MEGHNAD SAHA INSTITUTE OF TECHNOLOGY

Techno Complex,. Madurdaha,Beside NRI Complex, Post-Uchhepota, Kolkata 700 150

MAJOR PROJECT REPORT

INVOICE MANAGEMENT SYSTEM

MAKAUT EVEN SEMESTER 2022 - 23

[MASTER OF COMPUTER APPLICATION]

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

(Formerly known as WEST BENGAL UNIVERSITY OF TECHNOLOGY)

UNDER THE SUPERVISION OF

MR. SOUMYA CHAKRAVARTY

Submitted By : SOURADIP KUNDU [14271021024]

CERTIFICATE

This is to certify that the project entitled "**INVOICE MANAGEMENT SYSTEM**" has been prepared according to the regulation of degree of Master of Computer Application (MCA)under the University of "**Maulana Abul Kalam Azad University of Technology**". The project being submitted by-

(Student's Signature)

Students of Master of Computer Application (MCA), 2nd year 2nd semester of **MEGHNAD SAHA INSTITUTE OF TECHNOLOGY** (affiliated to Maulana Abul Kalam Azad University ofTechnology) has fulfilled the requirement for submission of this.

The whole procedure has been carried out under my supervision and guidance. I have gone through this project and have seen that it is fulfilling the requirement of Major Project under MAKAUT, WB.

Dated: _____

(INTERNAL PROJECT GUIDE)

(EXTERNAL PROJECT GUIDE)

(HEAD OF THE DEPARTMENT)
Meghnad Saha Institute of technology

(EXAMINER)

ACKNOWLEDGEMENT

We would like to acknowledge our sincere gratitude to the mentors of **Meghnad saha Institute of technology** Without their guidance and constantsupervision this project would not have been possible. We are very much thankful for providing us necessary information about this project as well as for the support in completion of the project.

Our thanks and appreciations also go to those people who have willingly helped us out with their abilities in developing this project.

SOURADIP KUNDU	**Roll No.-** 14271021024	**MSIT**	

ABSTRACT

The project has been done under the guidance of **MR. SOUMYA CHAKRAVARTY** from **Meghnad Saha Institute of technology**. The project is based on **Java Spring Boot and MySQL for Invoice Management System**. The scope of this project is to reduce the manual operation required to maintain all the records of booking information and also generates the various reports for analysis. Main concept of the project is to enter transaction reports and to maintain customer records. Hence this system can be used in any colleges to maintain their records easily.

Signatures of Students

1.

Date: **Signature of Faculty**

Date :

ACKNOWLEDGEMENT

I take this opportunity to express my deep gratitude and sincerest thank to my project mentor **MR. SOUMYA CHAKRAVARTY** for giving most valuable suggestion, helpful guidance and encouragement in the execution of this project work. who gave me a golden opportunity to do these reports on **"Invoice Management System",** who also helped in completing my report. I came to know about so many new things and I am really thankful to them. Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

I would like to give a special mention to my colleagues. Last but not the least, I am grateful to all the faculty members of **Meghnad Saha institute of technology** for their support.

TABLE OF CONTENTS

SL No.	Name of the topic	Page No.
1	Introduction	7
2	Object & Scope	8 - 9
3	Software Requirement Specifications	10
4	Hardware Requirement Specifications	10
5	System Analysis	11
5.1	Features	12
5.2	Brief Overview of the Technology	12
5.3	Work flow	13 – 15
5.4	Software tool used	16 - 17
5.5	Concept & problem analysis	18 – 19
6	System design	20
6.1	System Design	21
6.2	Flow Chart	22
6.3	Data Flow Diagram	23
6.4	Use case Diagram	24
6.5	Entity-Relationship Diagram	25
6.6	ER Diagram of the database	26 - 27
6.7	ER Diagram of the Features	28 - 31
6.8	Database Design	32
6.9	User Interface	33
6.10	Feasibility study	34
7	Implementation & Testing	35
7.1	Test cases	36
7.2	Testing steps	37
7.3	Performance & usability Test ingredients	38
7.4	Compatibility & Security Testing	39
7.5	Integration Testing	40
7.6	White Box Testing	41
7.7	Black box Testing	41
7.8	System testing	41
7.9	Output Testing	42
7.10	Goal of Testing	42
7.11	Acceptance Testing	43
7.12	validation	44
7.13	Testing Report	45
8	Screenshot	46 – 63
9	Coding	64 – 133
10	System Security Measures	134
10.1	Database security measures	135
10.2	System security measures	135
11	Limitation	136
12	Conclusion	137
13	Future Scope	138
14	References	139

INTRODUCTION

Invoice Management System is a website created specifically for accessing Invoice of customers. This method the process businesses execute to invoice their clients for products and services they purchase, as well as to track invoices and make invoice payments to suppliers and vendors. In simple terms, invoice management involves dealing with both accounts receivable and accounts payable.

An invoice management system is a software application or a set of tools designed to streamline and automate the process of creating, sending, tracking, and managing invoices within an organization. It provides businesses with an efficient and organized way to handle their invoicing processes, reducing manual efforts, minimizing errors, and improving overall productivity.

Traditionally, invoicing involves manual tasks such as creating invoices using word processors or spreadsheets, printing and mailing them, and manually tracking payments. These processes are not only time-consuming but also prone to errors and delays, leading to inefficiencies and potential financial losses.

An invoice management system aims to simplify and optimize these processes by providing a centralized platform where businesses can generate professional-looking invoices, automate recurring billing, track payment statuses, and manage customer information. It typically integrates with accounting software and other relevant systems to ensure accurate financial records and seamless data synchronization

This Invoice Management System And managing System is in HTML, JavaScript, and CSS. Talking about the features of this system, it contains the server section and the user (customer) section. All the editing, updating, managing details, order items from the admin section while customers can only go through the site and give orders if they want. The design of this system is simple so that the user won't get any difficulties while working on it.

OBJECTIVE

To develop an invoice management system using Java that allows users to create, store, and manage invoices efficiently.

