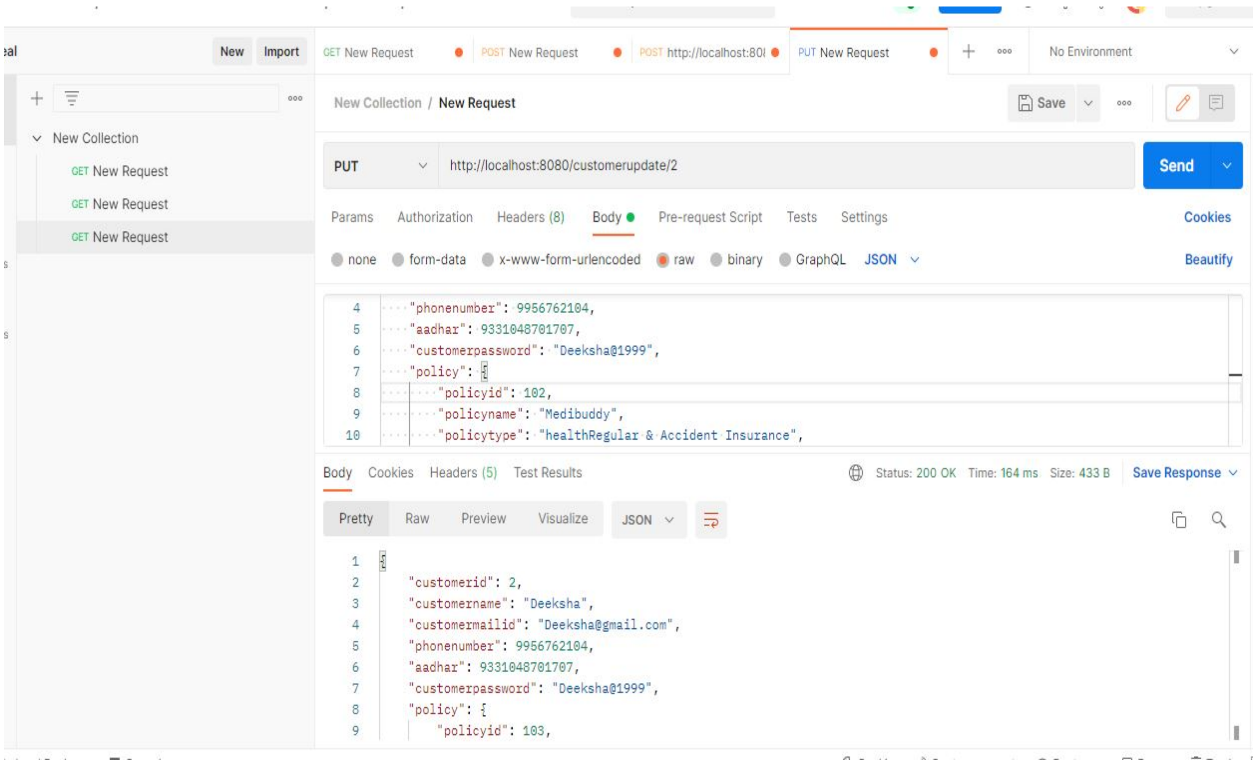
```
{
    "policyid": 103,
    "policyname": "Medicare",
    "policytype": "Health Accident Insurance",
    "policytenure": 24
}
```

Status: 200 OK Time: 266 ms Size: 263 B

```
{
    "policyid": 103,
    "policyname": "Medicare",
    "policytype": "Health Accident Insurance",
    "policytenure": 24
}
```

- Postman POST request allows appending data to the endpoint.
- This is a method used to add information within the request body in the server. It is commonly used for passing delicate information. Once we send some the request body via POST method, the API in turn yields certain information to us in Response.
- Using HTTP and there is a URL link localhost and port number it will help to post the data as well as update fetch and delete method is also help to use.
- Post method: POST helps to send the data of the customer and also policy to create a row for the particular fields.

