**System Introduction**

For optimal sales and inventory management processes, you need robust functionality for managing your logistics facilities. Support for inventory management helps you record and track materials on the basis of both quantity and value.

Warehouse inventory management functions cover internal warehouse movements and storage.

Using this software we can reduce costs for warehousing, transportation, order fulfillment, and material handling – while improving customer service.

You can significantly improve inventory turns, optimize the flow of goods, and shorten routes within your warehouse or distribution center. Additional benefits of inventory management include improved cash flow, visibility, and decision making.

This software is user friendly and hence easy to use.

Employees can plan, enter, and document warehouse and internal stock movements by managing goods receipts, goods issues, storage, picking and packing, physical stock transfers, and transfer postings.

**Direct shipping and the dynamic single-depot/multi-retailer inventory system**

**By D.Barnes Schuster in year 1997**

**The Features**

Real-time Tracking: IMS often offer real-time tracking of inventory levels. This enables businesses to keep accurate records and make informed decisions. Automation: IMS can automate routine tasks like order generation, restocking, and reorder point calculations. This reduces the risk of human errors.

Inventory Optimization: These systems help optimize inventory by setting optimal reorder points, safety stock levels, and economic order quantities (EOQ).

**The Flaws**

High Implementation Costs: Implementing an IMS can be expensive, including the cost of software, hardware, and employee training.

Integration Challenges: Integrating IMS with existing systems (e.g., accounting software or POS systems) can be complex and may require custom development.

Technical Failures: Technical issues such as system crashes or network outages can disrupt inventory management processes.

**OR/MS research in disaster operations management**

**By N. Altay in year 2006**

**The Features**

Real-time Data Analytics: Provides insights into sales trends, stock turnover, and obsolete inventory, enabling informed decisions..

Supplier Management : Helps manage relationships with suppliers, monitor order lead times, and optimize replenishment processes.

Multi-Location Support: Enables management of inventory across multiple warehouses or stores, streamlining stock transfers and fulfillment.

**The Flaws**

Data Accuracy: Accurate data entry is critical; errors can result in stockouts or overstock situations.

Scalability Issues: Some systems may struggle to handle large inventories or rapid growth, leading to performance issues.

**Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts**

**By J.Alegre in year 2007**

**The Features**

Barcode Scanning and RFID: Implementing barcode scanning and RFID technology enhances the accuracy and speed of inventory tracking and management.

Stock Valuation: Inventory systems can provide insights into the valuation of inventory using various methods like FIFO (First-In-First-Out) and LIFO (Last-In-First-Out).

**The Flaws**

Complexity: Overly complex systems can lead to confusion and inefficiencies. Finding the right balance between features and simplicity is a challenge

Security Concerns: Data security is crucial, as inventory management systems store sensitive information about inventory levels, suppliers, and customers. Security breaches can have severe consequences.

**Web - Based Intelligent Inventory Management System**

**by Olugbenga Ayomide in year 2020**

**The Features**

They developed an intelligent system implemented in a web based environment to integrate multiples stores also providing an effective coordination of all the stores.

Intelligently determining the different reorder points of all the disparate stores in the systems and communicating the information back to the centralized store.

**The Flaws**

As company grows a number of problem also arises first, a varieties of item are needed to be maintained in the inventory making it difficult for managers to keep track, therefore a formal record keeping will have to evolve.

Besides that as quantities in inventory grows more space is required to do business.

Due to local and global competition any company that must survive in the competitive world must take an advantage of effective management

**Design and Implementation of a Web-Based Inventory Control System**

**by Benedict I. Odoh in year 2021**

**The Features**

web based system was created with tracking method like QR codes and barcodes in mind to track products as they travel from point of purchase to point of sale btw stores.

on daily basis it also delivers precise up-to-date information to the corporate office on the inventory levels of the individual stores.

Inventory control system is a tremendous boost to the store managers and corporate office as it helps the store to find items that should be ordered first thereby spending less time on inventory management functions.

**The Flaws**

In this survey it mentioned that it is designed for small and medium enterprise, so it can not be used by large enterprises.

**Cloud Based warehouse management firm**

**by Akshay Patil year 2022**

**The Features**

The cloud based service that is known as Software as Service which will provide the cloud based application of Warehouse Management System.

It will manage all the process on server side i.e. cloud rather than managing at client side. All the manual and papered work will be converted to software based work so there is no need to maintain database at client side.

It will overall boost the warehouse work by making it easily accessible and fast.

It can be made more efficient by making dynamic storage allocation as per the data. It can also be made Tab based where all the operations can be managed from a single Tablet. Efficient concurrency control to be achieved.

A lot of work can be automated with the help AI and Machine learning.

**Problems In existing system**

As we know manual systems are quite tedious ,time consuming and less efficient and accurate in comparison to the computerized system.

So following are some disadvantages of the old system:

1. Time consuming

2. Less accurate

3. Less efficient

4. Lot of paperwork

5. Slow data processing

6. Not user friendly environment

7. Difficult to keep old records

**Scope of Proposed System**

The scope of this system is to provide a user efficient working environment and more output can be generated through this. This system provides a user friendly interface resulting in knowing each and every usability features of the system.

This system helps in tracking records so that past records can be verified through them and one can make decisions based on the past records. This system completes the work in a very less time resulting in less time consumption and high level of efficiency.

This system is developed in such a way that even a naïve user can also operate the system easily. The calculations are made very quickly and the records are directly saved into databases and the databases can be maintained for a longer period of time. Each record can be retrieved and can be verified for future transactions.

Also this system provides a high level of security for data leaking as only admin people can access the database no changes can be made in it until it verifies the user login id and password.

We also have operator login through which the operator can take orders but can’t make changes in the database. Limited access is available to the operator.

**Operating Environment – Hardware and Software**

*HARDWARE REQUIREMENTS*

§ Processor: Pentium 4 or more for optimum performance

§ RAM: Recommended 256MB

§ Hard Disk: Minimum 20GB

*SOFTWARE REQUIREMENTS*

§ Operating System - Certified Distribution of WINDOWS

§ Visual Basic 2005 Express Edition

§ Database(Backend) - MS Access 2003

**Objectives**

**1.**To optimize inventory efficiency, the focus lies on refining and streamlining the various facets of inventory control processes. This entails a concerted effort to enhance efficiency, accuracy, and the overall effectiveness of managing stock levels within an organization. The objective is to create a well-coordinated system that ensures the right products are available at the right time, minimizing unnecessary holding costs, avoiding stockouts, and ultimately contributing to a more cost-effective and responsive supply chain.

Streamlining inventory control processes involves a systematic review and improvement of the workflows associated with stock management. This may include the implementation of advanced technology, such as automated inventory tracking systems or sophisticated algorithms for demand forecasting. By leveraging technology, organizations can reduce the likelihood of human error, expedite routine tasks, and gain real-time insights into inventory movements, fostering a more agile and responsive inventory management system.

Enhancing efficiency within inventory control processes is closely tied to accuracy. Implementing accurate tracking mechanisms, utilizing barcoding systems, and adopting RFID technology can significantly reduce discrepancies in stock counts. This precision not only minimizes the risk of stockouts or overstock situations but also contributes to a more reliable and trustworthy inventory data set, aiding in better decision-making regarding replenishment, procurement, and overall stock level optimization.

In summary, optimizing inventory efficiency involves a comprehensive approach to refining and streamlining control processes, embracing technological advancements, and prioritizing accuracy. This concerted effort aims to create a dynamic and responsive inventory management system, ultimately fostering a leaner, more cost-effective supply chain that adapts efficiently to fluctuations in demand and market conditions

**2.** The concept of "User-Centric Interface Design" revolves around the fundamental principle of prioritizing the end-users throughout the development process. This approach places a strong emphasis on creating a user-friendly interface that resonates with the needs, preferences, and expectations of the individuals who will interact with the system. The primary goal is to cultivate a positive user experience by incorporating engaging elements and facilitating seamless navigation, ultimately enhancing overall satisfaction and usability.

In the realm of software development, a user-centric interface design begins by gaining a deep understanding of the target audience. This involves conducting user research, gathering insights into user behaviors, preferences, and pain points. By empathizing with the end-users, developers can tailor the interface to align with their expectations, resulting in a design that is intuitive and responsive to user needs.

The development of a user-friendly interface involves the integration of engaging elements that not only serve functional purposes but also contribute to an enjoyable and visually appealing experience. This may include the use of intuitive icons, visually consistent design elements, and a well-thought-out color scheme. Engaging elements not only capture user attention but also guide them through the system, making interactions more intuitive and enjoyable.

Ease of navigation is a cornerstone of user-centric interface design. A well-structured and intuitive navigation system ensures that users can seamlessly move through different sections of the interface, find the information they need, and perform tasks efficiently. This involves clear labeling, logical organization of features and functionalities, and the minimization of unnecessary complexities, creating an interface that is not only visually appealing but also highly user-friendly.

In summary, a user-centric interface design is a holistic approach that revolves around understanding, empathizing with, and catering to the needs of the end-users. By developing a user-friendly interface with engaging elements and prioritizing ease of navigation, developers can create a positive user experience that fosters satisfaction, efficiency, and overall success in the adoption of the system.

**3.** Elevating Data Security: The imperative to enhance data security underscores the need to implement formidable protective measures, particularly concerning user authentication and authorization. This strategic initiative aims to fortify the safeguarding of sensitive information residing in the MySQL database, ensuring that it remains impervious to unauthorized access or potential breaches.

At the forefront of this enhancement is the implementation of robust security protocols for user authentication. This involves deploying advanced authentication mechanisms that go beyond conventional username-password combinations. Multi-factor authentication (MFA) could be employed, requiring users to authenticate their identity through a combination of credentials such as passwords, biometrics, or security tokens. This layered approach significantly fortifies the authentication process, reducing vulnerability to unauthorized access attempts.