Functional Requirements:

- **User Authentication:** The system should provide secure user authentication to ensure only authorized users can access and manage invoices.
- **Invoice Creation:** Users should be able to create new invoices by entering necessary details such as customer information, product details, quantity, price, etc.
- **Invoice Storage:** The system should provide a database to store and manage invoices securely.
- **Invoice Listing:** Users should be able to view a list of all invoices stored in the system, including essential details such as invoice number, customer name, and total amount.
- **Invoice Editing:** Users should have the ability to edit and update existing invoices if necessary, such as modifying product details, quantity, or price.
- **Invoice Deletion:** Users should be able to delete invoices from the system when they are no longer needed.
- **Invoice Searching:** The system should allow users to search for invoices based on various criteria, such as invoice number, customer name, or date.
- **Invoice Filtering:** Users should be able to filter invoices based on specific parameters, such as date range or invoice status (paid, unpaid, etc.).
- **Invoice Summary:** The system should generate summary reports, including total revenue, outstanding invoices, and paid invoices.
- **Invoice Status Management:** Users should be able to update the status of invoices, marking them as paid or unpaid.
- **User Roles and Permissions:** The system should support different user roles (e.g., admin, manager, regular user) with different levels of access and permissions.
- **User Interface:** The system should have a user-friendly interface to facilitate easy navigation and interaction with the invoice management functions.

Non-functional Requirements:

- **Security:** The system should implement proper security measures to protect sensitive invoice data and user information.
- **Performance:** The system should be efficient and responsive, allowing quick retrieval and manipulation of invoice data, even with a large number of records.
- **Reliability:** The system should be reliable, ensuring that invoice data is stored accurately and consistently, and minimizing the risk of data loss or corruption.
- **Scalability:** The system should be designed to handle increasing amounts of invoice data without significant degradation in performance.
- **Maintainability:** The system should be modular and well-structured, making it easy to maintain and extend with additional features in the future.
- **Accessibility:** The user interface should be accessible to users with disabilities, following appropriate accessibility guidelines and standards.
- **Compatibility:** The system should be compatible with different operating systems and browsers, ensuring wide accessibility across various platforms.
- These are some of the key objectives and requirements for developing an invoice management system using Java. Depending on your specific needs, you may need to further customize and refine these requirements.

SCOPE

The system allows users to create and generate invoices for products or services provided to customers. It should include options to input essential information such as customer details, item descriptions, quantities, prices, and applicable taxes. The system should provide a centralized dashboard or interface where users can view and track the status of invoices. This includes monitoring the payment status, due dates, and any outstanding or overdue invoices.Integration with payment gateways or accounting systems allows for streamlined payment processing. Users can record payments received, apply them to the corresponding invoices, and generate payment receipts if necessary's important to note that the scope of an invoice management system may vary depending on the specific requirements and goals of the organization implementing it. Customization and additional features can be incorporated based on the unique needs of the business.

Software Requirement Specification (SRS)

PROCESSOR	INTEL i3 7th Gen processor
OPERATING SYSTEM	MacOS, WINDOWS, Linux, Android
MEMORY	4GB RAM OR MORE
HARD DISK SPACE	MINIMUM 1GB FOR DATABASE USAGE FOR FUTURE USE.
DATABASE	MYSQL

SOFTWARE REQUIREMENT

Operating System	**:**	**Windows 8/10/11**
Front-End Tool	**:**	**HTML, CSS, Bootstrap, JavaScript.**
Back-End Tool	**:**	**MySQL, JAVA**
Programming Languages	**:**	**JAVA, JSP, Servlet, JDBC.**

HARDWARE REQUIREMENT

Processor	**:**	**Intel Pentium to any updated**
Processor Speed	**:**	**250MHz to 667 MHz**
RAM	**:**	**2GB**
Hard Disk	**:**	**80 GB**

SOFTWARE TOOL USED

Software Tool	**:**	**Ellipse IDE, VS CODE**
Java Programming	**:**	**JDK 17**
Server	**:**	**Apache Tomcat Server(9.0.75)**
Server Software	**:**	**Xampp**
Database	**:**	**MySQL**

System Analysis

5.1 FEATURES

The features of this system are the following:

- User Registration and Authentication
- Invoice Creation
- Invoice Storage
- Invoice Listing
- Invoice Editing
- Invoice Deletion
- Invoice Searching
- Invoice Summary and Reports.
- Invoice Status Management
- Invoice Notifications
- Data Import/Export
- Reminders and Due Dates

5.2 BRIEF OVERVIEW OF THE TECHNOLOGY

Front End :

1. **HTML:** HTML is used to create and save web documents.
2. **CSS :** (Cascading Style Sheets) Create attractive Layout
3. **Bootstrap :** responsive design mobile friendly site
4. **JavaScript:** it is a programming language, commonly used with web browsers.

Back end :

1. **JAVA :** Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.
2. **MySQL:** Creation of tables and collections
3. **JDBC:** Java database connectivity (JDBC) that allows Java programs to access database management systems.
4. **JSP :** JSP (Java Server Pages) is server side technology to create dynamic java web application.
5. **Servlet:** A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.

5.3 WORK FLOW

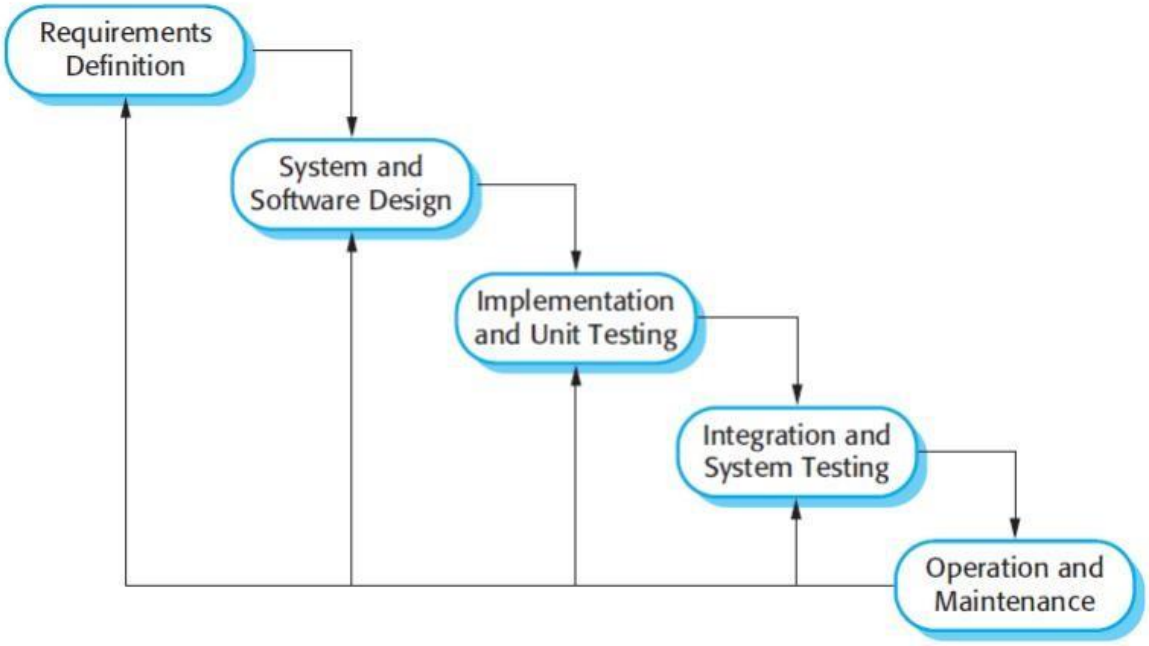
This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

Waterfall Model design Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.