5.1.6 POST MAN (PUT METHOD)



PUT http://localhost:8080/customerupdate/2

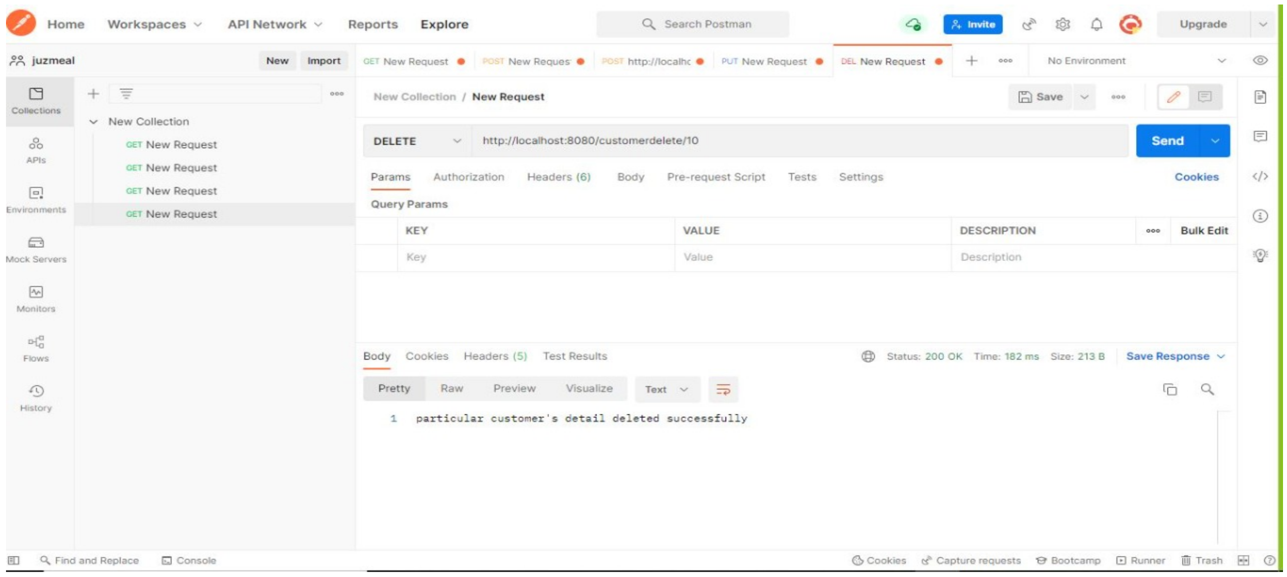
```
    "phonenumber": 9956762104,
    "aadhar": 9331048701707,
    "customerpassword": "Deeksha@1999",
    "policy": {
        "policyid": 102,
        "policyname": "Medibuddy",
        "policytype": "healthRegular & Accident Insurance",
```

Status: 200 OK Time: 164 ms Size: 433 B

```
{
    "customerid": 2,
    "customername": "Deeksha",
    "customermailid": "Deeksha@gmail.com",
    "phonenumber": 9956762104,
    "aadhar": 9331048701707,
    "customerpassword": "Deeksha@1999",
    "policy": {
        "policyid": 103,
```

- Click on the New menu from the Postman application.
- SAVE REQUEST pop-up comes up.
- The Request name gets reflected on the Request tab
- Move to the Body tab below the address bar and select the option raw
- Then choose JSON from the Text dropdown
- It will helps to update the information for the existing customers and policy. Update the data through the postman.
- we have to update the (total record or particular field) of the existing customers.
- Also it will be updated in the database.. Through the existing customers and policy I'd or name we can able to update the data