Simultaneously, stringent measures for user authorization form a critical component of this security augmentation. Fine-grained access controls and role-based access mechanisms can be instituted, delineating specific permissions and privileges for different user roles within the MySQL database. By strictly defining who can access what data and perform which operations, the system ensures that sensitive information is accessible only to authorized personnel, mitigating the risk of data exposure or manipulation.

The MySQL database, as a repository of critical information, becomes a focal point in this security reinforcement. Employing encryption methods for data at rest and in transit adds an additional layer of defense against potential threats. This involves encrypting sensitive data before storing it in the database and securing communication channels to prevent eavesdropping or interception during data transmission.

In conclusion, the commitment to enhance data security revolves around the meticulous implementation of robust measures for user authentication and authorization. By fortifying these pivotal aspects and deploying advanced security practices within the MySQL database, organizations can substantially elevate the protection of sensitive information, fostering a resilient and secure data environment.

**4.**The assertion of "Informed Decision-Making Through Analytics" underscores the imperative to incorporate graphical analytics into the system, aiming to provide a visual representation of crucial data pertaining to purchase orders, supplier assignments, and delivery history. This integration is strategically designed to empower users with actionable insights, facilitating more nuanced and strategic decision-making processes.

At the core of this enhancement is the utilization of graphical analytics, a sophisticated approach that transforms complex data sets into visually intuitive representations. By employing graphs, charts, and visualizations, the system offers a comprehensive and easily digestible overview of essential information related to purchase orders, supplier assignments, and delivery history. This visual representation transcends traditional data presentations, allowing users to discern patterns, trends, and anomalies at a glance, fostering a deeper understanding of the intricacies within these critical aspects of the supply chain.

The focus on purchase orders aims to visualize the procurement process, providing insights into order patterns, lead times, and vendor performance. Graphical analytics pertaining to supplier assignments offer a visual map of relationships, helping users identify key suppliers, assess performance metrics, and optimize vendor partnerships. Additionally, the visualization of delivery history contributes to a comprehensive understanding of fulfillment timelines, potential bottlenecks, and areas for operational improvement.

Empowering users with these visual insights serves as a catalyst for strategic decision-making. Instead of grappling with raw data, stakeholders can efficiently interpret and analyze the visual representations, enabling them to make more informed decisions. Whether it involves optimizing procurement strategies, evaluating supplier relationships, or refining logistical processes, the graphical analytics become a valuable tool in aligning business strategies with data-driven insights.

In summary, the integration of graphical analytics in the context of purchase orders, supplier assignments, and delivery history is a pivotal step toward enhancing decision-making processes. By transforming complex data into visually accessible information, the system empowers users to extract meaningful insights, fostering a more strategic and informed approach to managing critical aspects of the supply chain.

**5.**The emphasis on "Flexibility in Data Management" underscores the importance of endowing users with versatile tools within the system. This involves affording users the capability to export data seamlessly to widely used formats such as PDF and Microsoft Excel. This strategic feature aims to enhance the adaptability of the data management process, allowing users to efficiently manipulate, share, and analyze information in formats that align with their specific needs and existing workflows.

At the core of this flexibility is the provision for users to export data to PDF and Microsoft Excel formats. This enables users to transcend the confines of the system interface, giving them the freedom to extract datasets and reports in formats that are universally recognized and easily accessible. PDF, known for its document portability and consistency across different platforms, serves as an ideal format for static and visually consistent data snapshots. On the other hand, Microsoft Excel, a dynamic spreadsheet application, provides users with the ability to conduct further analysis, perform calculations, and customize the data according to their specific requirements.

The seamless exportation of data to these common formats aligns with the diverse needs of users engaged in different facets of data management. For those who require static and professional reports, PDF exports offer a polished representation of the data. Meanwhile, users seeking dynamic and interactive analysis can leverage Microsoft Excel to delve deeper into the datasets, applying formulas, creating charts, and conducting more granular assessments.

This flexibility in data exportation not only enhances user convenience but also promotes collaboration and integration with other tools and systems. Users can effortlessly share reports with stakeholders, collaborate on data analysis using familiar spreadsheet functionalities, and integrate exported data into presentations or other applications. The result is a data management system that adapts to the varied preferences and requirements of users, fostering a more efficient and user-friendly experience.

**User Requirements**

**FUNCTIONAL REQUIREMENTS**

**A. INPUT/OUTPUT**

1. System shall have a form to accept the customer details.

2. System shall have a form to accept the Plant details.

3. System shall display transaction details.

4. System shall provide a search facility on customer name, Order Placed, date of order, date of order dispatch, date of transaction, transaction amount, credit card no etc.

