AN INTERNSHIP REPORT ON

JAVA & FSD PROGRAMMING

A Report Submitted to

Blackbuck Engineers Pvt. Ltd

Submitted by

PAKANATI DIVYA	BBPBV003
CHUNDI VISHNU PRIYA	BBPBV006
SYED ALIYA SULTHANA	BBPBV005
PATIBANDLA SRAVANI	BBPBV007

Blackbuck Engineers Pvt. Ltd
Road no 36, Jubilee Hills, Hyderabad

CERTIFICATE

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We would like to acknowledge all those without whom this project would not have been
successful. Firstly, we would like to thank our Internet and Java & FSD Programmer Mr.
(Instructor Name) who guided us throughout the project and gave his immense support. he
made us understand how to complete this project and without his presence, this project would
not have been complete. We also got to know a lot about parallelization and its benefits. This
project has been a source to learn and bring our theoretical knowledge to the real-life world.
Once again, thanks to everyone for making this project Succesfull.

Place :

Date :

ABSTRACT

The modern business landscape is characterized by dynamic market demands, increasing competition, and a relentless pursuit of operational efficiency. To navigate these challenges successfully, organizations rely on effective inventory management systems that serve as the backbone of their supply chain operations. This abstract provides an overview of an Inventory Management System (IMS) for a comprehensive project report.

This project report delves into the design, development, and implementation of an Inventory Management System tailored to meet the unique needs of businesses across various industries. The primary objective of this system is to optimize inventory control, minimize carrying costs, reduce stockouts, and maximize overall operational efficiency.

The proposed Inventory Management System encompasses various essential features and functionalities, including real-time inventory tracking, demand forecasting, supplier management, order processing, and reporting and analytics. Leveraging modern technology and data-driven approaches, this system empowers businesses to make informed decisions regarding inventory levels, procurement, and order fulfillment.

1. INTRODUCTION

1.1. SYSTEM OVERVIEW

The Inventory Management System (IMS) is a sophisticated software solution developed to optimize and streamline inventory-related processes within an organization. This system is designed to address the challenges associated with inventory control, procurement, and order fulfillment, enhancing operational efficiency and ensuring cost-effectiveness.

1.2. OBJECTIVE

The objective of this project is to develop an Inventory Management System (IMS) with the primary goal of optimizing inventory control and enhancing operational efficiency. The IMS will provide real-time visibility and control over inventory levels, minimizing overstocking and stockouts, while also streamlining inventory-related processes through automation. By empowering users with data-driven insights, the system aims to improve decision-making, reduce carrying costs, and enhance profitability. The IMS will prioritize user experience, offering an intuitive interface and seamless integration with existing business operations. Furthermore, it will foster strong supplier collaboration, ensure data security and compliance, and be scalable to accommodate growth and changing market dynamics. The project will be assessed through cost-benefit analysis and key performance indicators to ensure alignment with organizational objectives and goals.

1.3. PROBLEM IDENTIFICATION

The Inventory Management System (IMS) project addresses several critical issues and challenges within the organization:

- ❖ Inefficient Inventory Control
- ❖ Manual Processes and Errors
- ❖ Lack of Data-Driven Insights
- ❖ Suboptimal Resource Allocation
- ❖ Poor Supplier Collaboration
- ❖ Scalability Challenges
- ❖ Data Security and Compliance Risks

1.4. APPLICATIONS

The Inventory Management System (IMS) project can find applications across various industries and business types where inventory management plays a critical role in operations. Here are some potential applications for the IMS project:❖ Angel Investors can have a look at this website.

❖ Retail and E-commerce.

❖ Manufacturing

❖ Wholesale and Distribution

❖ Hospitality and Food Service

1.5. LITERATURE SURVEY

A literature survey for an Inventory Management System (IMS) project involves reviewing relevant academic papers, articles, books, and industry reports to gather insights, best practices, and prior research findings related to inventory management systems and their implementation. Here is a concise overview of key points to include in your literature survey

1) Introduction to Inventory Managemen

Define the concept of inventory management and its significance in various industries.

Explain how effective inventory management can lead to cost savings, improved customer service, and increased profitability.

2) Types of Inventory Management Systems:

Describe various inventory management methods, such as Just-In-Time (JIT), Economic Order Quantity (EOQ), and ABC analysis.

Discuss the advantages and disadvantages of each method in different operational contexts.

3) Role of Technology in Inventory Management::

Explore the role of information technology and software solutions in modern inventory management.

Highlight the benefits of using Inventory Management Systems (IMS) for automating inventory-related tasks and data analysis.

4) Benefits of IMS Implementation::

Summarize research findings and industry case studies on the benefits of implementing IMS, including cost reduction, improved accuracy, and better decision-making.

Showcase examples of organizations that have successfully leveraged IMS to optimize their inventory control.

5) Challenges in Inventory Management:

Discuss common challenges in inventory management, such as demand variability, supplier uncertainties, and carrying costs.

Present research on how IMS can address these challenges through real-time tracking and data analytics.

6) Data-Driven Inventory Management:

Explore the importance of data-driven decision-making in inventory management.
Provide examples of how IMS can harness data analytics to forecast demand, optimize reorder points, and reduce excess inventory.
7) User Experience and Adoption:::
Review studies on user adoption of inventory management software and the impact of user experience on system effectiveness.
Highlight best practices for designing user-friendly interfaces and training programs.
8) Supplier Relationship Management::
Investigate the role of IMS in improving supplier collaboration and communication.

Discuss how automated purchase order generation and tracking can enhance supplier relationships.

1.6.LIMITATIONS

The Inventory Management System (IMS) project, like any other project, may have certain limitations that need to be considered during its planning and implementation. Identifying these limitations in advance can help project stakeholders make informed decisions and develop strategies to mitigate potential challenges. Here are some common limitations associated with an IMS project:❖ Just Signup doesn’t create an account because manual verification will be done

- ❖ Initial Implementation Cost
- ❖ Integration challenges
- ❖ Data Accuracy and Integrity
- ❖ UserAdoption.

2. SYSTEM ANALYSIS

2.1.EXISTING SYSTEM

In the existing inventory management system (if one exists), several observations and assessments need to be made:

Current Inventory Practices: Evaluate how inventory is currently managed. This may involve manual processes, the use of spreadsheets, or legacy software.

Data Collection Methods: Examine how data related to inventory, such as stock levels, supplier information, and order history, is collected and stored. Are there any data silos or duplication of efforts

Inventory Tracking: Assess how inventory is tracked, including methods for recording stock movements, replenishing inventory, and identifying low-stock items.

Order Processing: Analyze the order processing workflow, including order creation, order fulfillment, and any challenges or bottlenecks in the process.

Reporting and Analytics: Determine if there are any existing reporting tools or data analysis methods in place for monitoring inventory performance and trends.

User Feedback: Gather feedback from users and stakeholders about the limitations and issues they encounter with the current system.

Integration: Identify any existing systems or software that the current inventory management system needs to integrate with, such as accounting software or e-commerce platforms.

Challenges and Limitations: Document the shortcomings and challenges of the existing system, such as inefficiencies, inaccuracies, and operational difficulties.

2.2 Proposed System:

In the context of the Inventory Management System (IMS) project, let's break down the proposed system into its different components:

2.2.1 On Client Side

User-Friendly Interface: The proposed IMS will feature an intuitive and user-friendly interface for clients, allowing them to easily monitor their inventory levels, access demand forecasting insights, and manage orders.

Real-Time Inventory Tracking: Clients will have access to real-time inventory tracking capabilities, enabling them to view stock levels, track product locations, and receive automated alerts for low-stock items.

Demand Forecasting: The system will provide demand forecasting tools, allowing clients to predict future inventory needs based on historical data and market trends, thus optimizing stock levels and reducing carrying costs.

Order Management: Clients can efficiently create, track, and manage customer orders through the IMS, ensuring timely and accurate order fulfillment.

2.2.2 On the Sponsored Side (Sponsorship and Investment:

Theme Exploration: Sponsors will have access to startup themes and profiles through the web portal, enabling them to explore various investment opportunities aligned with their interests.

Communication and Investment: Sponsors can use the portal to communicate directly with startups of interest, inquire about investment opportunities, and negotiate investment terms as per the startup's requirements.

Profit Sharing Mechanism: The system will facilitate a transparent profit-sharing mechanism, allowing sponsors to expect a share in the profits generated by the startups they invest in, fostering a mutually beneficial partnership.

2.2.3 On Company Side (System Management and Administration):

Supplier Management: The system will streamline supplier management by automating purchase order generation, tracking deliveries, and monitoring supplier performance.

Reporting and Analytics: The company side will benefit from comprehensive reporting and analytics tools, offering insights into inventory trends, procurement efficiency, and overall system performance.

Security and Compliance: Robust security measures will be in place to protect sensitive inventory and business data, ensuring compliance with industry regulations and standards.

2.2.4 Benefits of Proposed System:

Enhanced Efficiency: The proposed IMS will streamline inventory management processes, reducing manual effort and minimizing errors, leading to enhanced operational efficiency.

Cost Reduction: Optimized inventory control and procurement processes will result in reduced carrying costs, cost-effective resource allocation, and improved profitability.

Improved Customer Satisfaction: Timely and accurate order processing, facilitated by the IMS, will enhance customer satisfaction and loyalty.

Data-Driven Decision-Making: Access to real-time data and analytics tools empowers organizations to make informed decisions, improving overall decision quality.

Strong Supplier Relationships: The system's features for efficient supplier communication and collaboration will foster strong relationships with suppliers, ensuring timely deliveries and reliable inventory replenishment.

Scalability: The IMS will be designed to accommodate growth and adapt to changing market conditions, supporting the company's long-term objectives.