5.1.7 POST MAN (DELETE METHOD)



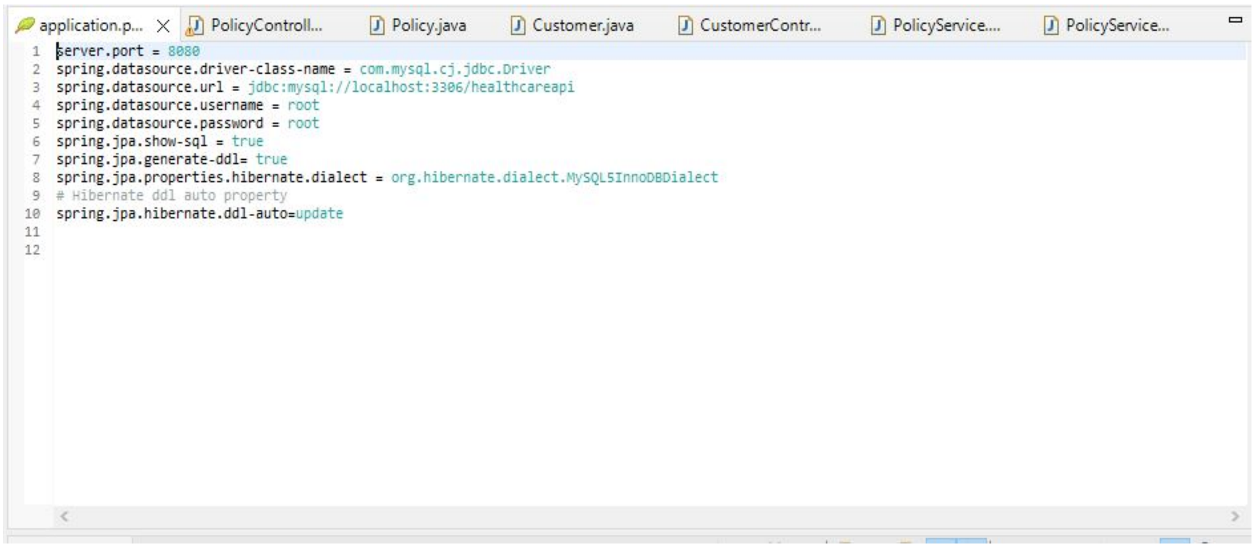
DELETE http://localhost:8080/customerdelete/10

Status: 200 OK Time: 182 ms Size: 213 B

```
particular customer's detail deleted successfully
```

- ➢ Select the “DELETE” in http methods drop down.
- ➢ Pass the request URI in address bar of Postman.
- ➢ Add authorization if applicable.
- ➢ Add headers if applicable.
- ➢ Click on Send button
- ➢ Through the user url request.. it will perform the action of the particular record
- ➢ So it will delete the record based on the request.

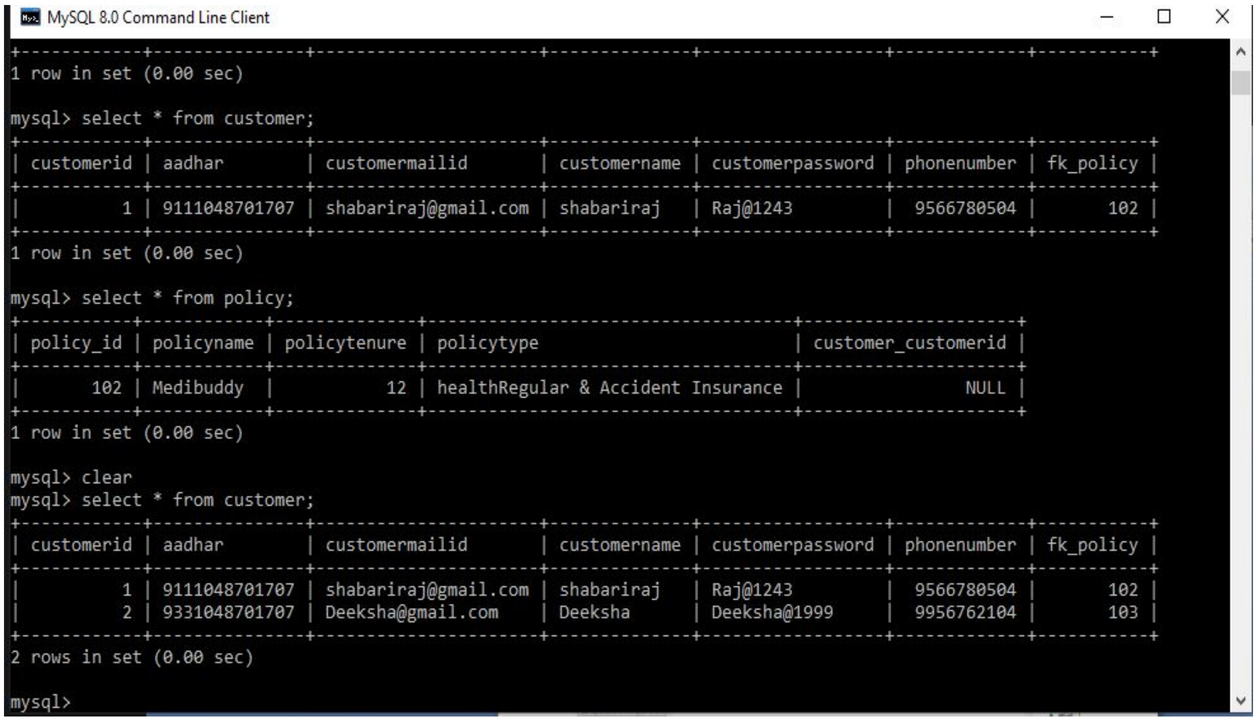
5.1.8 SPRING APPLICATION PROPERTIES



```
server.port = 8080
spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/healthcareapi
spring.datasource.username = root
spring.datasource.password = root
spring.jpa.show-sql = true
spring.jpa.generate-ddl= true
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
# Hibernate ddl auto property
spring.jpa.hibernate.ddl-auto=update
```