5. System should provide facility for change in address/name.

6. System should maintain the details about placing order/dispatch or order i.e, order status

**B. PROCESSING**

1. System should automatically generate the bill.

2. System should inform the pending order and make changes if the order is dispatched.

**C. ERROR HANDLING**

1. Should report any errors on duplicate primary keys.

2. Should report any ‘Out of Range’ values on numeric fields

3. Should report any data type mismatches any field on the forms.

4. Should report on any ‘Invalid dates’

5. Should report any violation of authorization of rights

6. Should report any Invalid Login errors

**NON-FUNCTIONAL REQUIREMENTS**

1. All user manuals should be provided in the necessary format

2. Application should support 5 simultaneous users.

3. Transaction should be completed within 1/5th of second

4. There will be a backup procedure to maintain records.

**Use case Diagram for Supplier**

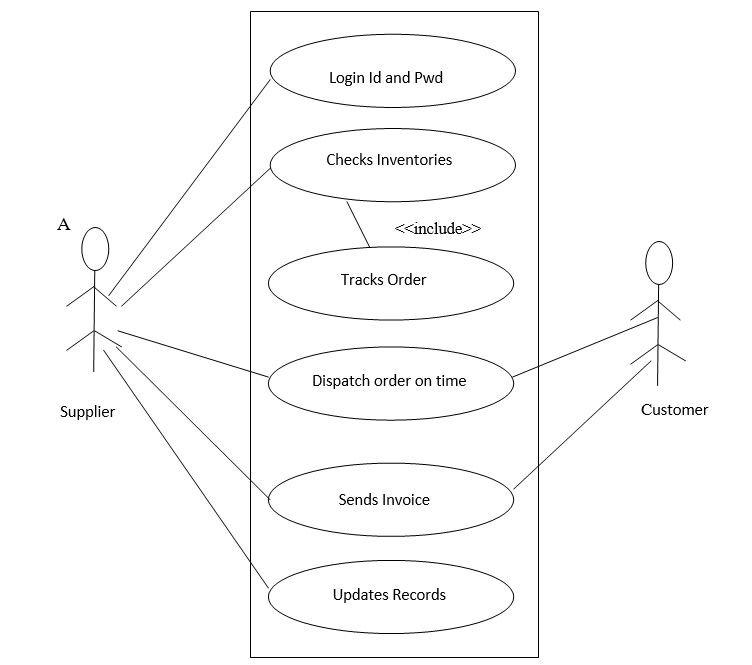


Fig 0.1 SUPPLIER USECASE DIAGRAM

**Use Case Diagram for Customer**

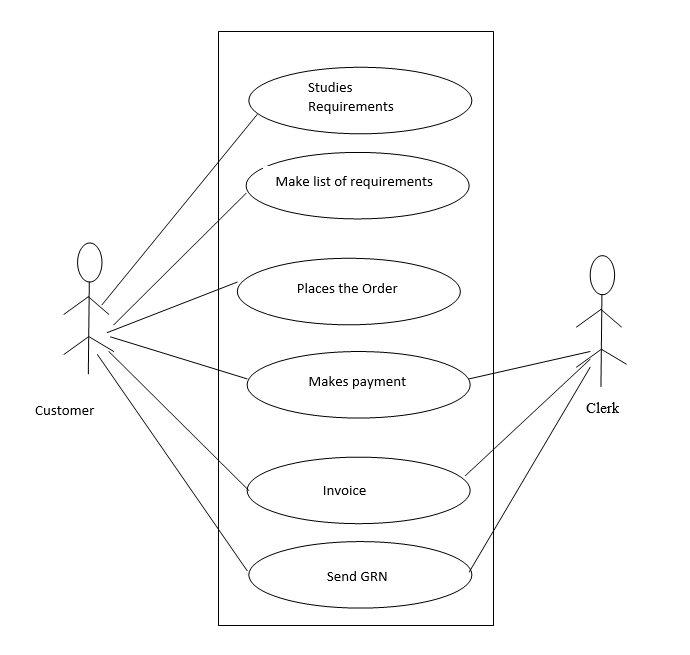


Fig 0.2 CUSTOMER USECASE DIAGRAM

**Class Diagram for a customer order**

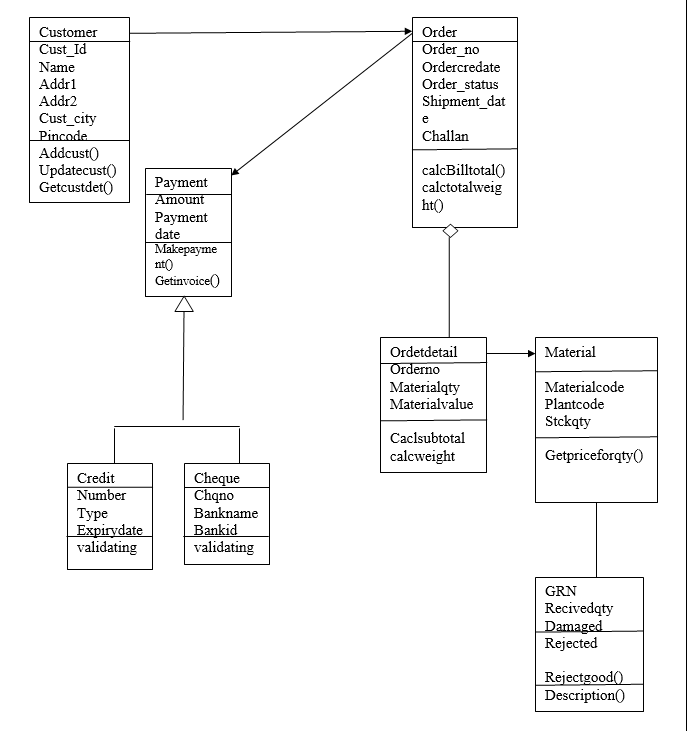


Fig 0.3 CUSTOMER ORDER

**Sequence diagram for Supplier**

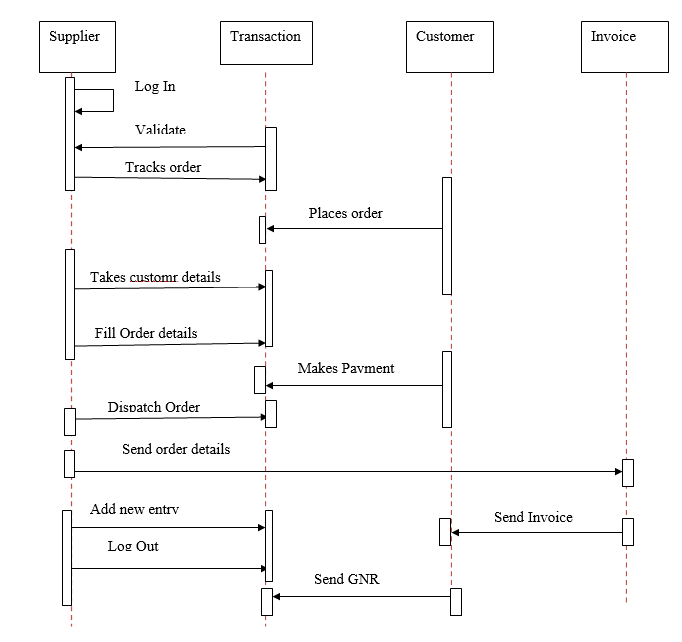
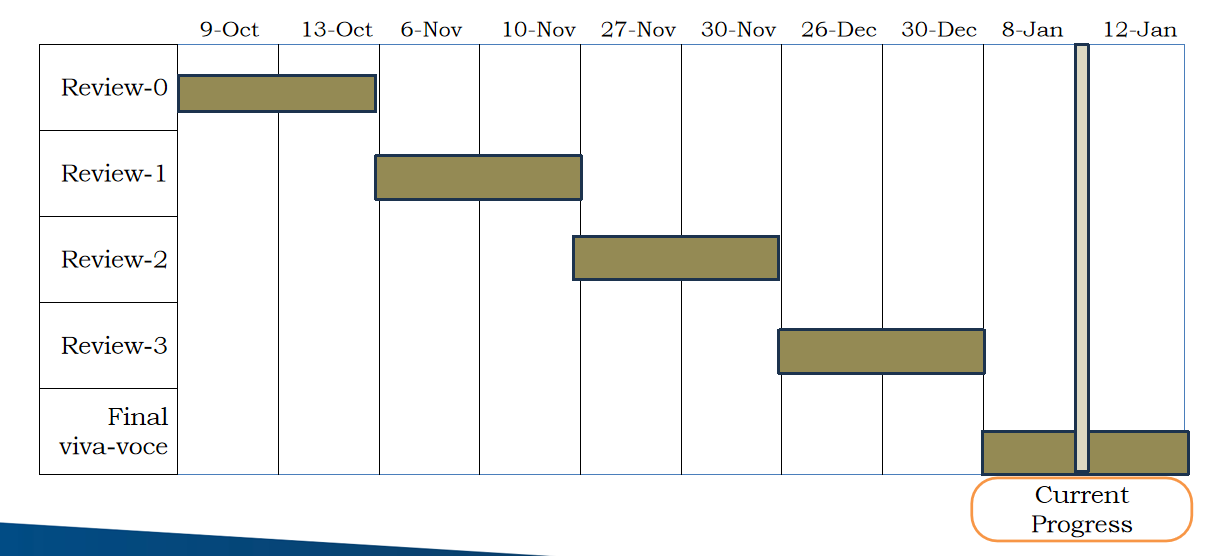