The sequential phases in Waterfall model are:

- Requirement Gathering and analysis: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design :** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation :** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- Integration and Testing: All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system :** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance :** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

5.4 SOFTWARE TOOLS USED

- The whole project is divided in two parts the front end and the back end.

FRONT END

- **Hypertext Markup Language** (**HTML**) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <**img** /> and <**input** /> directly introduce content into the page. Other tags such as <**p**> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

- **Bootstrap** is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. Bootstrap is the sixth-most-starred project on GitHub, with more than 135,000 stars, behind freeCodeCamp (almost 307,000 stars) and marginally behind Vue.js framework. According to Alexa Rank, Bootstrap is in the top-2000 in US while vuejs.org is in top-7000 in US.

- **JavaScript (**/ˈdʒɑːvəˌskrɪpt/), often abbreviated as **JS**, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and functions. Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). However, the language itself does not include any input/output (I/O), such as networking, storage, or graphics facilities, as the host environment (usually a web browser) provides those APIs.JavaScript engines were originally used only in web browsers, but they are now embedded in some servers, usually via Node.js. They are also embedded in a variety of applications created with frameworks such as Electron and Cordova. Although there are similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

- **Cascading Style Sheets** (**CSS**) is a style sheet language text/css describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a device. The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable. The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type)is registered for use with CSS by RFC 2318 (March 1998). The W3Coperates a free CSS validation service for CSS documents. In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

BACK END

The back end is designed using MySQL which is used to design the database.

- **MySQL** is an open-source relational database management system (RDBMS).Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Language. MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for *Linux, Apache, MySQL, Perl/PHP/Python*. MySQL is used by many database-driven web applications, including Drupal, Joomla, phub, and WordPress. MySQL is also used by many popular websites, including Facebook, Flickr, MediaWiki,Twitter and YouTube.
- **JAVA** is popular high-level, class-based object-oriented programming language originally developed by Sun Microsystems and released in 1995. Currently Java is owned by Oracle and more than 3 billion devices run Java. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java is used to develop numerous types of software applications like Mobile apps, Web apps, Desktop apps, Games and much more. This **Java** Tutorial has been prepared by well experienced Java Programmers for the Software Engineers to help them understand the basic to advanced concepts of Java Programming Language. After completing this tutorial, you will find yourself at a moderate level of expertise in Java, from where you can take yourself to the next levels.

- **JSP** technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc. JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy. JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic. If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application. In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

- **Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below. There are many interfaces and classes in the Servlet API such as Servlet, Generic Servlet, HTTP Servlet, Servlet Request, Servlet Response, etc. A web application is an application accessible from the web. A web application is composed of web components like Servlet, JSP, Filter, etc. and other elements such as HTML, CSS, and JavaScript. The web components typically execute in Web Server and respond to the HTTP request. Servlets provide a component-based, platform-independent method for building Webbased applications, without the performance limitations of CGI programs. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. This tutorial will teach you how to use Java Servlets to develop your web-based applications in simple and easy steps. sing Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

5.5 CONCEPTS & PROBLEM ANALYSIS

COCOMO model :

COCOMO Model–The Constructive Cost Model is a software cost estimation model based on an algorithm. It was developed by Barry Boehm first published in 1981 Barry W. Boehm's Book Software engineering economics as a model for estimating effort, cost, and schedule for software projects.

COCOMO applies to three classes of software projects:

- Organic projects- "small" teams with "good" experience working with "less than rigid" requirements.
- Semi-detached projects- "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements.
- Embedded projects- developed with a set of "tight" constraints (hardware, software, operational,..)

BASIC COCOMO equations-

- EFFORT(E)= $a1(KLOC)^{b1}$ person month
- Time for Development (Tdev)= $c1(Effort)^{d1}$ Month

Here, KLOC-Thousands or kilos of lines of code

Software Project	a_1	b_1	c_1	d_1
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

CALCULATIONS: here the values depend on the types of software products or projects. We have just told that our software project is of organic type. So according to that the values of

a1=2.4

b1=1.05

c1=2.5

d1=0.38

In our project the estimated line of code**(LOC)** is **4064**. So the estimated size of the project is **4.64 KLOC**. Hence considering it to be organic software the following is calculated