- So in a spring boot application, application. properties file is used to write the application-related property into that file. This file contains the different configuration which is required to run the application in a different environment, and each environment will have a different property defined by it.
- There are port number and hibernate query languages. It will help to show the structure of the table data and queries in the console
- Through the port number it will fetch the data from the local server.

5.1.9 MY SQL TABLES


MySQL 8.0 Command Line Client

```
+------------+---------------+---------------------+--------------+------------------+-------------+-----------+
1 row in set (0.00 sec)

mysql> select * from customer;
+------------+---------------+---------------------+--------------+------------------+-------------+-----------+
| customerid | aadhar        | customermailid      | customername | customerpassword | phonenumber | fk_policy |
+------------+---------------+---------------------+--------------+------------------+-------------+-----------+
|          1 | 9111048701707 | shabariraj@gmail.com | shabariraj   | Raj@1243         |  9566780504 |       102 |
+------------+---------------+---------------------+--------------+------------------+-------------+-----------+
1 row in set (0.00 sec)

mysql> select * from policy;
+-----------+------------+-------------+-------------------------------------+---------------------+
| policy_id | policyname | policytenure | policytype                          | customer_customerid |
+-----------+------------+-------------+-------------------------------------+---------------------+
|       102 | Medibuddy  |          12 | healthRegular & Accident Insurance  |                NULL |
+-----------+------------+-------------+-------------------------------------+---------------------+
1 row in set (0.00 sec)

mysql> clear
mysql> select * from customer;
+------------+---------------+---------------------+--------------+------------------+-------------+-----------+
| customerid | aadhar        | customermailid      | customername | customerpassword | phonenumber | fk_policy |
+------------+---------------+---------------------+--------------+------------------+-------------+-----------+
|          1 | 9111048701707 | shabariraj@gmail.com | shabariraj   | Raj@1243         |  9566780504 |       102 |
|          2 | 9331048701707 | Deeksha@gmail.com    | Deeksha      | Deeksha@1999     |  9956762104 |       103 |
+------------+---------------+---------------------+--------------+------------------+-------------+-----------+
2 rows in set (0.00 sec)

mysql>
```

- **As the customer tables and policy tables are inserted a data's into an mysql database.**

CHAPTER 6

6.1 CONCLUSION:

Everyone needs to be well informed and concerned about the quality of care. Everyone means patients and their families, consumer agents and advocates, health professionals, administrators of health plans and facilities, purchasers of health care services, and policymakers at all levels.

So as a part of this health care management process we can give the insurance policy to the customers and help them .

CHAPTER 7

7.1 FUTURE WORK :

- The main goal for the future work will be developing an complete technical support team for helping the customers problems and solving there problems.
- some of new policies also will be added.