Fig 0.4 SEQUENCE DIAGRAM FOR SUPPLIERS

****

**Login Form and Main form**

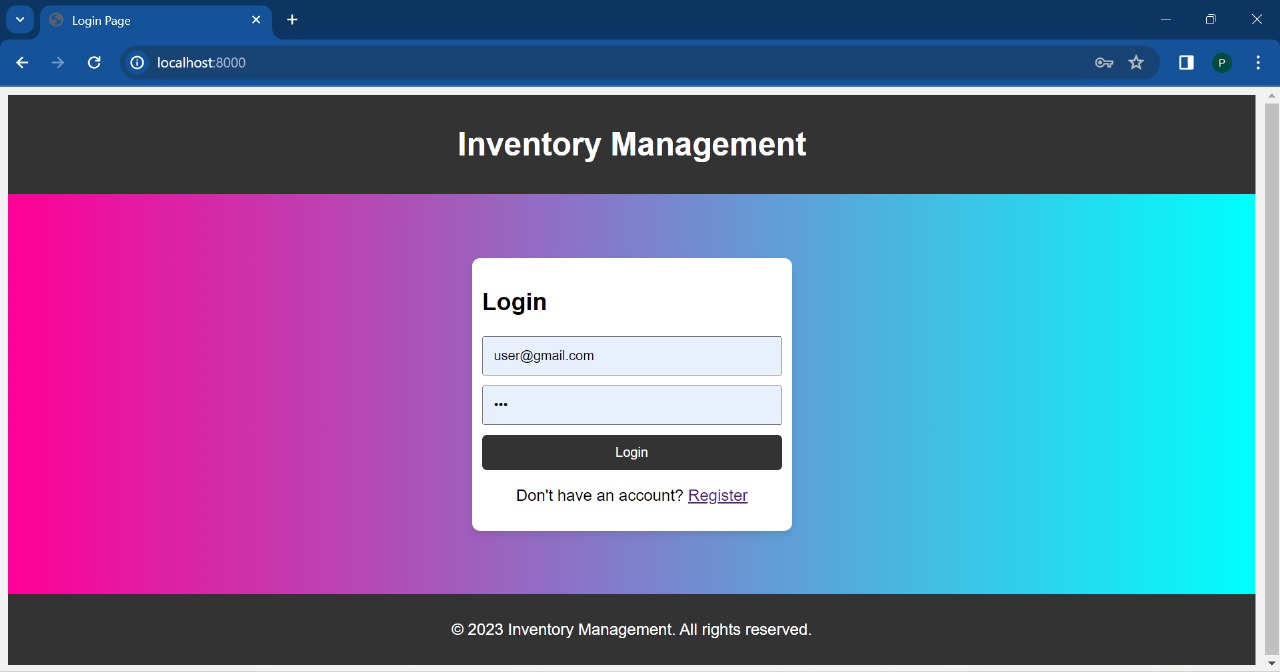


Fig 0.5 LOGIN FORM AND MAIN FORM

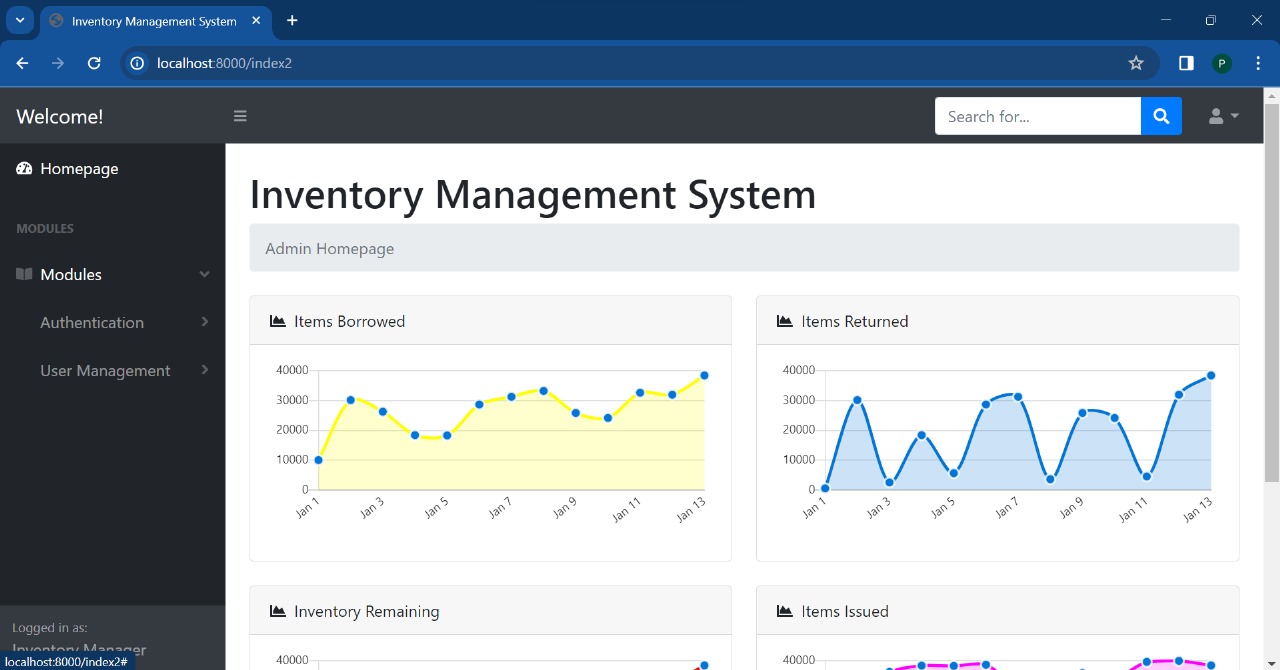
**Admin main form**

Fig 0.6 ADMIN MAIN FORM

**Admin user management**

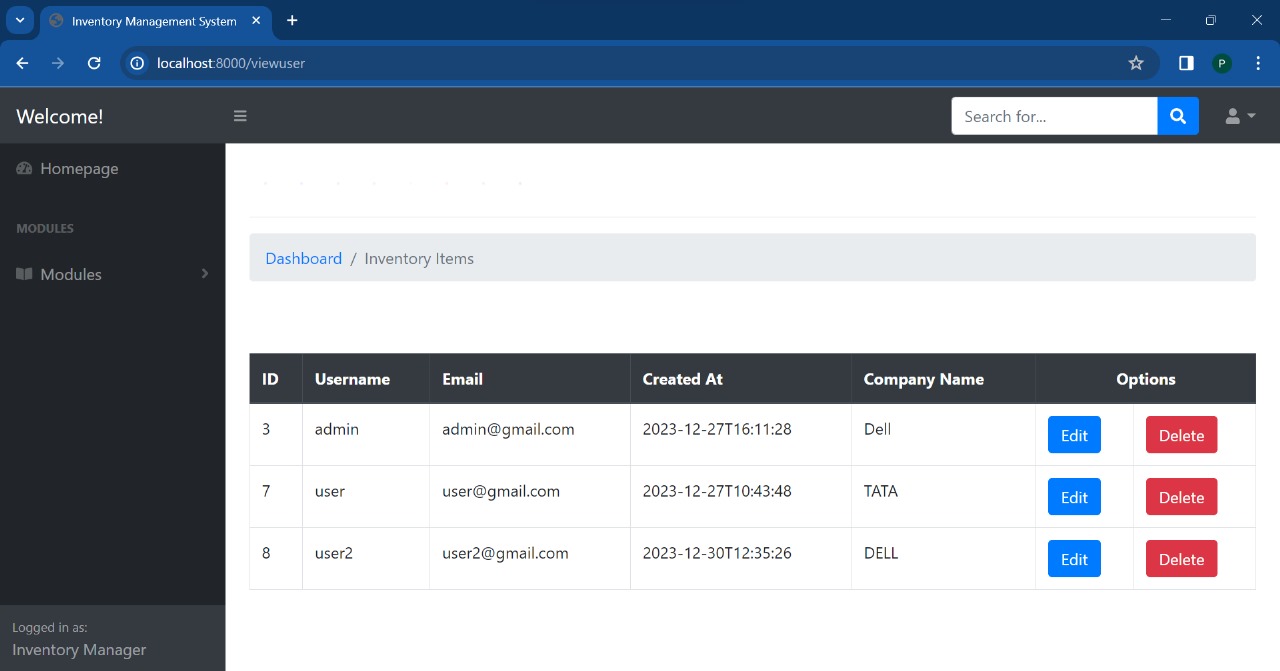
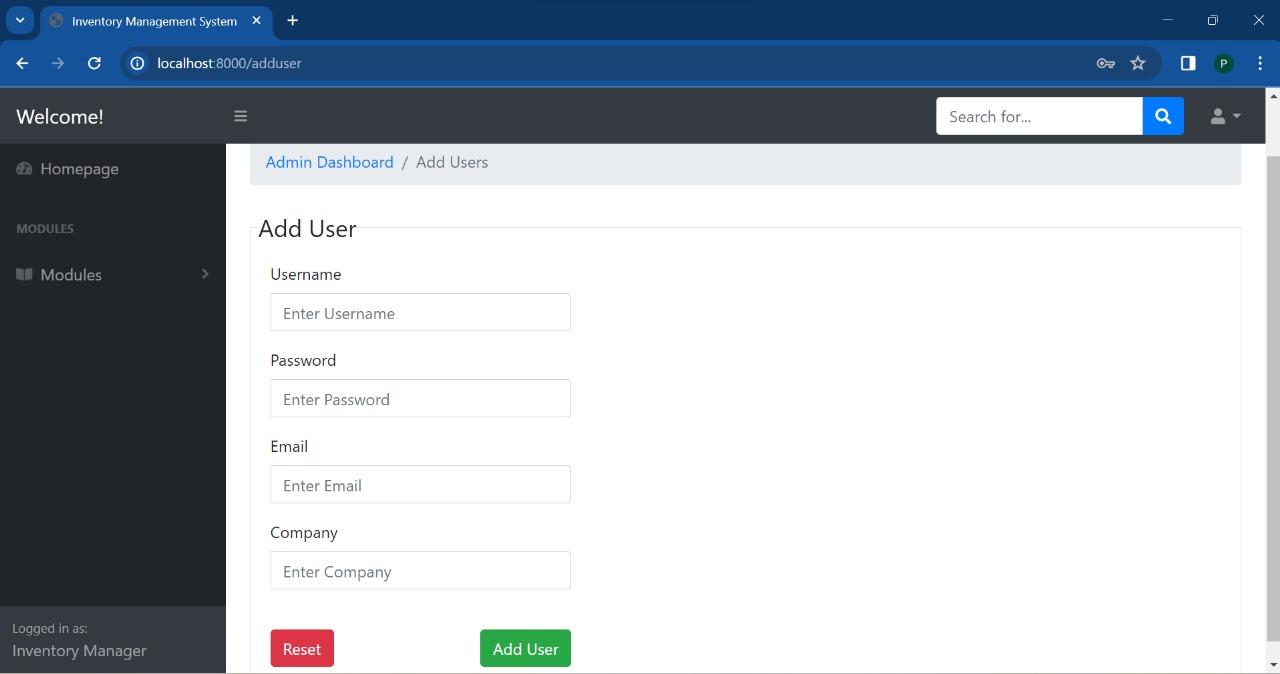