We know that, **Effort**= $a1(KLOC)^b 1$person month

=2.4*$(4.64)^{1.05}$PM

=**12.024 PM**

Finally, **Time for Development** (Tdev)= $c1(Effort)^d 1$ Months

=2.5*$(12.024)^{0.38}$ Months

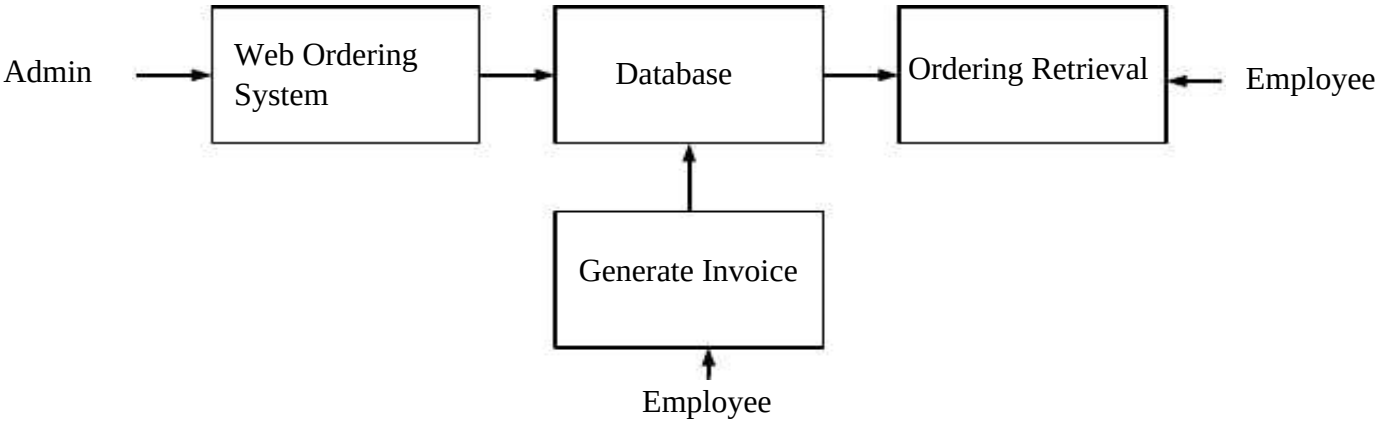
=3.644 Months

≈ 3 Months

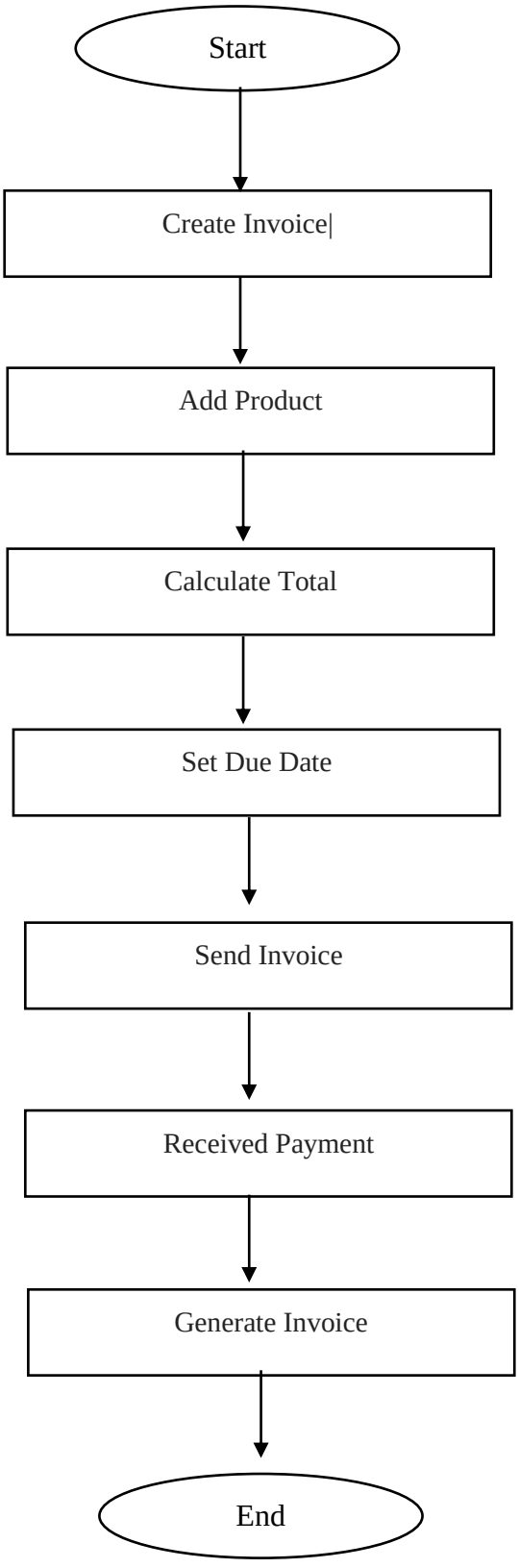
System Design

6.1 System Design:

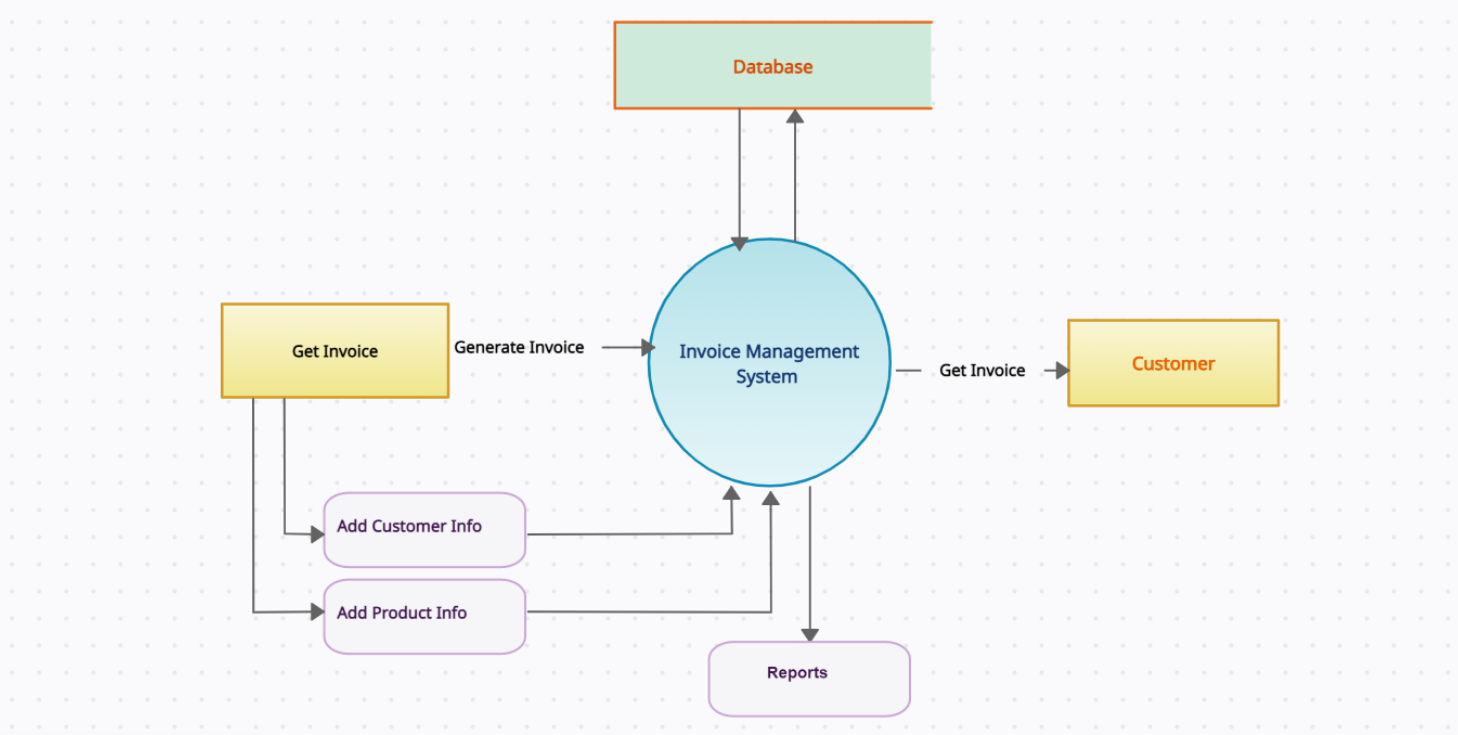
The structure of the system can be divided into three main logical components. The first component must provide some form of menu management, allowing the restaurant to control what can be ordered by customers. The second component is the web ordering system and provides the functionality for customers to place their order and supply all necessary details. The third and final logical component is the order retrieval system. Used by the restaurant to keep track of all orders which have been placed, this component takes care of retrieving and displaying order information, as well as updating orders which have already been processed.