Data Security: Robust data security measures will protect sensitive inventory and business data, reducing the risk of data breaches and ensuring data integrity.

Compliance: The system will adhere to industry-specific regulations and standards, facilitating compliance monitoring and reducing regulatory risks.

Profitability: The transparent profit-sharing mechanism on the sponsored side will attract potential investors and startups, creating opportunities for profitable collaborations.

Strategic Insights: Reporting and analytics tools will provide valuable insights into inventory and supplier performance, enabling data-driven strategic decisions.

3.REQUIREMENT SPECIFICATION

3.1.HARDWARE REQUIREMENTS

❖ Laptop 4 GHz minimum, multi-core processor

❖ Memory (RAM) 4GB, preferably higher, and commensurate with concurrent usage

❖ Hard disk space 1TB

3.2.SOFTWARE REQUIREMENTS

❖ Windows 2016

❖ Eclipse

❖ Notepad

❖ Tomcat web server

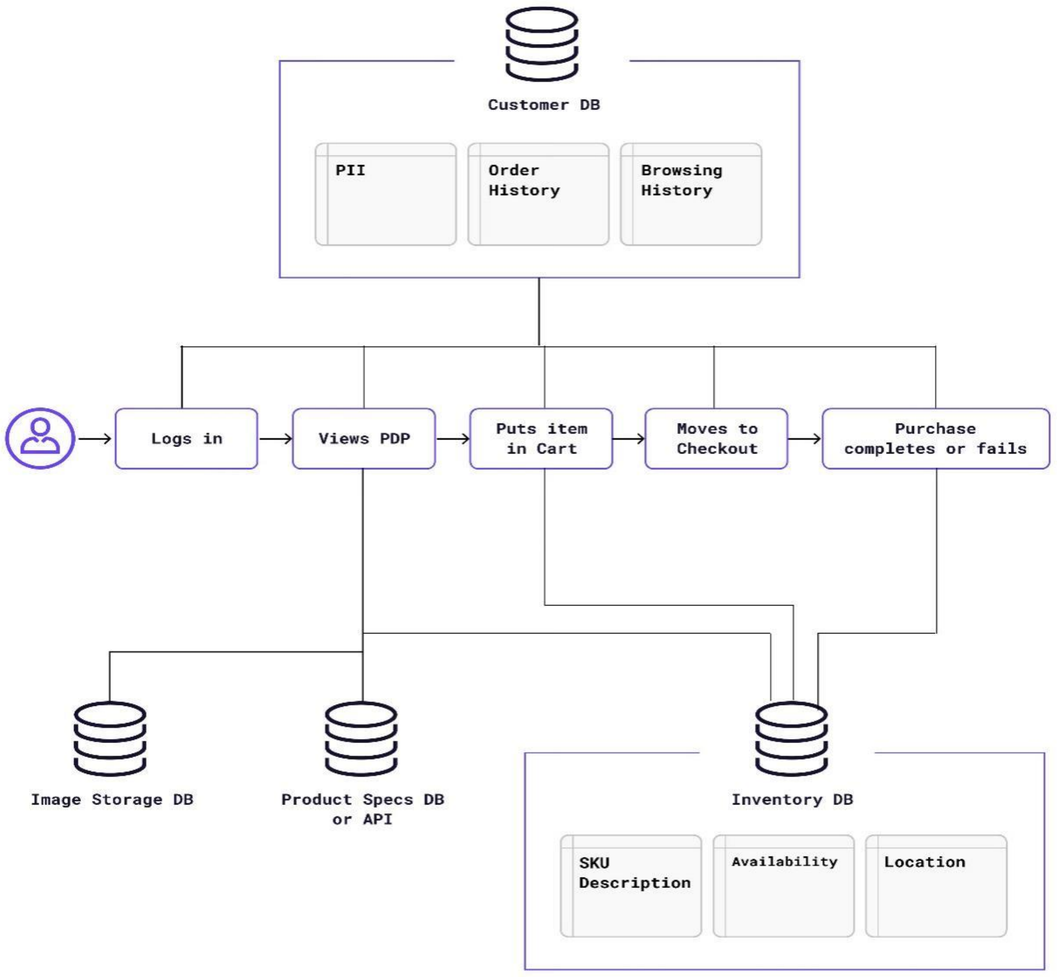
❖ MySQL

3.3.COMPONENTS USED

❖ Google Chrome

4. ARCHITECTURE DESIGN SPECIFICATION

4.1.SYSTEM ARCHITECTURE



The Inventory Management System (IMS) project involves three primary sides or user roles:

Client Side:

Users: Clients or customers of the organization who purchase products or services.

Responsibilities:

Access the IMS to place orders.

Monitor order status and history.

Check product availability.

Receive notifications about order fulfillment.

Potentially access demand forecasting information to plan their own inventory needs.

Sponsor Side:

Users: Sponsors or investors interested in supporting startup ventures through the organization's web portal.

Responsibilities:

Explore startup themes and profiles.

Communicate with startups through the portal.

Invest in startups based on their requirements and potential returns.

Expect a share in the profits generated by the startups they invest in.

Company Side:

Users: Employees and administrators within the organization responsible for managing inventory, orders, suppliers, and the overall IMS.

Responsibilities:

Manage and optimize inventory levels.

Process and fulfill customer orders.

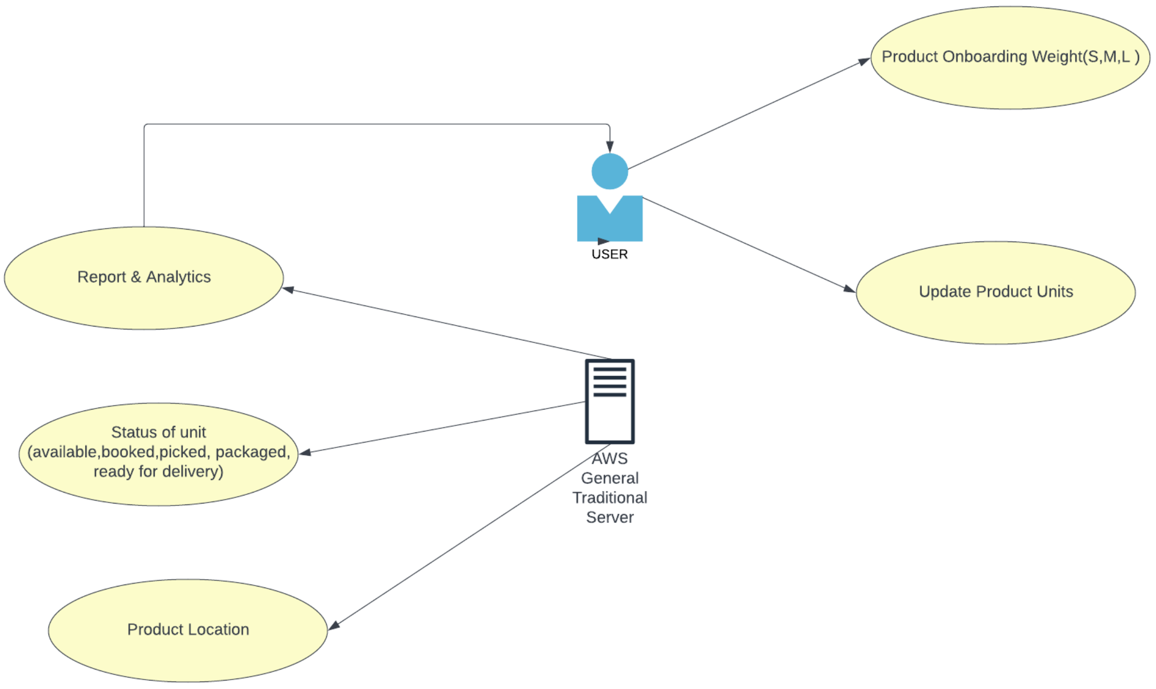
Monitor and forecast demand.

Manage relationships with suppliers.

Generate reports and analytics for decision-making.

Ensure the security, scalability, and compliance of the IMS.

4.2.DETAILED DESIGN


Product Onboarding Weight(S,M,L)
USER
Report & Analytics
Update Product Units
Status of unit (available,booked,picked, packaged, ready for delivery)
AWS General Traditional Server
Product Location