Fig 0.7 ADMIN USER MANAGEMENT

**Admin to add users**

****Fig 0.8 ADMIN TO ADD USER

**User main page**

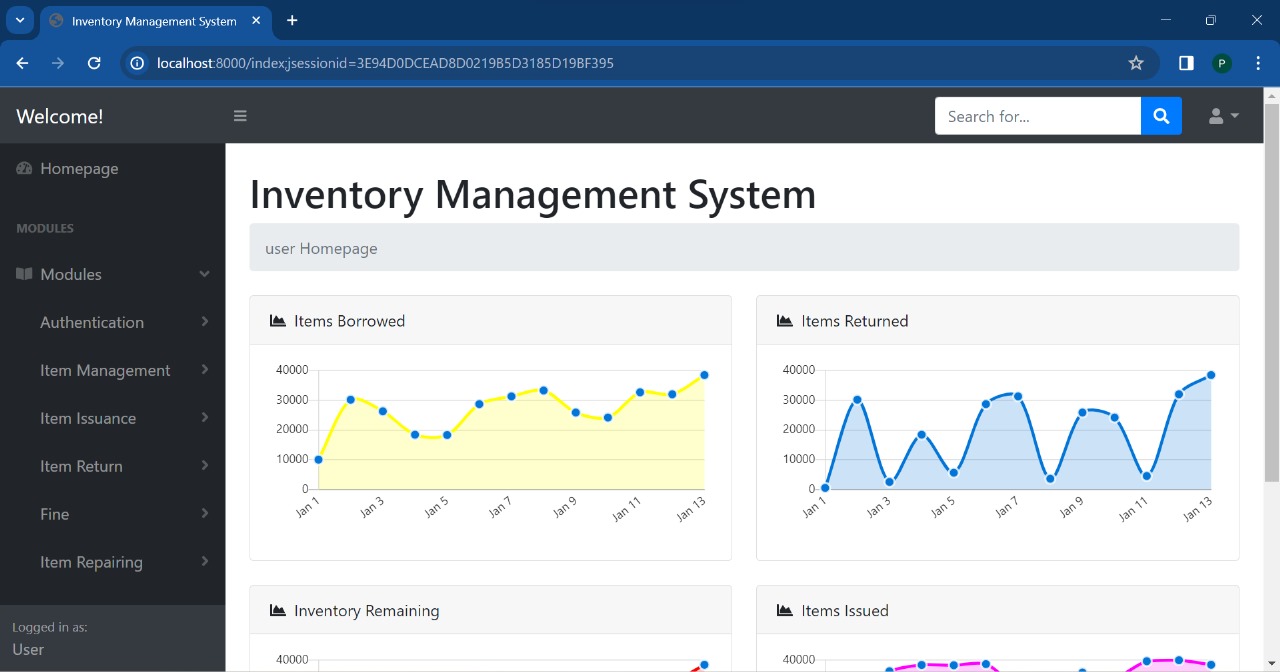
****

Fig 0.9 USER MAIN PAGE

**Modules in user**

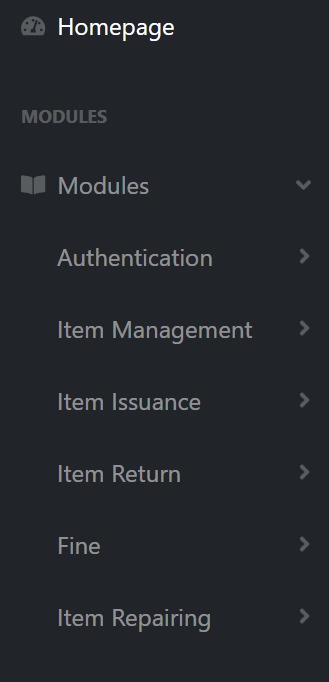
****

Fig 1.0 MODULES IN USER

**Add Items to the inventory**

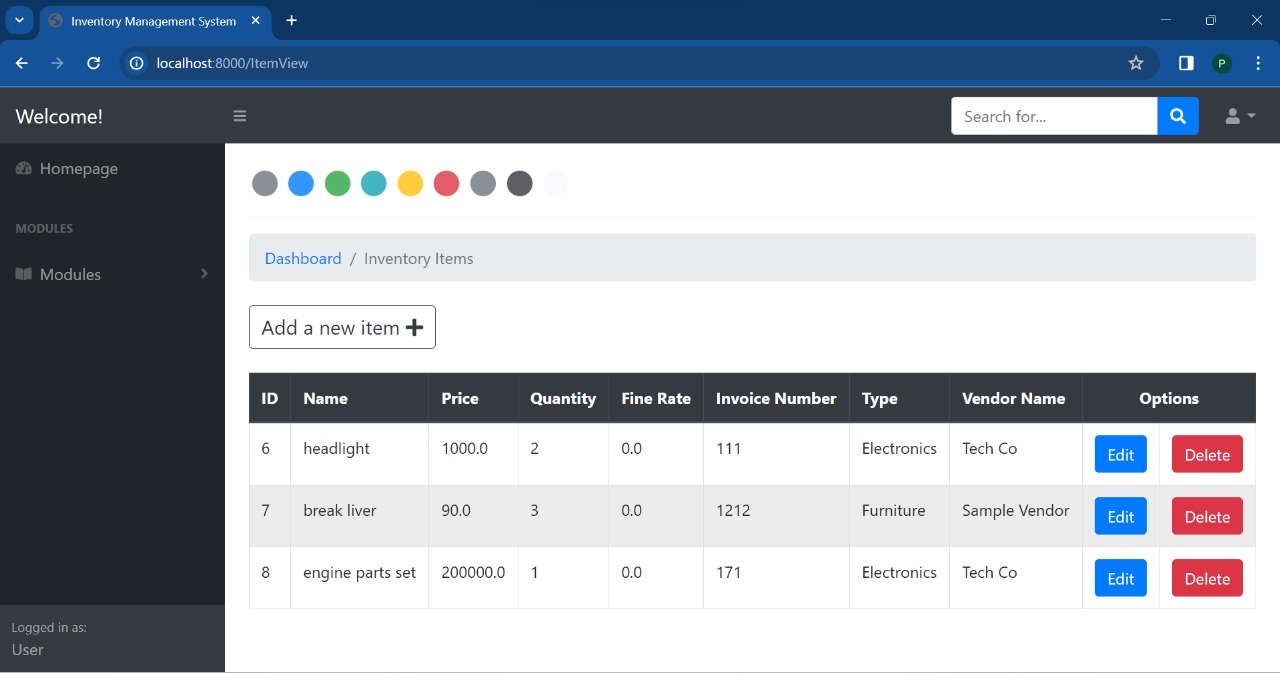
****

Fig 1.1 ADD ITEMS

**Item issuance page**

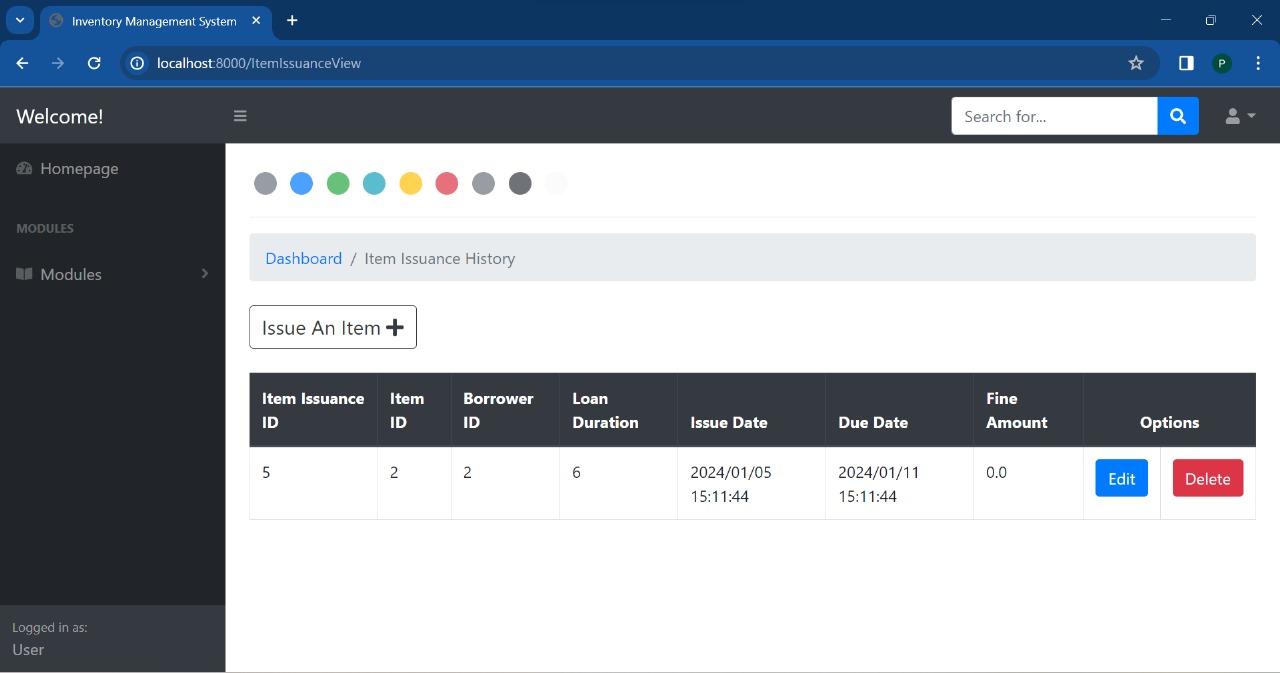
****

Fig 1.2 ITEM ISSUANCE PAGE

**Item return module**

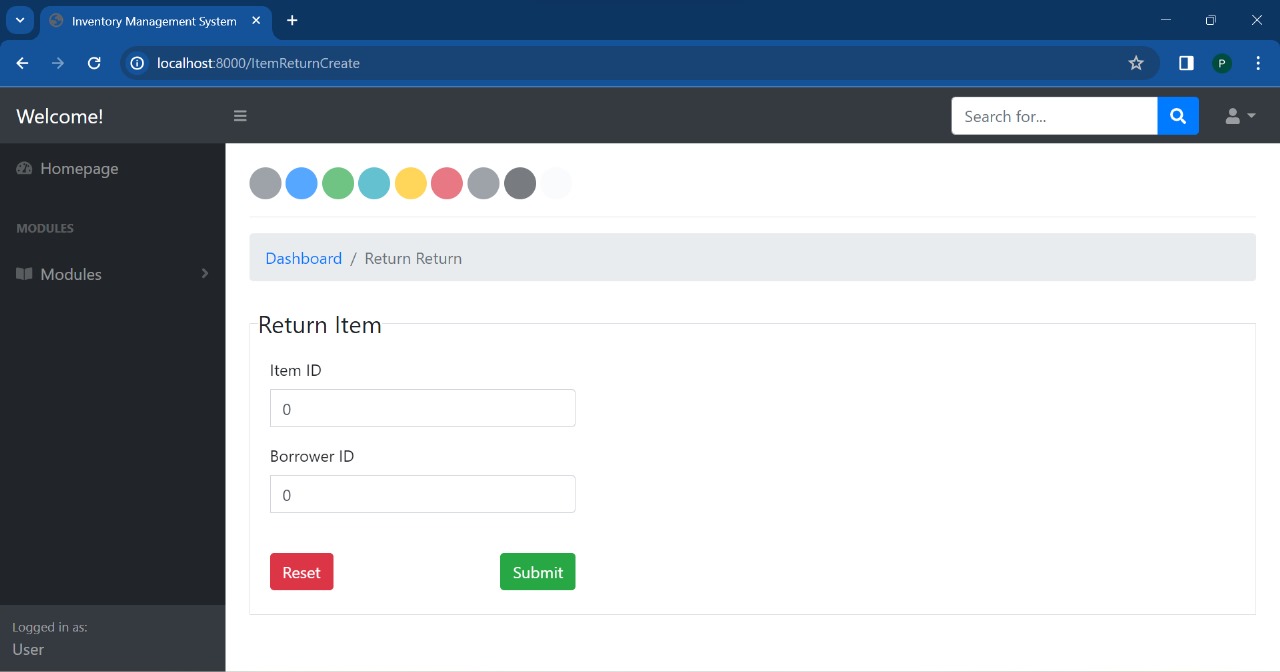
****

Fig 1.3 ITEM RETURN MODULE

**Fine calculation module**

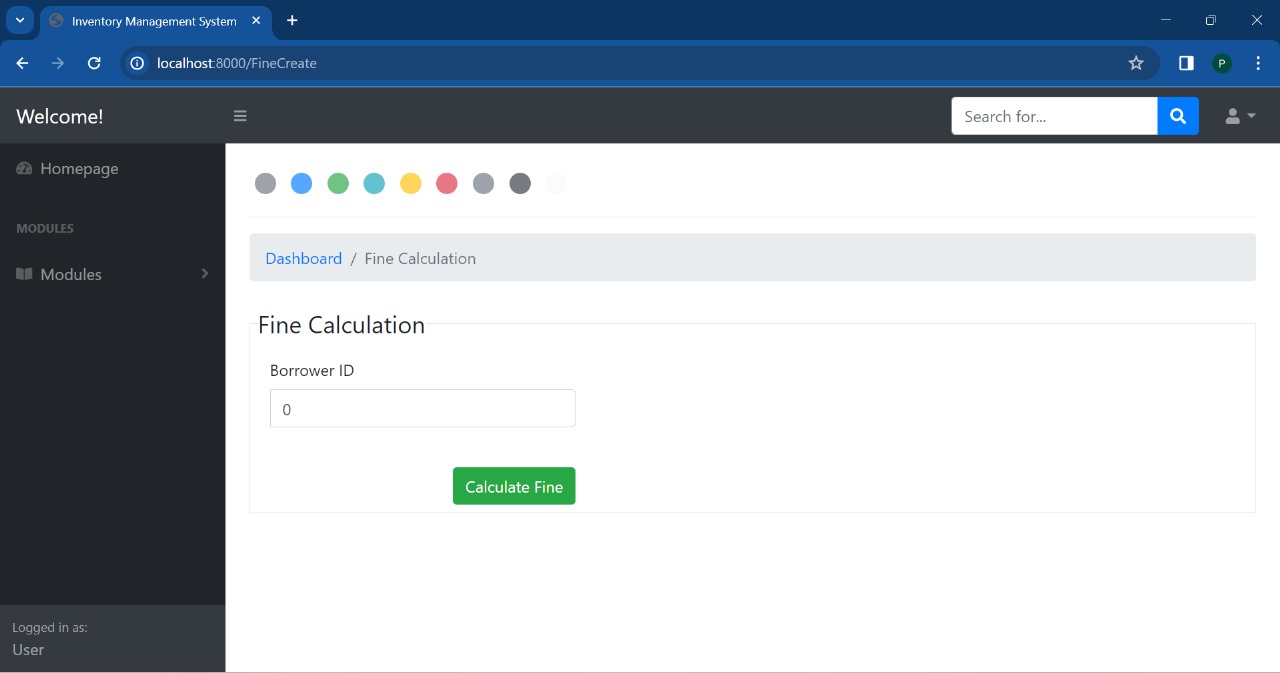
****

Fig 1.4 FINE CALCULATION

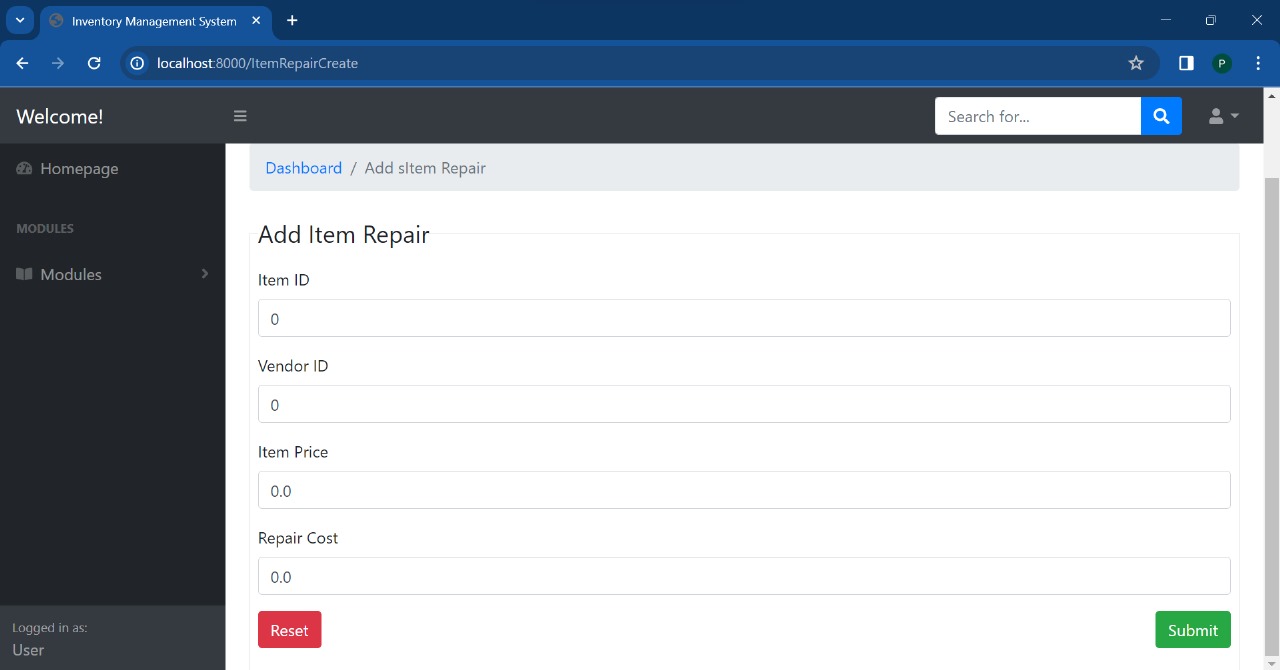
**Item repair module**

Fig 1.5 ITEM REPAIR

Testing presents an interesting anomaly for the software engineer. During earlier software engineering activities, the engineer attempts to build software from an abstract concept to a tangible product. Now comes testing. The engineer creates a series of test cases that are intended to “demolish” the software that has been built. In fact, testing is the one step in the software process that could be viewed (psychologically, at least) as destructive rather than constructive.

Software engineers are by their nature constructive people. Testing requires that the developer discard preconceived notions of the “correctness” of software just developed and overcome a conflict of interest that occurs when errors are uncovered.

If testing is conducted successfully (according to the objectives stated previously), it will uncover errors in the software. As a secondary benefit, testing demonstrates that software functions appear to be working according to specification, that behavioral and performance requirements appear to have been met. In addition, data collected as testing is conducted provide a good indication of software reliability and some indication of software quality as a whole. But testing cannot show the absence of errors and defects, it can show

Only that software errors and defects are present. It is important to keep this (rather gloomy) statement in mind as testing is being conducted.

**Testing principles**

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guide software testing:

All tests should be traceable to customer requirements

Tests should be planned long before testing begins

80 percent of all errors uncovered during testing will likely be traceable to 20 percent of all program components. The problem, of course, is to isolate these suspect components and to thoroughly test them.

Testing should being “in the small” and progress toward testing “in the large”.

Exhaustive testing is not possible

To be most effective an independent third party should conduct testing

A rich variety of test case design methods have evolved for software. These methods provide the developer with a systematic approach to testing. More important, methods provide a mechanism that can help to ensure the completeness of tests and provide the highest likelihood for uncovering errors in software.

**Any engineered product (and most other things) can be tested in one of two ways:**

Knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational

While at the same time searching for errors in each function; (2) knowing the internal

Working of a product, tests can be conducted to ensure that “all gears mesh,” that is, internal operations are performed according to specifications and all internal components have been adequately exercised. The first test approach is called black box testing and the second, white-box testing.

* **Testing performed were:**
  + UNIT TESTING
  + INTEGRATION TESTING
  + DATABASE TESTING
  + RECOVERY TESTING
  + FUNCTIONALITY TESTING
    - SMOKE TEST
    - SANITY TEST
  + COMPATIBILITY TESTING
  + LOAD TESTING
  + SYSTEM TESTING
  + PERFORMANCE TESTING
  + USER ACCEPTANCE TESTING

**White box testing**

Sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white-box testing methods, the software engineer can derive test cases that (1) guarantee that all independent paths within a module have been exercised at least once, (2) exercise all logical decisions on their true and false sides, (3) execute all loops at their boundaries and within their operational bounds, and (4) exercise internal data structures to ensure their validity.   