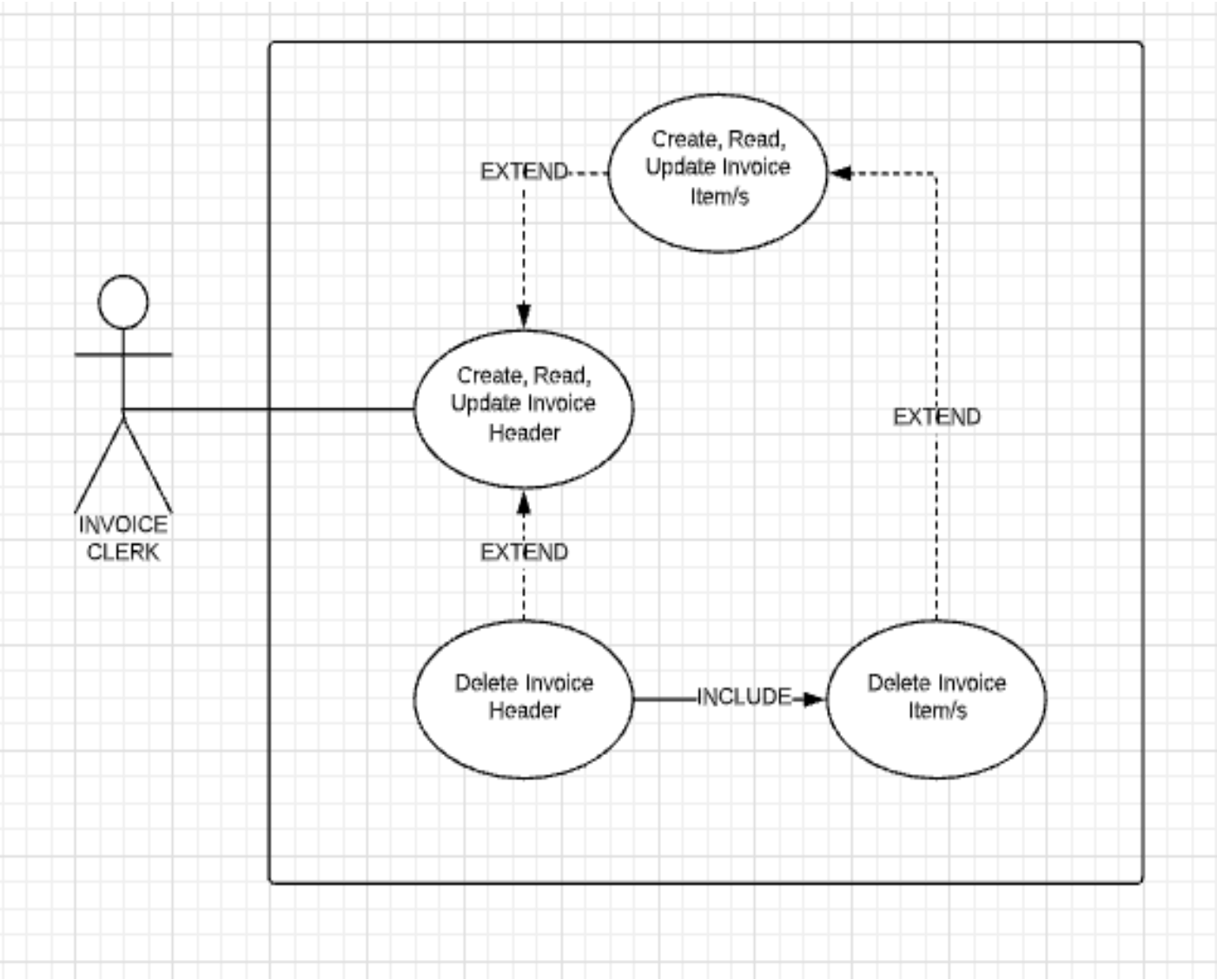
6.2 Flow Chart Diagram :