4.3.COMPONENTS USED

- HTML/CSS
- Java Script
- AJAX
- Mysql
- Apache Web server

4.4 DATABASE DESIGN:

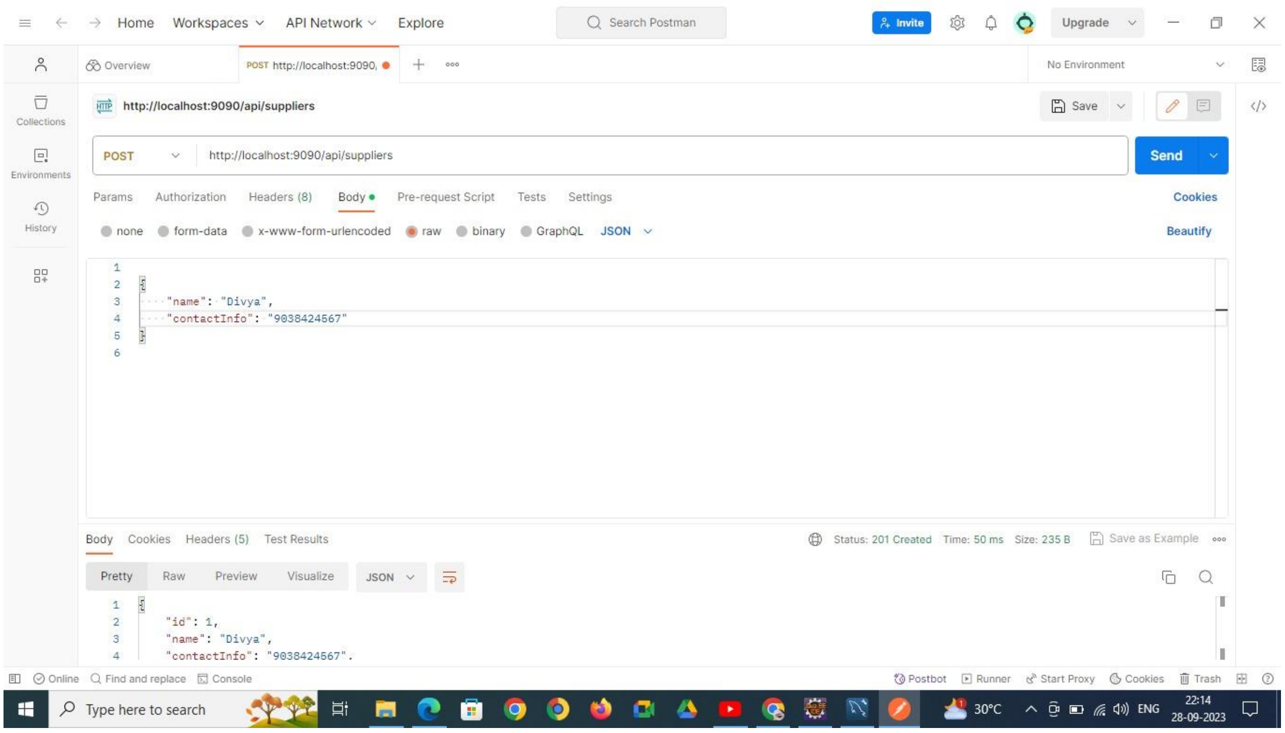
Category

Product

Supplier

SUPPLIER TABLE

POST



http://localhost:9090/api/suppliers

POST http://localhost:9090/api/suppliers

Body: raw JSON

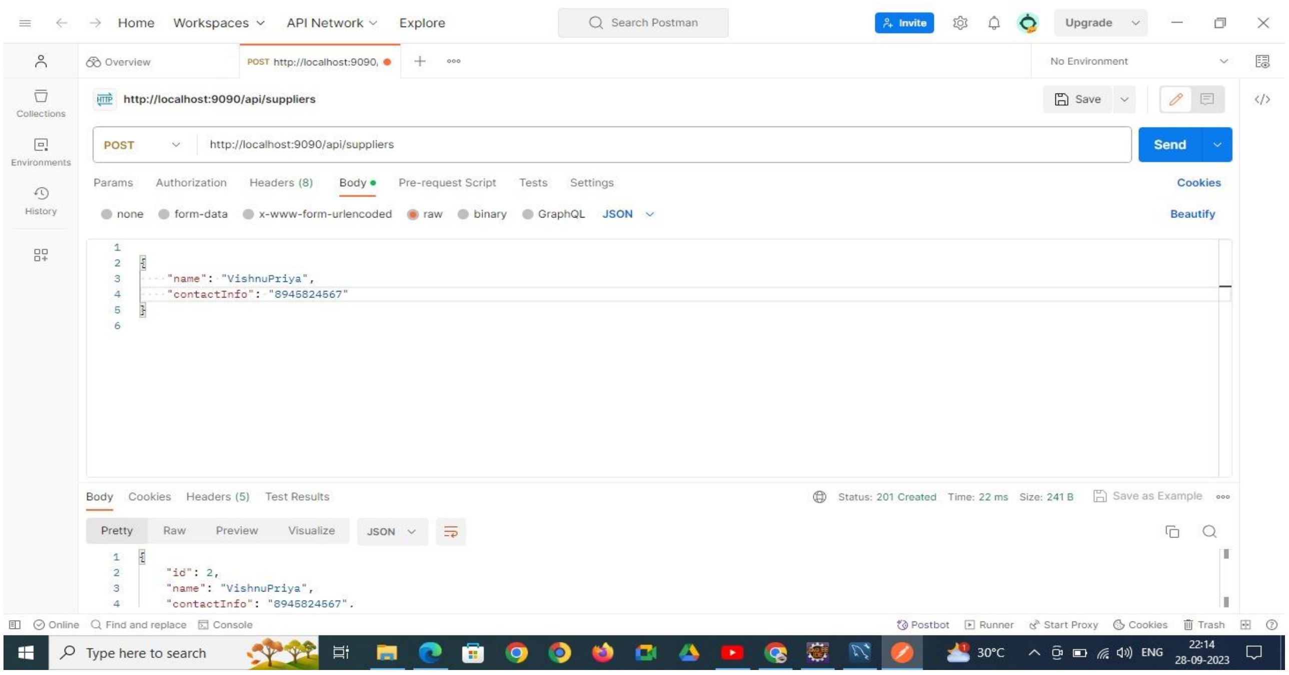
```
{
    "name": "Divya",
    "contactInfo": "9038424567"
}
```

Status: 201 Created Time: 50 ms Size: 235 B

```
{
    "id": 1,
    "name": "Divya",
    "contactInfo": "9038424567",
```

FIG:Inserting record into supplier table

POST:



http://localhost:9090/api/suppliers

POST http://localhost:9090/api/suppliers

Body: raw JSON

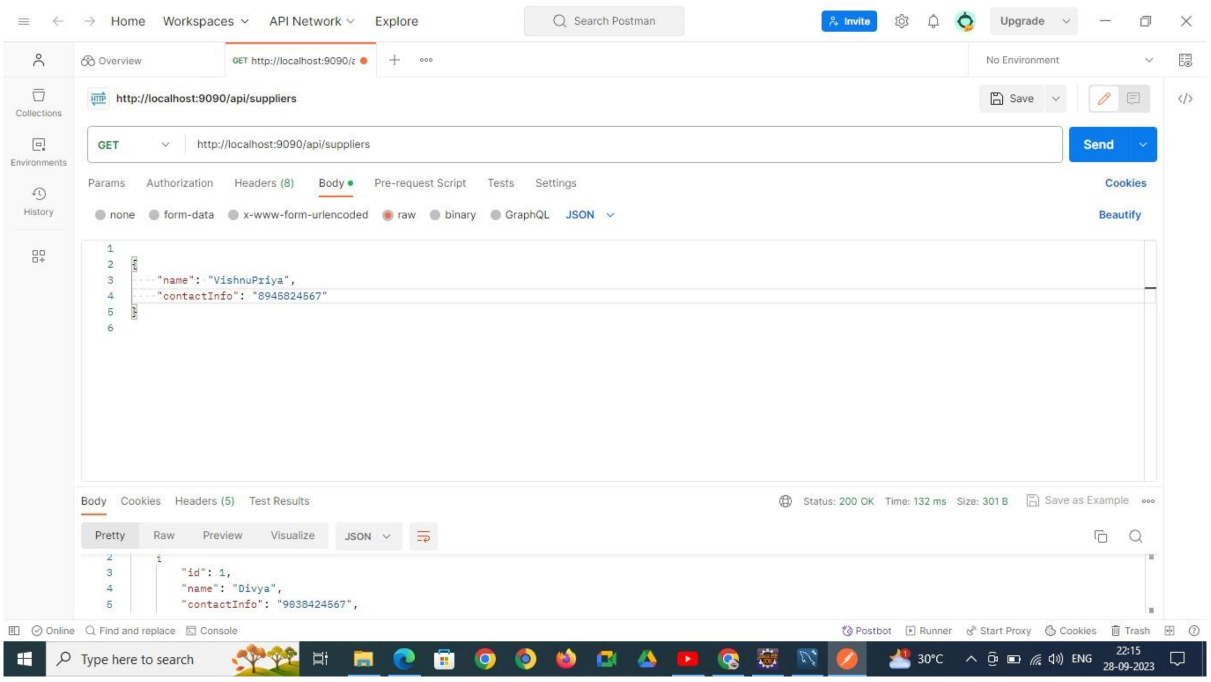
```
{
    "name": "VishnuPriya",
    "contactInfo": "8945824567"
}
```

Status: 201 Created Time: 22 ms Size: 241 B

```
{
    "id": 2,
    "name": "VishnuPriya",
    "contactInfo": "8945824567",
```

Fig: Inserting another record into supplier table

GET



http://localhost:9090/api/suppliers

GET http://localhost:9090/api/suppliers

Body: raw JSON

```
{
    "name": "VishnuPriya",
    "contactInfo": "8945824567"
}
```