White-box testing of software is predicated on close examination of procedural detail. Providing test cases that exercise specific sets of conditions and/or loops tests logical paths through the software. The “status of the program” may be examined at various points to determine if the expected or asserted status corresponds to the actual status. Basis path testing is a white-box testing technique first proposed by Tom McCabe. The basis path method enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

In this system, the system was tested for the calculation matters were the data provided for giving the right output or not. If wrong data was provided then what it is throwing error or accepting.

**Black box testing**

Also called behavioral testing, focuses on the functional requirements of the software. That is, black box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white-box techniques. Rather, it is a complementary approach that is likely to uncover a different class of error than white-box methods. When computer software is considered, black box testing alludes to tests that are conducted at the software interface. Although they are designed to uncover errors, black-box tests are used to demonstrate that software functions are operational, that input is

Properly accepted and output is correctly produced and that the integrity of external information is maintained. A black-box test examines some fundamental aspect of a system with a little regard for the internal logical structure of the software. Black-box testing attempts to find errors in the following categories:

1. Incorrect or missing functions,
2. Interface errors,
3. Errors in data structures or external database access,
4. Behavior or performance errors, and
5. Initialization and termination errors. By applying back-box techniques, we derive a set of test cases that satisfy the following criteria:
   1. Test cases that reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing and
   2. Test cases that tell us something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand.

White-box testing should not, however, be dismissed as impractical. A limited number of important logical paths can be selected and exercised. Important data structures can be probed for validity. The attributes of both black and white box testing can be combined to provide an approach that validates the software interface and selectively ensures that the internal workings of the software are correct.

Black box testing for this system was done to check the internal testing i.e, the system is working properly in each case or no. What kind of errors are there in database design.

# **Testing Process**

The testing process can be shown as:

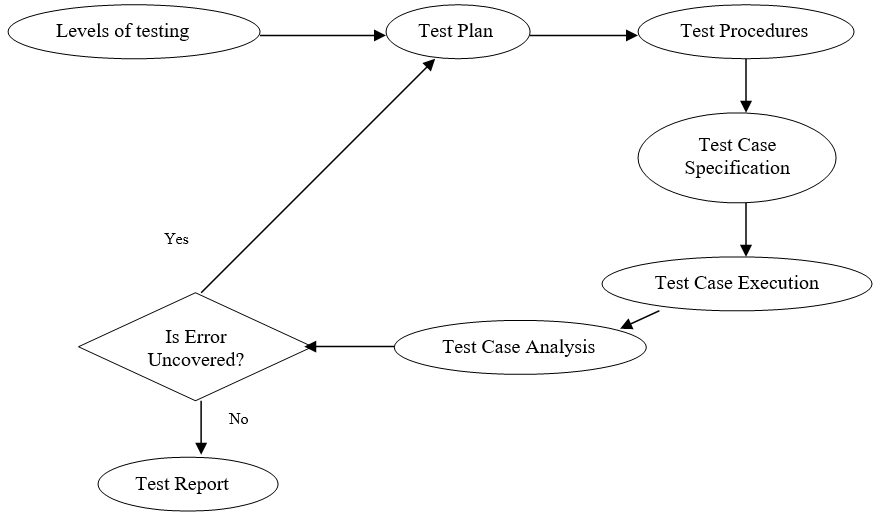


FIG 1.6 TESTING PROCESS

The Inventory Management System, boasting an array of comprehensive features, stands as a cornerstone solution for businesses seeking an efficient and seamless way to oversee their inventory, suppliers, orders, and user interactions. This sophisticated system goes beyond mere inventory tracking, offering a holistic approach to inventory management that empowers businesses to optimize their operational processes. Its user-friendly interface ensures that even users with varying levels of technological proficiency can navigate and leverage the system with ease

One of the system's key strengths lies in its robust security measures, implementing state-of-the-art authentication mechanisms to safeguard sensitive information. This ensures that only authorized personnel can access and manipulate critical data, providing peace of mind for businesses concerned with data integrity and confidentiality

In essence, the Inventory Management System is not merely a tool for maintaining inventory; it is a comprehensive solution that contributes to the overall enhancement of business operations. Its user-friendly design, stringent security measures, scalability, and data export capabilities collectively position it as an invaluable asset for businesses looking to streamline their inventory management processes and, consequently, boost overall efficiency

**REFERENCES**

* **Books**
* Inventory Management: Principles, Concepts, and Techniques" by Gopalan, M.N. - This

book provides a comprehensive overview of inventory management principles,

* Practical Machine Learning for Computer Vision" by Martin Görner, Valliappa

Lakshmanan, and Ryan Gillard

* **Academic Paper**
* Machine Learning for Inventory Optimization: A Critical Review by Kevin La and Rohan Dutta
* An Overview of AI Applications in Inventory Management" by A. S. A. Irandegani, et al. - This paper explores the various AI applications in inventory management and can serve as a valuable reference.