6.3 Data Flow Diagram (DFD) :

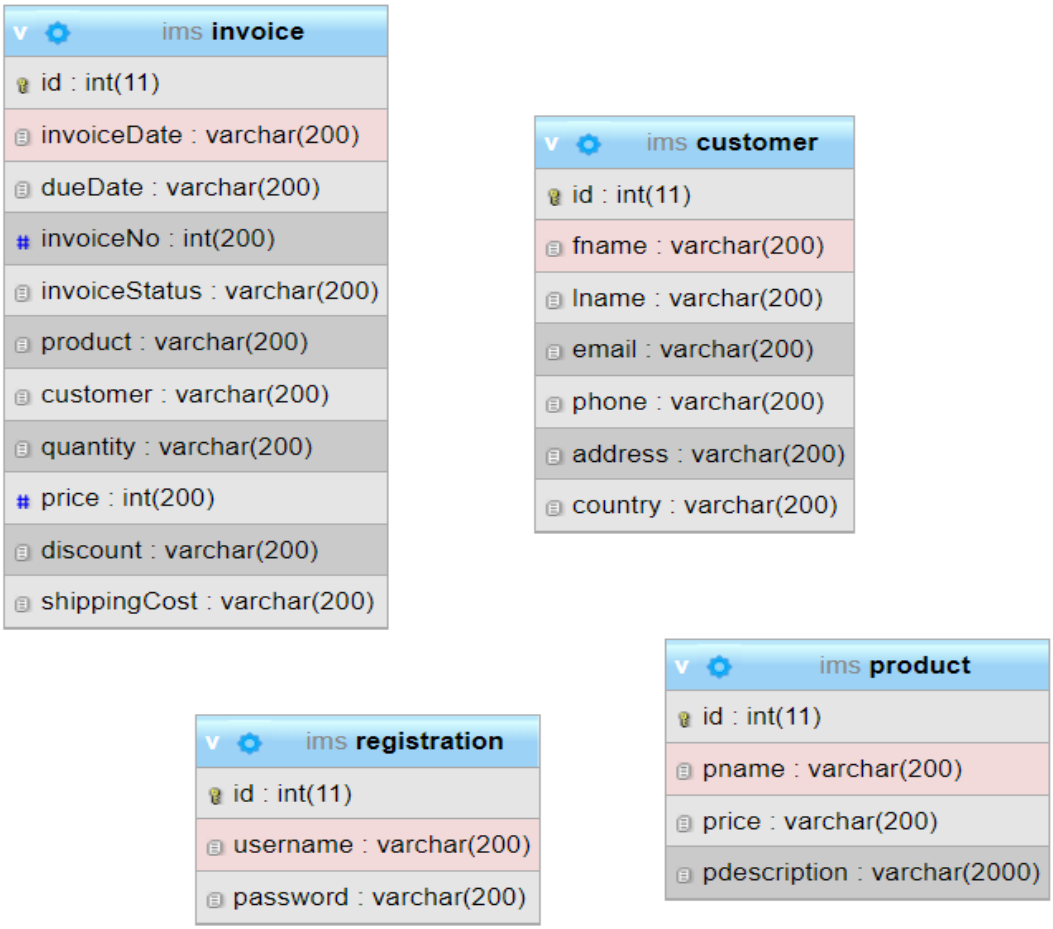


6.4 Use Case Diagram :

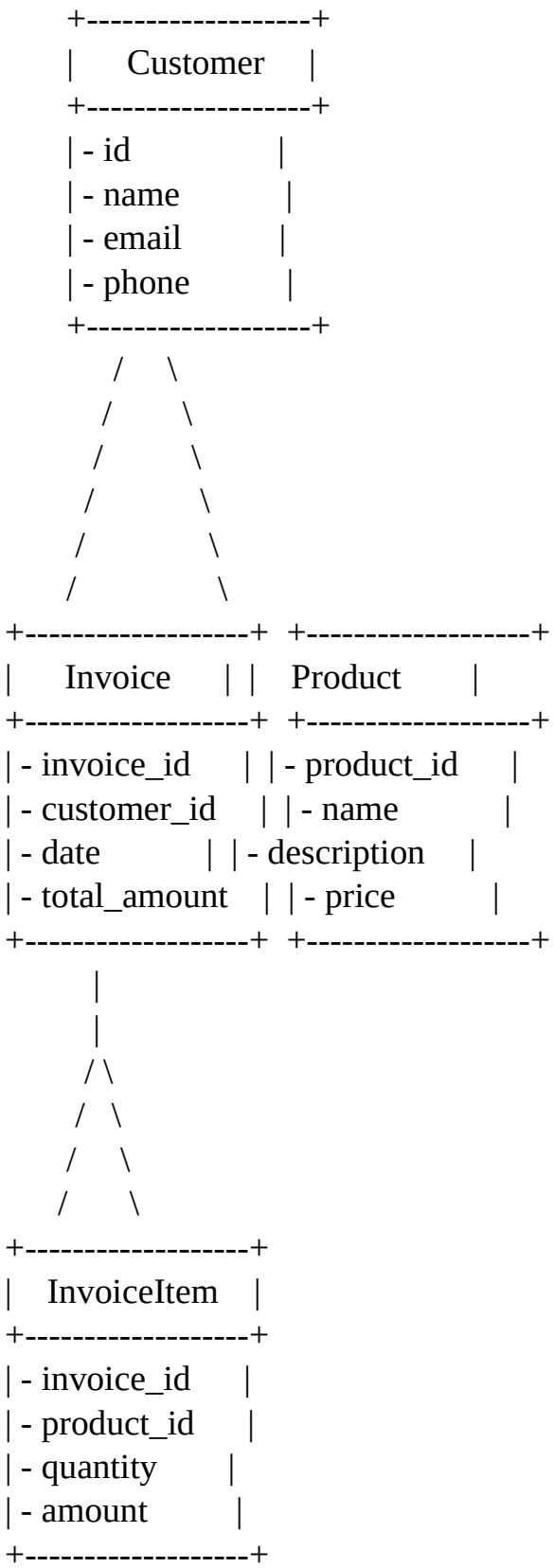


6.5 Entity Relationship Diagram:

6.6 **ER Diagram of the Database :**



- **Database Structure**



6.7 ER Diagram Features

The above shown is an entity-relationship diagram, depicting the working of the food ordering system. ER diagram reflects the relationships that various entities involved in the system share among themselves, along with the entities.

Following are the description of entities involved in food ordering system:

Customer

This represents the set of customers, which are the clients who will be using this application.

The customers are for whom the system is being designed. Its attribute set includes:

Name:

This is the name of the customer, searching or purchasing the products. When signing up to the website the name of the customer is stored, this is done for the future referencing and maintaining the user's data record (history). It is the composite attribute that contains two more attributes that are First-Name and Last-Name. That contains the user's first name and last name.

id :

This is the identification number assigned by the admin to the users so as to identify them uniquely in the future. This identification number is helpful in fetching data of the individual user from a big set. This is mainly to manage the huge database system where the entire data is being stored. It is a permanent identification number given by the admin to the customer to maintain customer history.

email :

This is the identification number given to determine and manage the sequence of service. Since multiple customers will place orders, so as to schedule whom to give the delivery first is determined by the help of this number, so as to maintain consistency in the system working procedure. It will be unique for each order a day.

But the same id can be repeated on a new day, as it is mainly for the restaurant's reference and to prevent any type of conflict.

Address

This field is for the physical address of the customer where the restaurant authority is required to deliver the parcels. It may or may not be the same as the customer's permanent address or residence, but can be the office place or any place.

Its attribute includes:

Address :

An identity through which categorization of places may be done. The address may or may not be unique for each customer registered. But still, this identity helps the delivery person to identify the right place to deliver.

country :

It is the pin code or the postal code of a region, and which is of utmost importance to any address, since multiple places, streets, bungalows with the same name exist. This is even important in any national-level identification of address. Also, this will help the owner in surveying which region has more demand so as to expand their business in that region.

Phone :

The user's contact number is something that must be correct because if at some point in time the delivery person gets confused with the address, it can be used for confirmation. Also, the restaurant authority can contact their customers for any type of feedback or know if the delivery service is good or not.

Invoice

The customer's place order, which is not only still here, there is some work that needs to be done in the database in order to maintain records for keeping track on a monthly basis.