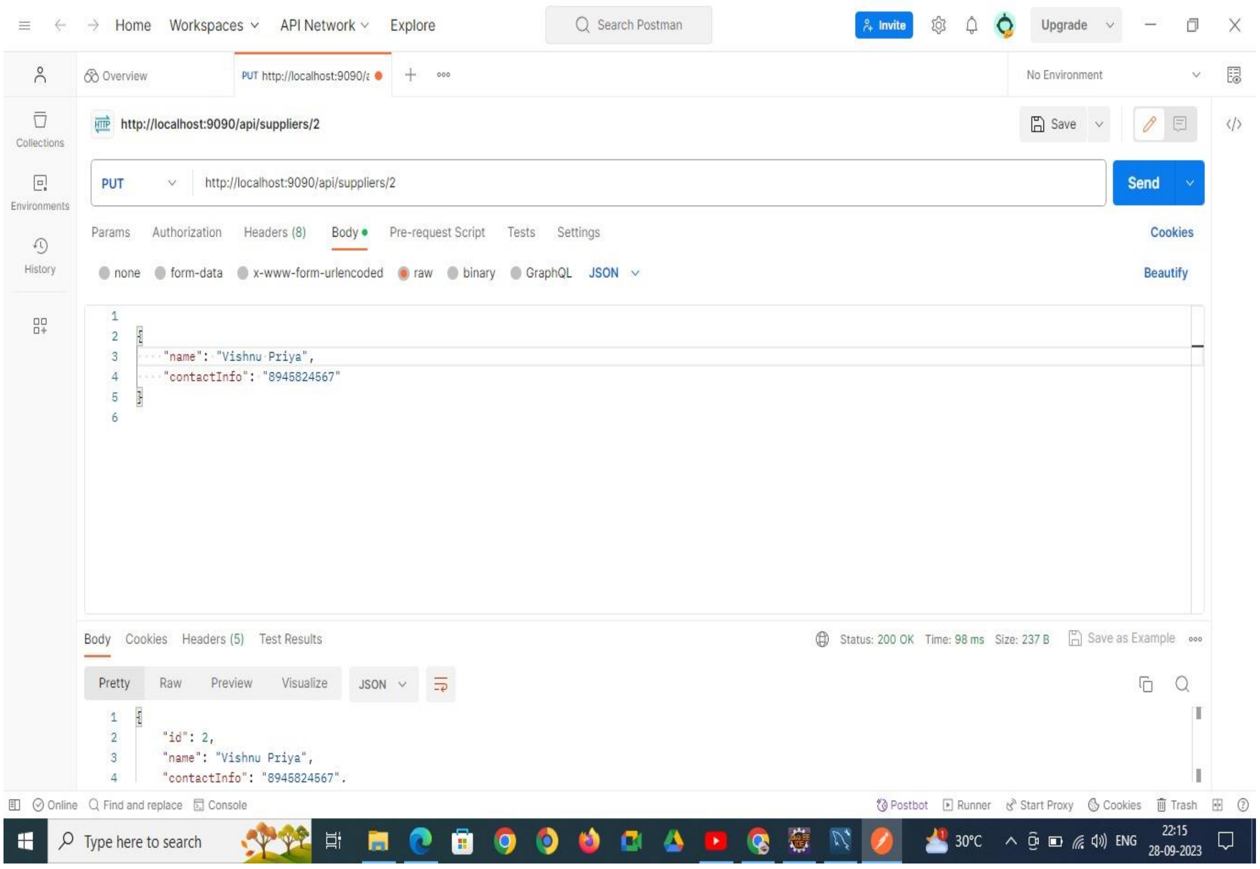
Status: 200 OK Time: 132 ms Size: 301 B

```
    {
        "id": 1,
        "name": "Divya",
        "contactInfo": "9038424567",
```

30°C ENG 22:15 28-09-2023

FIG:Fetching data from supplier table

PUT:



http://localhost:9090/api/suppliers/2

PUT http://localhost:9090/api/suppliers/2

Body: raw JSON

```
{
    "name": "Vishnu Priya",
    "contactInfo": "8945824567"
}
```

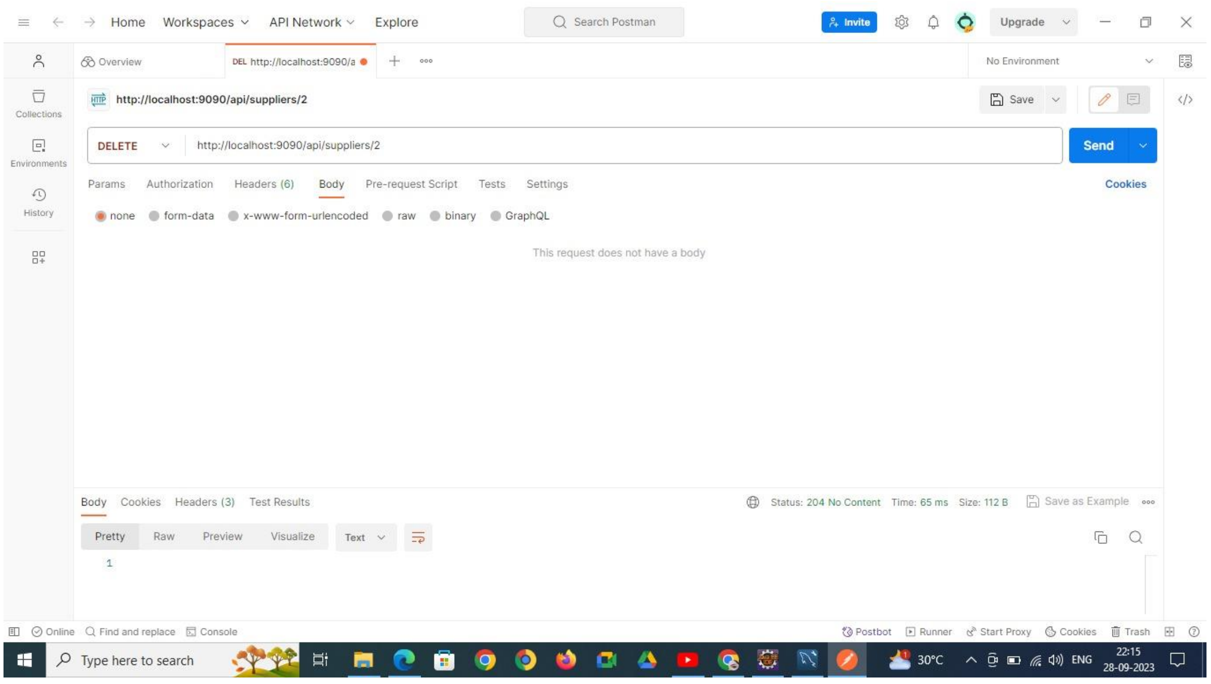
Status: 200 OK Time: 98 ms Size: 237 B

```
{
    "id": 2,
    "name": "Vishnu Priya",
    "contactInfo": "8945824567",
```

30°C ENG 22:15 28-09-2023

FIG:Updating The record of supplier table

DELETE



http://localhost:9090/api/suppliers/2

DELETE http://localhost:9090/api/suppliers/2

Body: [● none] form-data x-www-form-urlencoded raw binary GraphQL

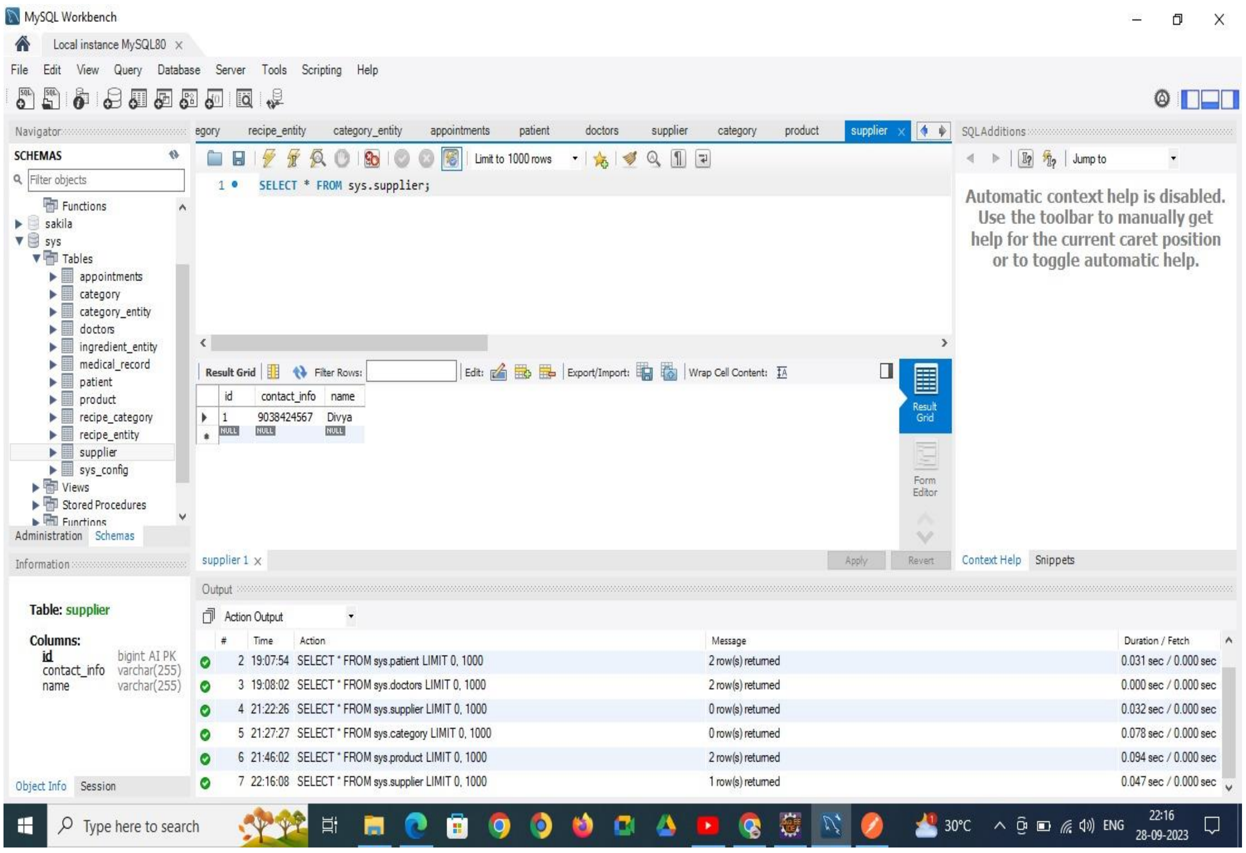
This request does not have a body

Body Cookies Headers (3) Test Results — Status: 204 No Content Time: 65 ms Size: 112 B

```
1
```