**MAIN CODE**

// Import necessary Spring Boot and other required packages

// Annotate the main class with @SpringBootApplication

@SpringBootApplication

public class ImsApplication {

// Define the main method

public static void main(String[] args) {

// Run the Spring Boot application

SpringApplication.run(ImsApplication.class, args);

}

}

**MAIN PAGE PROGRAM**

package com.example.IMS.controller;

import javax.validation.Valid;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import com.example.IMS.model.User;

import com.example.IMS.service.UserService;

@Controller

public class HomeController {

private final UserService userService;

@Autowired

public HomeController(UserService userService) {

this.userService = userService;

}

@GetMapping("/")

public String login(){

return "login";

}

@GetMapping("/index2")

public String Index2(){

return "index2";

}

@GetMapping("/index")

public String Index(){

return "index";

}

@GetMapping("/adduser")

public String showAddUserForm(Model model) {

model.addAttribute("user", new User());

return "admin/adduser";

}

@PostMapping("/adduser")

public String addUser(@ModelAttribute("user") @Valid User user, BindingResult bindingResult) {

if (bindingResult.hasErrors()) {

return "admin/adduser";

}

// Save the user to the database (Assuming you have a UserService)

userService.saveUser(user);

// Redirect to the user list page or another appropriate page

return "redirect:/adduser";

}

}

**ITEM CONTROLLER**

// Import necessary Spring and project-specific packages

// Annotate the class as a Controller

@Controller

public class ItemIssuanceController {

// Inject the necessary services and components

@Autowired

private ItemIssuanceService itemIssuanceService;

@Autowired

private ItemService itemService;

@Autowired

private BorrowerService borrowerService;

@Autowired

private ItemIssuanceConvertor itemIssuanceConvertor;

// Handle GET requests to "/ItemIssuanceView"

@GetMapping("/ItemIssuanceView")

public String viewIssuedItems(Model model) {

// Retrieve and add the ItemIssuanceDtoList to the model for the view

model.addAttribute("ItemIssuanceDtoList", itemIssuanceConvertor.modelToDto(itemIssuanceService.getAllIssuedItems()));

return "/Item Issuance/View";

}

// Handle GET requests to "/ItemIssuanceCreate"

@GetMapping("/ItemIssuanceCreate")

public String createIssuance(Model model) {

// Initialize a new ItemIssuanceDto and add it to the model

ItemIssuanceDto itemIssuanceDto = new ItemIssuanceDto();

model.addAttribute("itemIssuanceDto", itemIssuanceDto);

return "/Item Issuance/Create";

}

// Handle POST requests to "/ItemIssuanceCreate"

@PostMapping("/ItemIssuanceCreate")

public String createIssuance(@Valid @ModelAttribute("itemIssuanceDto") ItemIssuanceDto itemIssuanceDto, BindingResult result) {

// Implementation for creating item issuance

// ...

return "redirect:/ItemIssuanceView";

}

// Handle GET requests to "/ItemIssuanceEdit/{id}"

@GetMapping("/ItemIssuanceEdit/{id}")

public String editIssuance(@PathVariable(value = "id") long id, Model model) {

// Implementation for editing item issuance

// ...

return "/Item Issuance/Edit";

}

// Handle GET requests to "/ItemIssuanceDelete/{id}"

@GetMapping("/ItemIssuanceDelete/{id}")

public String deleteIssuance(@PathVariable(value = "id") long id, Model model) {

// Implementation for displaying delete confirmation page

// ...

return "/Item Issuance/Delete";

}

// Handle POST requests to "/ItemIssuanceDelete/{id}"

@PostMapping("/ItemIssuanceDelete/{id}")

public String deleteIssuance(@PathVariable(value = "id") long id, @ModelAttribute("itemIssuanceDto") ItemIssuanceDto itemIssuanceDto) {

// Implementation for deleting item issuance

// ...

return "redirect:/ItemIssuanceView";

}

}

**ITEM REPAIR CONTROLLER**

// Import necessary Spring and project-specific packages

// Annotate the class as a Controller

@Controller

public class ItemRepairController {

// Inject the necessary services and components

@Autowired

private ItemRepairService itemRepairService;

@Autowired

private VendorService vendorService;

@Autowired

private ItemService itemService;

@Autowired

private ItemRepairConvertor itemRepairConvertor;

// Handle GET requests to "/ItemRepairView"

@GetMapping("/ItemRepairView")

public String viewRepairItems(Model model) {

// Retrieve and add the ItemRepairDtoList to the model for the view

model.addAttribute("ItemRepairDtoList", itemRepairConvertor.modelToDto(itemRepairService.getAllRepairItems()));

return "/Item Repair/View";

}

// Handle GET requests to "/ItemRepairCreate"

@GetMapping("/ItemRepairCreate")

public String createRepair(Model model) {

// Initialize a new ItemRepairDto and add it to the model

ItemRepairDto itemRepairDto = new ItemRepairDto();

model.addAttribute("itemRepairDto", itemRepairDto);

return "/Item Repair/Create";

}

// Handle POST requests to "/ItemRepairCreate"

@PostMapping("/ItemRepairCreate")

public String createRepair(@Valid @ModelAttribute("itemRepairDto") ItemRepairDto itemRepairDto, BindingResult result) {

// Validate vendor and item IDs

String err = vendorService.validateVendorId(itemRepairDto.getVendorId());

if (!err.isEmpty()) {

ObjectError error = new ObjectError("globalError", err);

result.addError(error);

}

err = itemService.validateItemId(itemRepairDto.getItemId());

if (!err.isEmpty()) {

ObjectError error = new ObjectError("globalError", err);

result.addError(error);

}

// Check for validation errors

if (result.hasErrors()) {

return "/Item Repair/Create";

}

// Save the ItemRepair to the database

itemRepairService.saveItemRepair(itemRepairConvertor.DtoToModel(itemRepairDto));

return "redirect:/ItemRepairView";

}

// Handle GET requests to "/ItemRepairEdit/{id}"

@GetMapping("/ItemRepairEdit/{id}")

public String editRepair(@PathVariable(value = "id") long id, Model model) {

// Retrieve the existing ItemRepair from the database

ItemRepair itemRepair = itemRepairService.findItemRepairById(id);

// Add the ItemRepairDto to the model for the view

model.addAttribute("itemRepairDto", itemRepairConvertor.modelToDto(itemRepair));

return "/Item Repair/Edit";

}

// Handle GET requests to "/ItemRepairDelete/{id}"

@GetMapping("/ItemRepairDelete/{id}")

public String deleteRepair(@PathVariable(value = "id") long id, Model model) {

// Retrieve the existing ItemRepair from the database

ItemRepair itemRepair = itemRepairService.findItemRepairById(id);

// Add the ItemRepairDto to the model for the view

model.addAttribute("itemRepairDto", itemRepairConvertor.modelToDto(itemRepair));

return "/Item Repair/Delete";

}

// Handle POST requests to "/ItemRepairDelete/{id}"

@PostMapping("/ItemRepairDelete/{id}")

public String deleteRepair(@PathVariable(value = "id") long id, @ModelAttribute("itemRepairDto") ItemRepairDto itemRepairDto) {

// Delete the ItemRepair by ID

itemRepairService.deleteItemRepairById(id);

// Redirect to the view page

return "redirect:/ItemRepairView";