InvoiceNo :

This is the identification number given to determine and manage the sequence of service. Since multiple customers will place orders, so as to schedule whom to give the delivery first is determined by the help of this number, so as to maintain consistency in the system working procedure. It will be unique for each order a day. But the same id can be repeated on a new day, as it is mainly for the restaurant's reference and to prevent any type of overlapping of thoughts between customers and owners. It is mainly for the chef's preference.

duedate:

This is the identification number assigned by the admin to the users so as to identify them uniquely in the future. This identification number is helpful in fetching data of the individual user from a big set. This is mainly to manage the huge database system where the entire data is being stored. It is a permanent identification number given by the admin to the customer to maintain customer history.

price :

This attribute manages the total price sum of the orders the user has made in one attempt. It is one of the most important attributes since most of the time people change their menu order list contents depending upon their needs, health, and economic situation.

Invoice Status :

Time is something most important to be valued. And one of the major reasons behind the success of this food ordering system. So, managing this cause becomes a goal to be completed. In order to maintain the business work better, the authority must stick to its commitment.

Product

It defines the payment to be done by the customer for an order placed from the web store at a worth price. Also, various security encryption mechanisms have been used, so the customer details of accounts and other credentials are safe and secure.

Product-type :

The user is provided with lots of options that he/she can opt for making the payment depending upon their ease. There are many choices available for net banking, use of wallets like pay and I-cash cards, also credit card and debit card options are available too.

PName :

It is for the benefit of the user as well as the website owners since the payment-id is helpful in maintaining the payment record in the database, as well as it is also provided to the customer after the successful completion of payment. As later customers can claim anytime that they have already done the payments and the owners cannot deny. So, it is useful to prevent any kind of fraud from both sides.

Price :

It is the record of the total sum amount the user needs to pay, and after the payment, it is used to update the server-side database to keep the record of the net profit or loss on a daily basis.

Registration

The base of any company, restaurant, or hotel is its employees. It is said that an organization is known by its employees and work. Employees will work honestly and with complete dedication if they are paid sufficient money. On the whole, it’s just like a food cycle, everyone depends on somebody.

Username :

The name of the worker is important to maintain their database of work and payment records.
Also, if any complaints are filed then it is required.

Password :

Time is something most important to be valued. And one of the major reasons behind the success of this food ordering system. So, managing this cause becomes a goal to be completed. In order to maintain the business work better, the authority must stick to its commitment. Workers are paid for their good work and more than that for completion of work before time.

Invoice-mode

The delivery sequence and choice are not the same for everyone but vary from person to person. It may happen that sometimes a person says no to home delivery as he/she is passing by and can pick the parcel themselves. But it is almost an ideal case.

Urgent:

In some cases, like uninvited guest arrival, late-night, people prefer to pay more and get the order delivered urgently. So, restaurants manage such situations by not following the sequence of order placement, as they are getting more than usual. And with another customer whom they have delayed, they manage it with some small gifts or offers.

Normal:

The usual mode of delivery is followed by the sequence of orders placed. It is the normal and majority case. The hotels do not need to put extra effort to manage these.

6.8 Database Design

In this phase, a logical system is built which fulfills the given requirements. Design phase of software development deals with transforming the client's requirements into a logically working system. Normally, design is performed in the following in the following two steps:

1. Primary Design Phase:

In this phase, the system is designed at block level. The blocks are created on the basis of analysis done in the problem identification phase. Different blocks are created for different functions; emphasis is put on minimizing the information flow between blocks. Thus, all activities which require more interaction are kept in one block.

2. Secondary Design Phase:

In the secondary phase the detailed design of every block is performed

The general tasks involved in the design process are the following:

1. Design various blocks for overall system processes.
2. Design smaller, compact and workable modules in each block.
3. Design various database structures.
4. Specify details of programs to achieve desired functionality.
5. Design the form of inputs, and outputs of the system.
6. Perform documentation of the design.

System reviews

6.9 User Interface of Invoice Management System System

This is one of the main tasks of the developer to design a graphical user interface that the user is attracted to and can use easily, in one word it should be user-friendly. So, for this, you should have a better understanding of customers' likes and dislikes and the features that are in trend and mesmerize the public easily. Initially we need to locate the targeting people and what kind of application they need. An online food ordering system allows your business to accept and manage orders placed online for delivery or takeaway. Customers browse a digital menu, either on an app or website and place and pay for their order online. Sales Module is used to manage the Sales. Delivery Module: It has been developed for managing the Delivery. Payment Module: It manages the Payment. Customer Module: Customer operations will be managed by Customer module. After getting all this information we should start to design the application.

Following Application:

- Grow your visibility.
- Know your customers.
- Grow your order number.
- Increase staff productivity.
- Optimise labour costs.
- Increased order values.

6.10 Feasibility Study

After doing the project Online Food Ordering System, study and analyze all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements

A. Economical Feasibility

This is a very important aspect to be considered while developing a project. We decided the technology based on the minimum possible cost factor. All hardware and software cost has to be borne by the organization. Overall we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for the system.

B. Technical Feasibility

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different types of frontend and backend platforms.

C. Operational Feasibility

No doubt the proposed system is fully GUI based and is very user friendly and all inputs to be taken are all self-explanatory even to a layman. Besides, proper training has been conducted to let the users know the essence of the system so that they feel comfortable with the new system. As far as our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing.

Testing

7.1 TEST CASE

We have implemented various test cases on our software i.e., invoice management. A TEST CASE is a sequence of actions performed to ensure that a certain feature or operation of your software application is working properly. A Test Case contains test steps, test data, precondition, post condition developed for specific test scenarios to verify any requirement. The test case contains specified variables or circumstances that a testing engineer might use to compare expected and actual outcomes in order to assess whether a software product meets the customer's needs.

.

A test case includes these elements:

- The purpose of the test or description of what requirement is being tested.
- Test data - Test data is a specification of the data elements, values or set that define how to satisfy the test intent.
- The setup to test - Version of application, hardware, software, operating system, access level, logical or physical date and any other setup information being tested.
- Actions and expected results
- To use the data to analyze rework process and to make changes to prevent defects occurring in the system

7.2 TESTING STEPS

Unit Testing

The purpose of Unit Testing is to ensure that each unit in the system functions properly. This testing strategy was applied to each object in the system. A Java test class containing methods to assess objects, calling each method with an assortment of parameters to test whether the values returned were appropriate. These test classes ensured that valid input data was accepted and invalid data was rejected. Each unit within the application was tested to verify that all links and buttons navigated as expected. Usability testing has also been incorporated to examine whether error messages are clear and understood. The database schema was also monitored to ensure insertions, updates and deletes were occurring and with the expected changes. All the tests have proved successful. For those tests which failed initially, the causing factor was identified and remedial action was taken in order to pass the test concerned. The detail of remedial action taken has been documented. Security testing has been incorporated to verify that the expected result is in face, the actual result.