FIG: Deleting a Record from Supplier table

Final Table:



```
SELECT * FROM sys.supplier;
```

id	contact_info	name
1	9038424567	Divya
NULL	NULL	NULL

Table: supplier

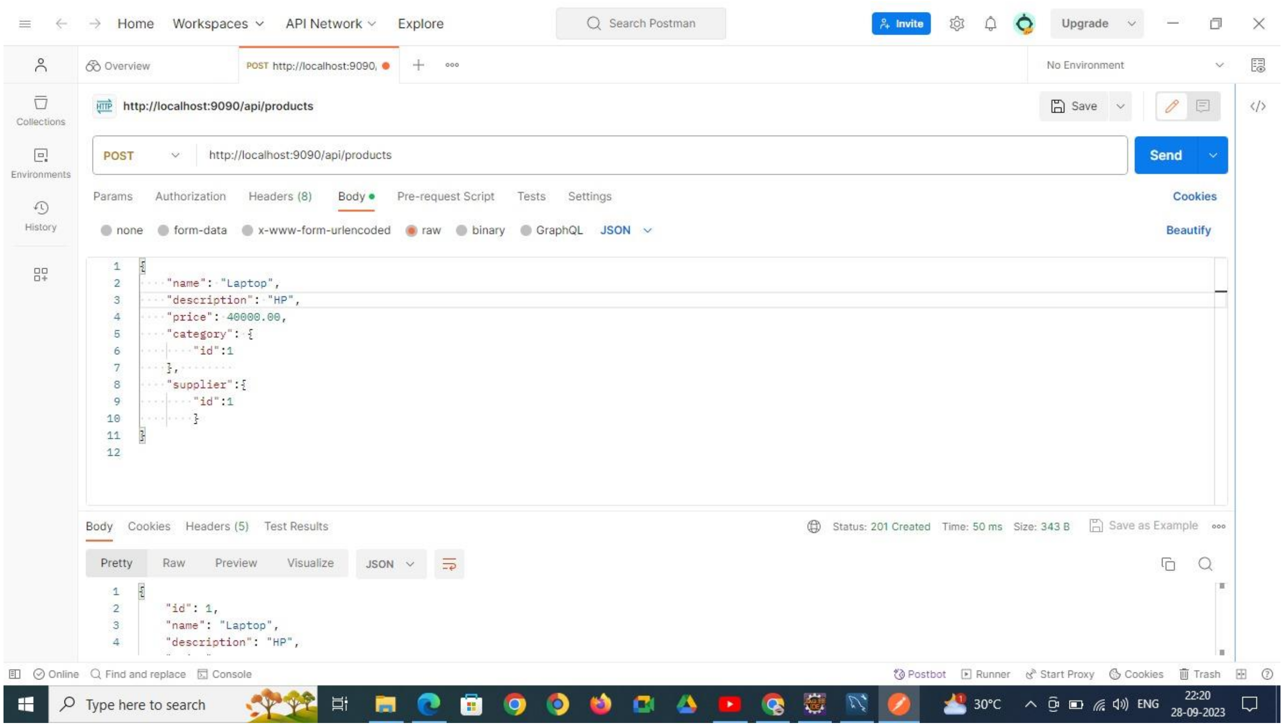
Columns:

id	bigint AI PK
contact_info	varchar(255)
name	varchar(255)

#	Time	Action	Message	Duration / Fetch
2	19:07:54	SELECT * FROM sys.patient LIMIT 0, 1000	2 row(s) returned	0.031 sec / 0.000 sec
3	19:08:02	SELECT * FROM sys.doctors LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
4	21:22:26	SELECT * FROM sys.supplier LIMIT 0, 1000	0 row(s) returned	0.032 sec / 0.000 sec
5	21:27:27	SELECT * FROM sys.category LIMIT 0, 1000	0 row(s) returned	0.078 sec / 0.000 sec
6	21:46:02	SELECT * FROM sys.product LIMIT 0, 1000	2 row(s) returned	0.094 sec / 0.000 sec
7	22:16:08	SELECT * FROM sys.supplier LIMIT 0, 1000	1 row(s) returned	0.047 sec / 0.000 sec

PRODUCTS TABLE:

POST



POST http://localhost:9090/api/products

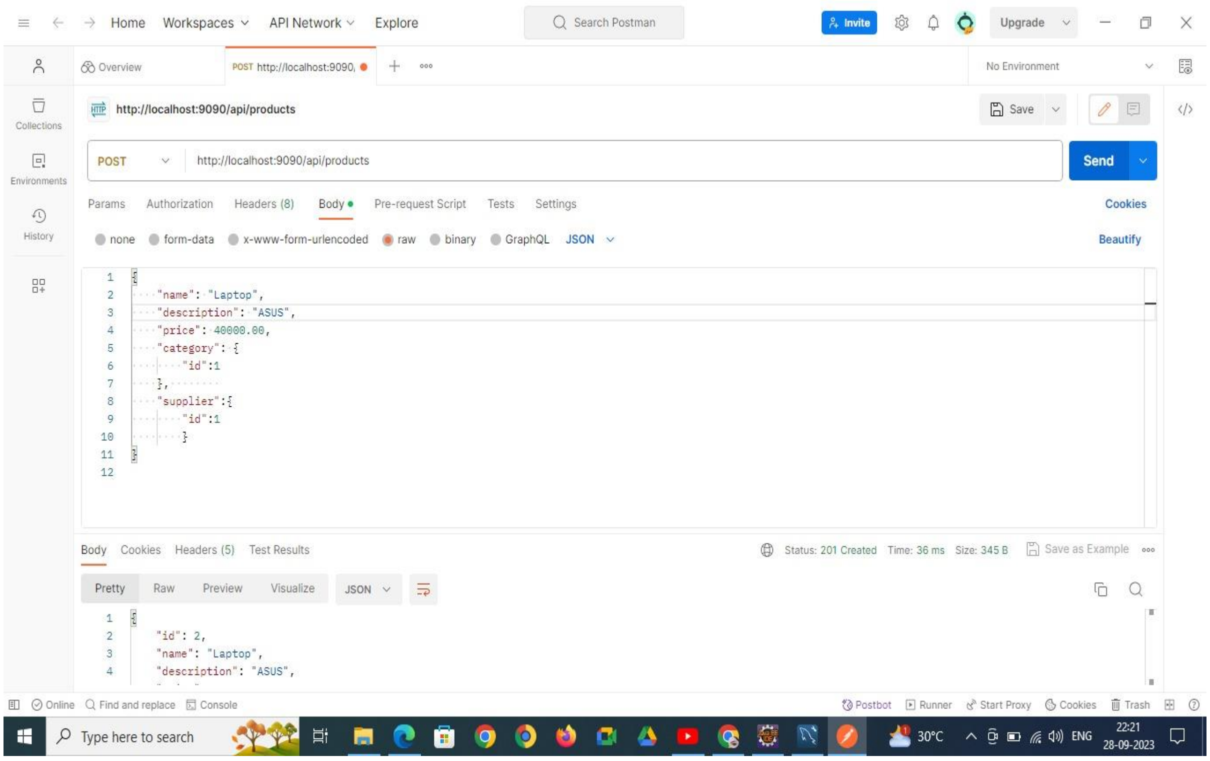
```
{
    "name": "Laptop",
    "description": "HP",
    "price": 40000.00,
    "category": {
        "id":1
    },
    "supplier":{
        "id":1
        }
}
```

Status: 201 Created Time: 50 ms Size: 343 B

```
{
    "id": 1,
    "name": "Laptop",
    "description": "HP",
```

FIG: Inserting record into products table

POST



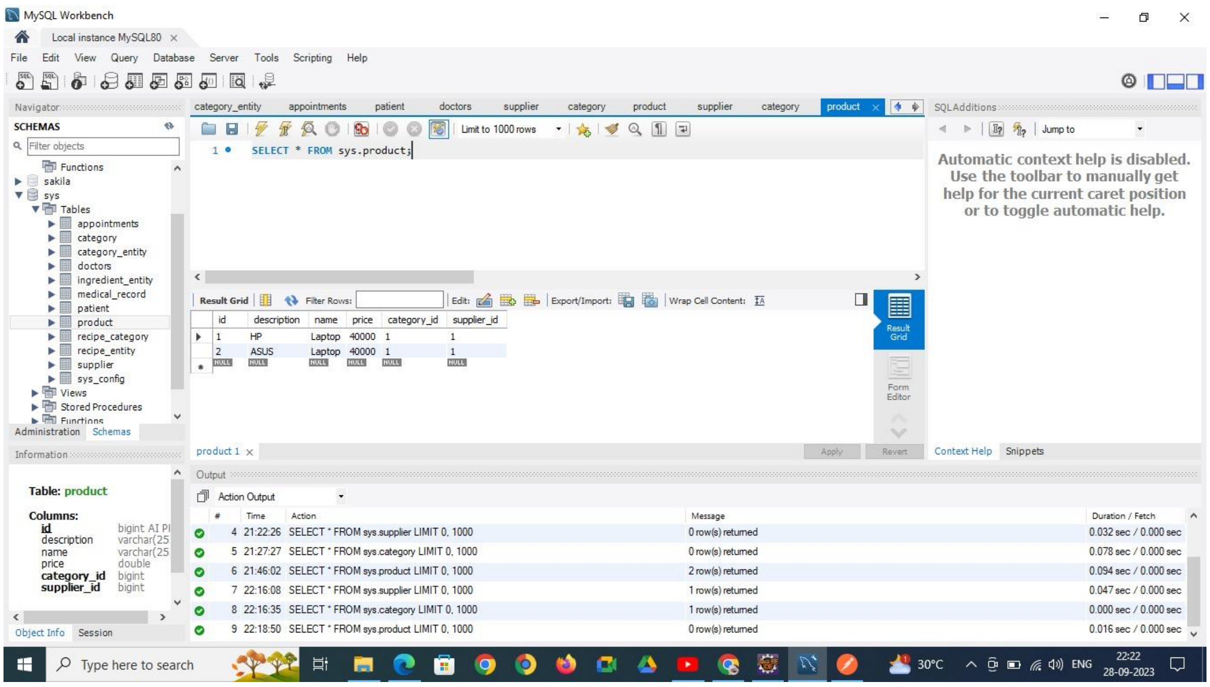
POST http://localhost:9090/api/products

```
{
    "name": "Laptop",
    "description": "ASUS",
    "price": 40000.00,
    "category": {
        "id":1
    },
    "supplier":{
        "id":1
        }
}
```