}

}

**MY SQL WORKBENCH**

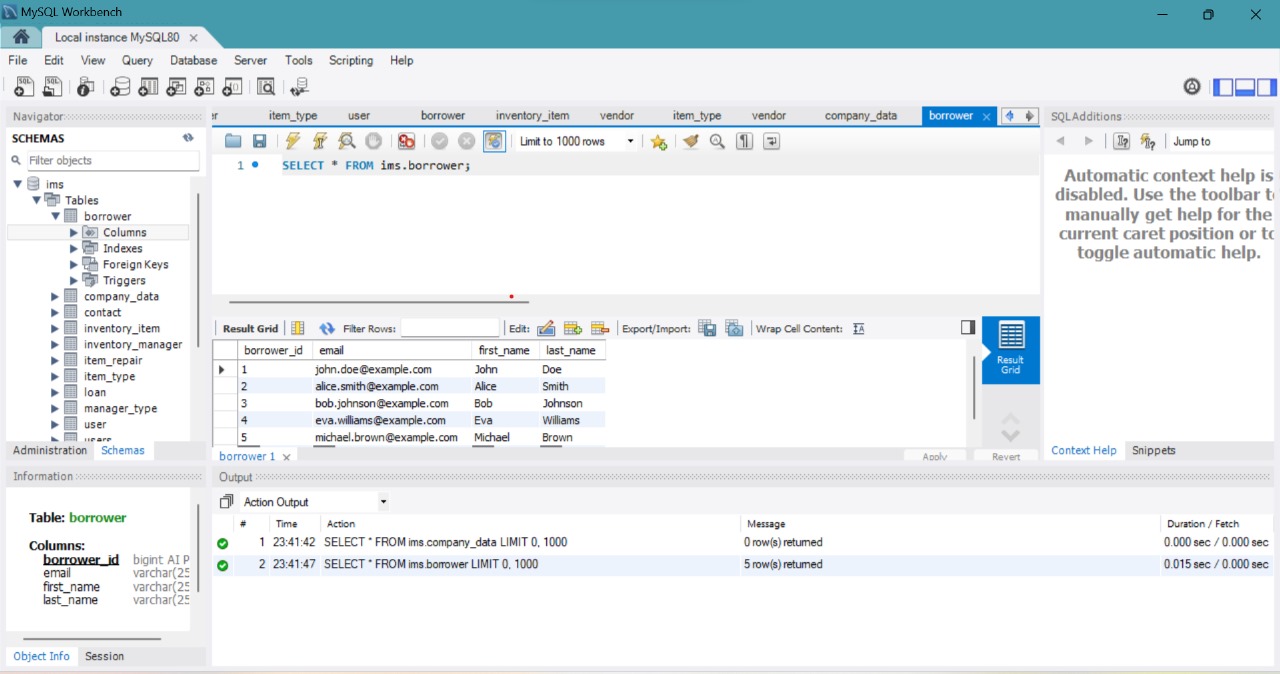
****

Fig 1.7 SQL WORKBENCH

**JAVA ECLIPSE**

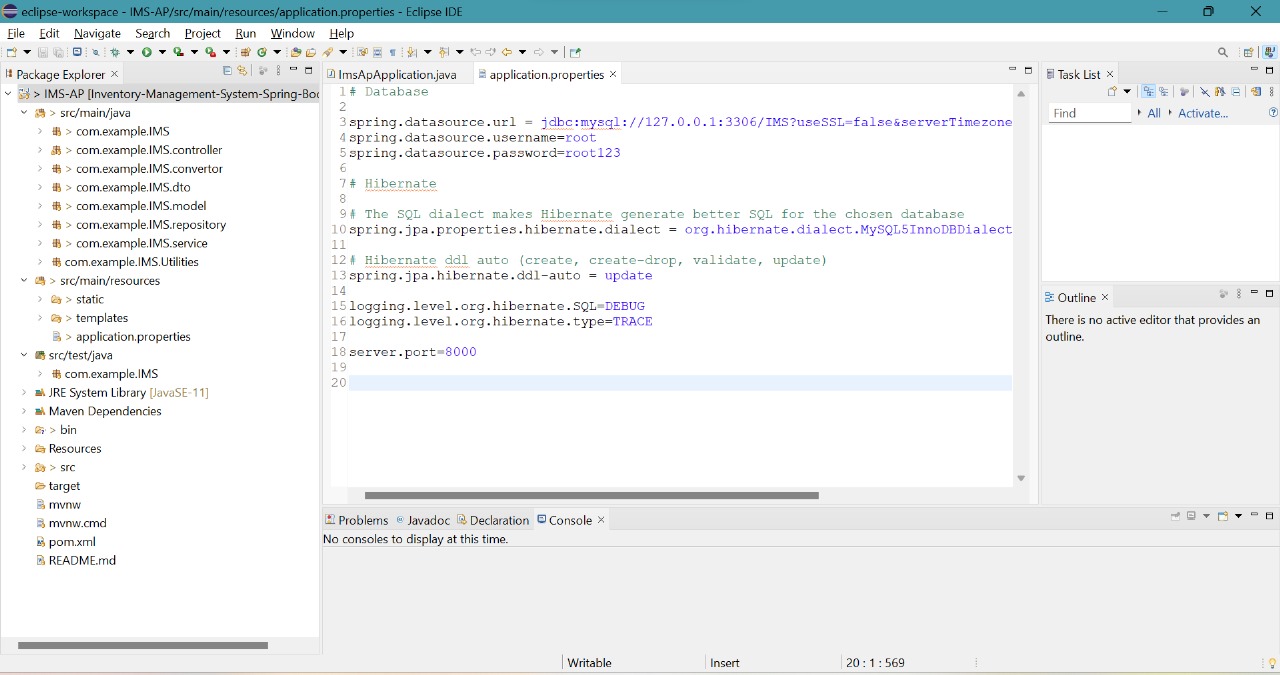


FIG 1.8 ECLIPSE WORKSPACE

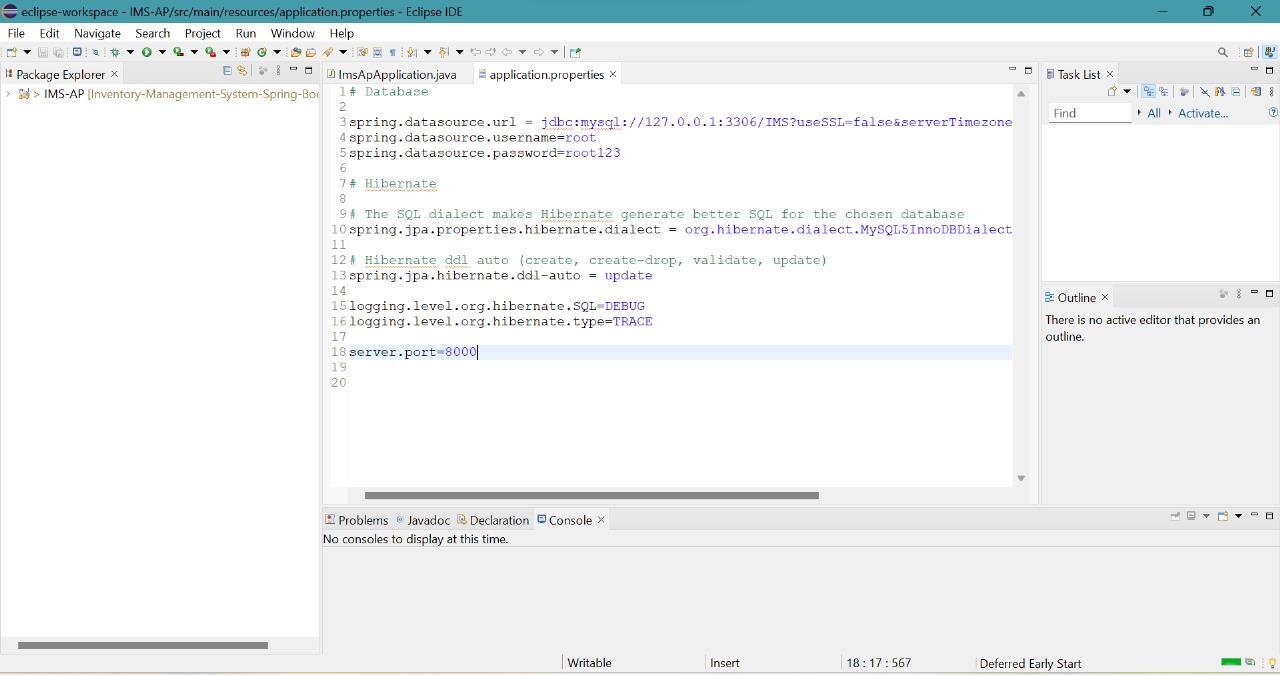


FIG 1.9 APPLICATION PROGRAM IN ECLIPSE

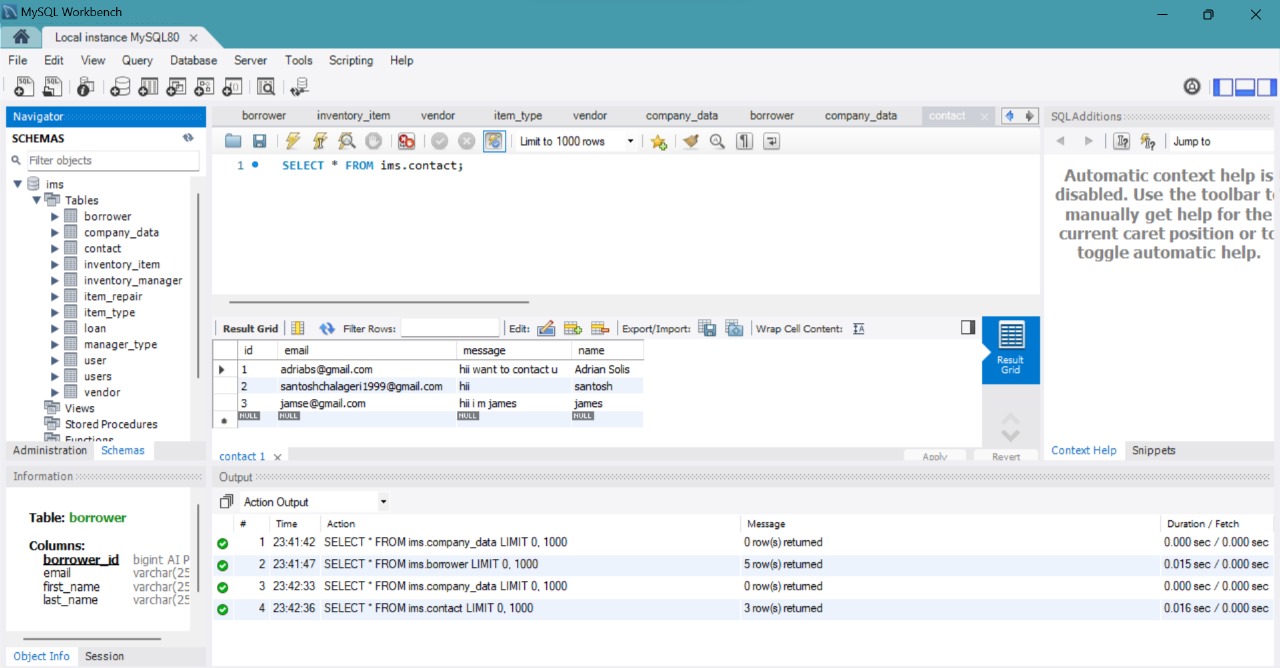


FIG 2.0 TABEL RECORDS IN MY SQL WORK BENCH