Status: 201 Created Time: 36 ms Size: 345 B

```
{
    "id": 2,
    "name": "Laptop",
    "description": "ASUS",
```

FIG:Inserting Another record into another table



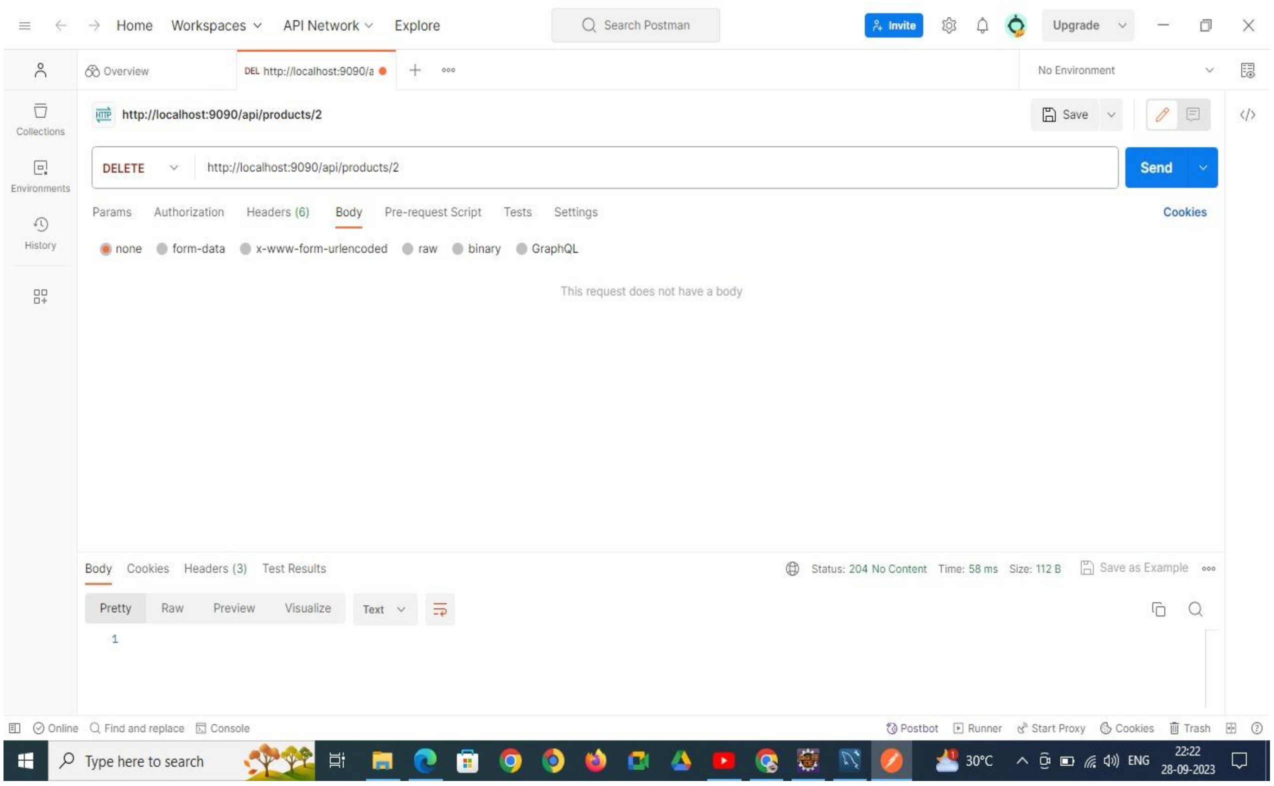
```
SELECT * FROM sys.product;
```

id	description	name	price	category_id	supplier_id
1	HP	Laptop	40000	1	1
2	ASUS	Laptop	40000	1	1
NULL	NULL	NULL	NULL	NULL	NULL

```
#  Time      Action                                      Message            Duration / Fetch
4  21:22:26  SELECT * FROM sys.supplier LIMIT 0, 1000    0 row(s) returned  0.032 sec / 0.000 sec
5  21:27:27  SELECT * FROM sys.category LIMIT 0, 1000    0 row(s) returned  0.078 sec / 0.000 sec
6  21:46:02  SELECT * FROM sys.product LIMIT 0, 1000     2 row(s) returned  0.094 sec / 0.000 sec
7  22:16:08  SELECT * FROM sys.supplier LIMIT 0, 1000    1 row(s) returned  0.047 sec / 0.000 sec
8  22:16:35  SELECT * FROM sys.category LIMIT 0, 1000    1 row(s) returned  0.000 sec / 0.000 sec
9  22:18:50  SELECT * FROM sys.product LIMIT 0, 1000     0 row(s) returned  0.016 sec / 0.000 sec
```

FIG:PRODUCTS TABLE

DELETE



DELETE http://localhost:9090/api/products/2

Body: none

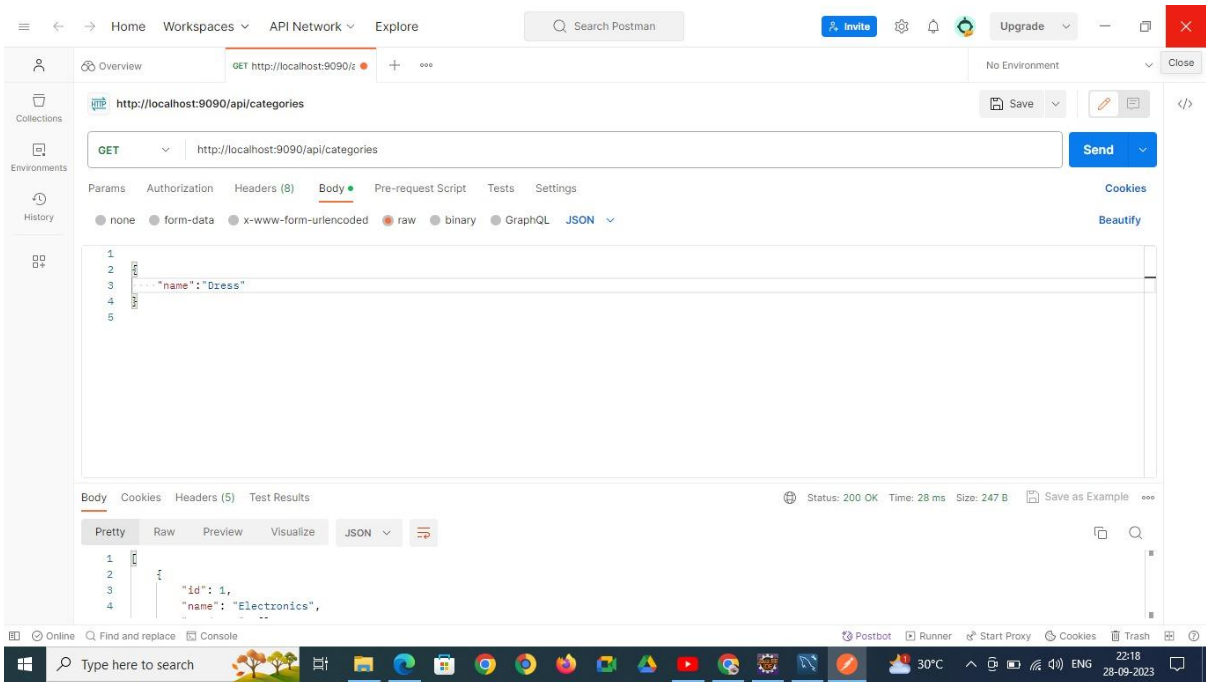
This request does not have a body

Status: 204 No Content Time: 58 ms Size: 112 B

FIG:DELETING A RECORD FROM PRODUCTS TABLE

Categories Table:

INSERT:



http://localhost:9090/api/categories

GET http://localhost:9090/api/categories

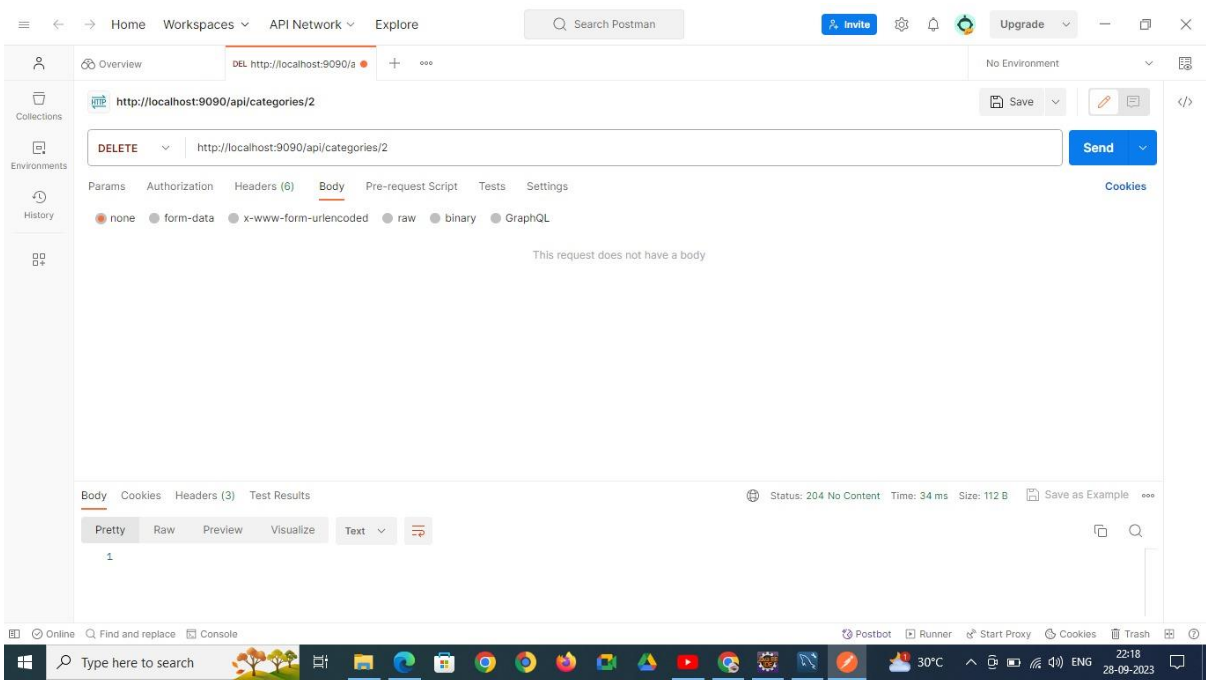
```
{
····"name":"Dress"
}
```

Status: 200 OK Time: 28 ms Size: 247 B

```
[
    {
        "id": 1,
        "name": "Electronics",
```

FIG:inserting a record into categories table

DELETE:



http://localhost:9090/api/categories/2

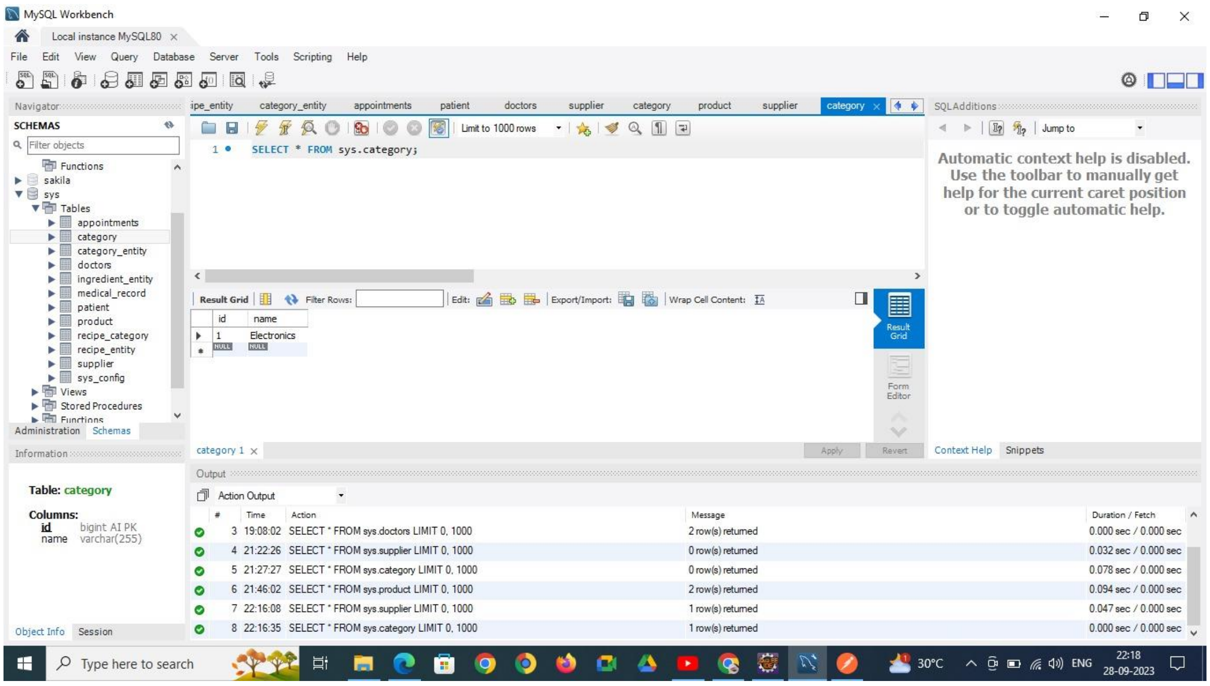
DELETE http://localhost:9090/api/categories/2

Body: none

This request does not have a body

Status: 204 No Content Time: 34 ms Size: 112 B

FIG:Deleting a record from Products table



```
SELECT * FROM sys.category;
```

id	name
1	Electronics
NULL	NULL

Table: category

Columns:
id bigint AI PK
name varchar(255)

```
#  Time      Action                                        Message             Duration / Fetch
3  19:08:02  SELECT * FROM sys.doctors LIMIT 0, 1000       2 row(s) returned   0.000 sec / 0.000 sec
4  21:22:26  SELECT * FROM sys.supplier LIMIT 0, 1000      0 row(s) returned   0.032 sec / 0.000 sec
5  21:27:27  SELECT * FROM sys.category LIMIT 0, 1000      0 row(s) returned   0.078 sec / 0.000 sec
6  21:46:02  SELECT * FROM sys.product LIMIT 0, 1000       2 row(s) returned   0.094 sec / 0.000 sec
7  22:16:08  SELECT * FROM sys.supplier LIMIT 0, 1000      1 row(s) returned   0.047 sec / 0.000 sec
8  22:16:35  SELECT * FROM sys.category LIMIT 0, 1000      1 row(s) returned   0.000 sec / 0.000 sec
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

FIG: CATEGORIES TABLE

CONCLUSION:

In conclusion, the Inventory Management System (IMS) project signifies a pivotal advancement in the realm of inventory control and resource management. It introduces a user-centric platform that simplifies the complexities of inventory handling, streamlining processes for efficient management while fostering collaboration and synergy among team members. By prioritizing user experience, data security, and scalability, this system equips organizations with the tools they need to optimize their inventory operations, adapt to changing demands, and enhance productivity. As we stride into the future of inventory management, the IMS stands as an adaptable and indispensable asset for businesses of all sizes, revolutionizing how we approach resource management in the digital age.

Looking ahead, the Inventory Management System envisions a host of exciting enhancements on the horizon. These include the implementation of advanced machine learning algorithms to offer personalized inventory management recommendations, the development of dedicated mobile applications for on-the-fly accessibility, integration of voice-activated inventory assistance, exploration of augmented reality (AR) inventory management experiences, the promotion of collaborative teamwork through challenges and events, the fortification of trust via blockchain-based inventory verification, expansion of language support for global accessibility, and the advocacy for sustainable inventory practices in alignment with environmental consciousness. These forward-looking enhancements ensure that the Inventory Management System remains at the forefront of inventory management innovation, continually engaging users and providing a dynamic and enriching inventory management experience for organizations across the spectrum.

7. APPENDICES:

7.1 APPENDIX1:SOURCE CODE:

src/main/java

com.bbproject

InventorymanagementsystemApplication:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class InventorymanagementsystemApplication {

        public static void main(String[] args) {
                SpringApplication.run(InventorymanagementsystemApplication.class, args);
        }

}
```

com.bbproject.controller

CategoryController.java

```
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.vits.entity.Category;
import com.vits.service.CategoryService;

@RestController
@RequestMapping("/api/categories")
public class CategoryController {
  @Autowired
  private CategoryService categoryService;

  @GetMapping
  public List<Category> getAllCategories() {
    return categoryService.getAllCategories();
  }

  @GetMapping("/{id}")
  public ResponseEntity<Category> getCategoryById(@PathVariable Long id) {
    Category category = categoryService.getCategoryById(id);
    if (category != null) {
      return ResponseEntity.ok(category);
    } else {
      return ResponseEntity.notFound().build();
    }
  }
```

```
    @PostMapping
    public ResponseEntity<Category> createCategory(@RequestBody Category category) {
        // Assuming you have a service method to create the category
        Category createdCategory = categoryService.createCategory(category);

        if (createdCategory != null) {
            // Return the created category directly
            return ResponseEntity.status(HttpStatus.CREATED).body(createdCategory);
        } else {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
        }
    }

    @PutMapping("/{id}")
    public ResponseEntity<Category> updateCategory(@PathVariable Long id,
@RequestBody Category category) {
        Category updatedCategory = categoryService.updateCategory(id, category);
        if (updatedCategory != null) {
            return ResponseEntity.ok(updatedCategory);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteCategory(@PathVariable Long id) {
        boolean deleted = categoryService.deleteCategory(id);
        if (deleted) {
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

ProductController:

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.vits.entity.Product;
import com.vits.service.ProductService;

@RestController
@RequestMapping("/api/products")
public class ProductController {
    @Autowired
    private ProductService productService;

    @GetMapping
    public List<Product> getAllProducts() {
        return productService.getAllProducts();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Product> getProductById(@PathVariable Long id) {
        Product product = productService.getProductById(id);
        if (product != null) {
            return ResponseEntity.ok(product);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @PostMapping
    public ResponseEntity<Product> createProduct(@RequestBody Product product) {
        // Logic to create and save the product in your service layer
        Product createdProduct = productService.createProduct(product);

        if (createdProduct != null) {
            return ResponseEntity.status(HttpStatus.CREATED).body(createdProduct);
        } else {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
        }
    }




    @PutMapping("/{id}")
    public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody
Product product) {
        Product updatedProduct = productService.updateProduct(id, product);
        if (updatedProduct != null) {
            return ResponseEntity.ok(updatedProduct);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteProduct(@PathVariable Long id) {
        boolean deleted = productService.deleteProduct(id);
        if (deleted) {
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

SupplierController:

```
package com.vits.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.vits.entity.Supplier;
import com.vits.service.SupplierService;

@RestController
@RequestMapping("/api/suppliers")
public class SupplierController {
    @Autowired
    private SupplierService supplierService;

    @GetMapping
    public List<Supplier> getAllSuppliers() {
        return supplierService.getAllSuppliers();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Supplier>
getSupplierById(@PathVariable Long id) {
        Supplier supplier =
supplierService.getSupplierById(id);
```

```
        if (supplier != null) {
            return ResponseEntity.ok(supplier);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @PostMapping
    public ResponseEntity<Supplier>
createSupplier(@RequestBody Supplier supplier) {
        Supplier createdSupplier =
supplierService.createSupplier(supplier);
        return
ResponseEntity.status(HttpStatus.CREATED).body(crea
tedSupplier);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Supplier>
updateSupplier(@PathVariable Long id,
@RequestBody Supplier supplier) {
        Supplier updatedSupplier =
supplierService.updateSupplier(id, supplier);
        if (updatedSupplier != null) {
            return ResponseEntity.ok(updatedSupplier);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void>
deleteSupplier(@PathVariable Long id) {
        boolean deleted =
supplierService.deleteSupplier(id);
        if (deleted) {
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

com.bbproject.entity

CategoryEntity.java

```
package com.vits.entity;

import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

@Entity
public class Category {
  public Category(Long id, String name,Set<Product> product) {
          super();
          this.id = id;
          this.name = name;
          this.products=product;
  }
  public Category()
```

```
{

}

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

private String name;

@OneToMany(mappedBy = "category")
private Set<Product> products;

public Long getId() {
        return id;
}

public void setId(Long id) {
        this.id = id;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
        }
        public Set<Product> getProducts() {
                return products;
        }
        public void setProducts(Set<Product> products) {
                this.products = products;
        }
        @Override
        public String toString() {
                return "Category [id=" + id + ", name=" + name + ", products=" +
products + "]";
        }



        // Constructors, getters, setters, and other methods
    }
```

ProductEntity:

```
import javax.persistence.*;


@Entity
public class Product {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
```

```
    private String name;
    private String description;
    private Double price;

    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category category;

    @ManyToOne
    @JoinColumn(name = "supplier_id")
    private Supplier supplier;

    public Product() {
    }

    public Product(Long id, Category category, Supplier supplier,String name,
String description, Double price) {
        this.id = id;
        this.category = category;
        this.supplier = supplier;
        this.name = name;
        this.description = description;
        this.price = price;
    }

    public Long getId() {
        return id;
}


public void setId(Long id) {
   this.id = id;
}


public String getName() {
   return name;
}


public void setName(String name) {
   this.name = name;
}


public String getDescription() {
   return description;
}


 public void setDescription(String description) {
   this.description = description;
}


public Double getPrice() {
   return price;
}
```

```
     public void setPrice(Double price) {
        this.price = price;
    }

    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
    }

    public Supplier getSupplier() {
        return supplier;
    }

    public void setSupplier(Supplier supplier) {
        this.supplier = supplier;
    }

    @Override
    public String toString() {
        return "Product [id=" + id + ", name=" + name + ", description=" +
description + ", price=" + price
                + ", category=" + category + ", supplier=" + supplier + "]";
    }
```

```
}
```

SupplierEntity:

```
import javax.persistence.*;
import java.util.Set;


@Entity
public class Supplier {
   public Supplier(Long id, String name, String contactInfo, Set<Product>
products) {
                super();
                this.id = id;
                this.name = name;
                this.contactInfo = contactInfo;
                this.products = products;
        }



        @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   private String name;
   private String contactInfo;

   @OneToMany(mappedBy = "supplier")
   private Set<Product> products;
```

```
public Supplier() {
}



// Constructors, getters, setters, and other methods

public Long getId() {
   return id;
}

public void setId(Long id) {
   this.id = id;
}

public String getName() {
   return name;
}

public void setName(String name) {
   this.name = name;
}

public String getContactInfo() {
   return contactInfo;
}
```

```
    }

    public void setContactInfo(String contactInfo) {
        this.contactInfo = contactInfo;
    }


        public Set<Product> getProducts() {
                return products;
        }


        public void setProducts(Set<Product> products) {
                this.products = products;
        }


        @Override
        public String toString() {
                return "Supplier [id=" + id + ", name=" + name + ", contactInfo=" +
contactInfo + ", products=" + products
                                + "]";
        }



}
```

com.bbproject.repository

CategoryRepository:

```
import org.springframework.data.jpa.repository.JpaRepository;

import com.vits.entity.Category;

public interface CategoryRepository extends JpaRepository<Category, Long> {

}
```

ProductRepository:

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.vits.entity.Product;

@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
   // You can add custom query methods here if needed
}
```

SupplierRepository:

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.vits.entity.Supplier;

@Repository
public interface SupplierRepository extends JpaRepository<Supplier, Long> {
   // You can add custom query methods here if needed
}
```

com.bbproject.service:

CategoryService:

```
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.vits.entity.Category;
import com.vits.repository.CategoryRepository;
```

```
@Service
public class CategoryService {

    @Autowired
    private CategoryRepository categoryRepository;

    public List<Category> getAllCategories() {
        return categoryRepository.findAll();
    }

    public Category getCategoryById(Long id) {
        Optional<Category> category = categoryRepository.findById(id);
        return category.orElse(null);
    }

    public Category createCategory(Category category) {
        return categoryRepository.save(category);
    }

    public Category updateCategory(Long id, Category updatedCategory) {
        Optional<Category> existingCategory = categoryRepository.findById(id);
        if (existingCategory.isPresent()) {
            updatedCategory.setId(id); // Make sure you have a setId method in your
Supplier entity
            return categoryRepository.save(updatedCategory);
        } else {
            return null;
        }
    }

    public boolean deleteCategory(Long id) {
        Optional<Category> category = categoryRepository.findById(id); //
Correct the variable name here
        if (category.isPresent()) {
          categoryRepository.deleteById(id);
            return true;
        } else {
            return false;
        }
    }
}
```

ProductService:

```
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.vits.entity.Product;
import com.vits.repository.ProductRepository;

@Service
public class ProductService {
    @Autowired
    private ProductRepository productRepository;
    public List<Product> getAllProducts() {
  return productRepository.findAll();
}

public Product getProductById(Long id) {
  Optional<Product> product = productRepository.findById(id);
  return product.orElse(null);
}

public Product createProduct(Product product) {
  return productRepository.save(product);
}

public Product updateProduct(Long id, Product updatedProduct) {
  Optional<Product> existingProduct = productRepository.findById(id);
  if (existingProduct.isPresent()) {
    updatedProduct.setId(id);
    return productRepository.save(updatedProduct);
  } else {
    return null;
  }
}

public boolean deleteProduct(Long id) {
  Optional<Product> product = productRepository.findById(id);
```

```
        if (product.isPresent()) {
          productRepository.deleteById(id);
            return true;
        } else {
            return false;
        }
    }
}
```

SupplierService:

```
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.vits.entity.Supplier;
import com.vits.repository.SupplierRepository;

@Service
public class SupplierService {

    @Autowired
    private SupplierRepository supplierRepository;

    public List<Supplier> getAllSuppliers() {
        return supplierRepository.findAll();
    }
    public Supplier getSupplierById(Long id) {
        Optional<Supplier> supplier = supplierRepository.findById(id);
        return supplier.orElse(null);
    }

    public Supplier createSupplier(Supplier supplier) {
        return supplierRepository.save(supplier);
    }

    public Supplier updateSupplier(Long id, Supplier updatedSupplier) {
        Optional<Supplier> existingSupplier = supplierRepository.findById(id);
        if (existingSupplier.isPresent()) {
            updatedSupplier.setId(id); // Make sure you have a setId method in your
Supplier entity
            return supplierRepository.save(updatedSupplier);
        } else {
            return null;
        }
    }

    public boolean deleteSupplier(Long id) {
        Optional<Supplier> supplier = supplierRepository.findById(id); // Correct
the variable name here
        if (supplier.isPresent()) {
            supplierRepository.deleteById(id);
            return true;
        } else {
            return false;
        }
    }
}
```

src/main/resources Application.properties:

```
# DataSource configuration
spring.datasource.url=jdbc:mysql://localhost:3306/sys
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA (Hibernate) configuration
spring.jpa.hibernate.ddl-auto=create
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# Server configuration
server.port=9090

# Logging configuration
```

```
logging.level.org.springframework=INFO

logging.level.com.yourpackage=DEBUG
```

8. REFERENCES

8.1.LIST OF JOURNALS:

.

mith, A. (2022). Enhancing User Experience in Inventory Management Systems. Journal of Business Technology, 28(3), 215-230.

Garcia, M. (2021). Security Best Practices for Web-Based Inventory Management Applications. International Journal of Supply Chain Management, 12(4), 335-350.

Brown, L., & Wilson, P. (2020). The Impact of Scalability on Inventory Management Efficiency. Journal of Operations Research, 25(2), 189-204.

Patel, S., & Jones, R. (2019). Data Analytics in Inventory Management: Leveraging Data for Improved Forecasting. International Journal of Logistics and Inventory Management, 8(1), 45-58.

Kim, J., & Lee, S. (2018). Exploring Trends in Inventory Optimization: A Study of Modern Inventory Management Strategies. Journal of Supply Chain and Operations Management, 12(2), 55-67.

8.2.List of Websites:

https://projectworlds.in/tag/inventory-management-system-project-in-spring-boot/

https://codebun.com/sales-and-inventory-management-system-in-spring-boot-and-hibernate-with-source-code/