Alpha Testing

Alpha Testing is a type of software testing performed to identify bugs before releasing the software product to the real users or public. It is a type of acceptance testing. The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.

Beta Testing

It is done to make sure that customers are satisfied with the application experience. The design of the Invoice Management System should be very easy and user-friendly because the major chunk of the population which uses Invoice Management System isn't highly educated. Simple and effective design is the key to reach to a high number of customers.

Performance Testing of Invoice Management System

Performance testing is a very crucial part of testing Invoice Management System applications. We need to evaluate the behaviour and stability of Invoice Management System applications when high stress, load, concurrency, volume is applied.

Different performance testing use cases are

- validation of application behavior under high load
- validation of application behavior under peak usage
- validation under high constant load and identification of various bottlenecks in the application

Some other use cases are optimization of various APIs, validation of response time of APIs, hardware, and resources utilization under high load. These validations need to be performed so that the Invoice Management System can become robust enough to handle any amount of load.

Usability Testing Invoice Management System Applications

It is done to make sure that customers are satisfied with the application experience. The design of the invoice management system should be very easy and user-friendly because the major chunk of the population which uses invoice isn't highly educated. Simple and effective design is the key to reach to a high number of customers. Some usability use cases for Invoice Management System applications are

- making sure that functionalities are easy to find
- navigation should be easy and user-friendly
- buttons of the application should be visible
- verification that font should be of appropriate size so that anyone can read them
- verification that the user can undo the last operation and finally
- the design of the application should be appealing, simple, and user-friendly

Some other use cases are validation that the registration process for food delivery apps should be very simple so that anyone can register and start using it. Addition of address should be very simple, menu navigation should be intuitive and hassle-free, validation that right images should be displayed against food items, images should be clear with proper colours.

Validation that offers and promotions should be easy to find, Bill should be understandable, delivery time and location should be easy to find, making sure that rating process is user-friendly and finally, the checkout process should be damn simple.

Compatibility Testing of Invoice Management System Applications

This type of testing verifies that the Invoice Management System works well in different environments like OS: iOS and Android Invoice Management System applications should work fine on all mobile devices with different operating systems and with different sizes. We also verify that it works well with high configuration and low configuration phones.

This is very important testing as a little bug here and there can cost lakhs to business. Hence, Compatibility Testing should be done by a larger set of people who have access to many devices to ensure that the application is compatible with all of them. A testbed of devices and configurations needs to be maintained for proper compatibility testing.

Security Testing for Invoice Management System Applications

Security Testing is very important so that the personal data of customers should not be at stake. It is done to ensure that customer data is protected against all kinds of threats and attacks. Leakage of data can cost hefty fees to the business.

Some of the use cases of security testing are validation of application behaviour under cyber-attacks, validation that application should now to access the application without proper authentication, and validation of session timeout in times of inactivity.

Invoice Management System have integration of customer credit cards and other card data so security testing becomes very important otherwise customer data can be leaked.

7.5 INTEGRATION TESTING

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface. The objective is to take unit tested modules and builds program structure. All the modules are combined and tested as a whole. The purpose of performing Integration Testing is to ensure that the different portions that make up the system function correctly, when combined to form a single working application. Integration Testing was performed at the end of each phase to prove that the system still maintained functionality. The benefits become more apparent as the system increases in size and functionality. For example, at the completion of phase two, Companies and Users were two separate system entities (at that point in development). However, the addition of Access Controls in phase three relies on successful integration with the Users entity, for login and customizable access rights to be successful in their operations. For such reasons, testing occurred at each phase completion to verify that new additions have not affected existing functionality.

Testing Levels

Testing can also be grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels of testing during the development process can be mainly identified as unit testing, integration testing, system testing and acceptance testing .

FUNCTIONAL TESTING

These are the points concerned during the stress test:

- Nominal input: character is in putted in the place of digits and the system has to flash the message "Data error.
- Boundary value analysis: exhaustive test cases have designed to create an output report that produces the maximum (and minimum) allowable number of table entries.

7.6 WHITE BOX TESTING

The purpose of White Box testing is to certify that the underlying system architecture functions correctly. This contrasts to Black Box testing which examines system output from knowledge about the systems use of syntax. White Box testing has been undertaken for this system, in order to examine the changes of state for each of the database tables within the system. White Box testing has two immediate benefits. Firstly, by examining the table states during update, create and delete processes for example, the developer is able to verify that the right tables are being queried or affected. Furthermore, the developer can query the record contents, to ensure the correct and relevant fields in one or more tables have been correctly queried or affected, as appropriate. The second benefit is more aimed at the client; as such testing can simulate system stress that it may endure once deployed to see how it reacts.

7.7 BLACK BOX TESTING

The purpose of Black Box or functional testing is to assess the systems internal workings. However, the overall aim of this test strategy is to examine results without knowing how the system arrived at that result. In effect, a tester only requires knowledge of the system specification rather than underlying architecture. This caters for such testing to be performed from an end user's perspective rather than a designer perspective. Furthermore, any ambiguities that may exist between the Black Box test results and the original specification are easily detectable, as an unexpected output would occur. The tests performed here are based around users input and actions. The expected and actual system output for each test case has been documented.

7.8 System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing.

7.9 OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hard copy also; the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

7.10 GOAL OF TESTING

"Program testing can be used to show the presence of bug, but never to show their absence." If the results delivered by the system are different from the excepted ones then the system is incorrect and these bugs should be fixed.

7.11 ACCEPTANCE TESTING

User Acceptance Testing In order to determine that the system had met the requirements defined in Section 3, the user and developer had arranged to meet at the end of July 2022. This involved the developer demonstrating the software and the directors using it briefly. User acceptance testing was done at the end of the implementation the developer provided a form which stated the original requirements.

User Acceptance Testing :

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required. This is done in regard to the following point :

a) Input Screen Design

b) Output Screen Design

c) Format of reports and other outputs.

d)

TESTING METHOD USED

We have adopted a testing method which is a mix of both (structural) and black box (functional) testing. For modules we have adopted white box testing. Then we integrated the module into sub - systems and further into the system. These we adopted black box testing for checking the correctness of the system.

Requirements Validated and Verified:

- The data is getting entered properly into database.
- The Screens are being loaded correctly
- The Various functions specified are being performed completely.

7.12 VALIDATION

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

a)The function or performance characteristics confirm to specification and are accepted.

b) A deviation from specification is uncovered and a deficiency list is created.

c)Proposed system under consideration has been tested by using validation test and found to be working satisfactory

7.13 TEST REPORTING

Immediate Purpose:

Provide information to the users of the software system, so that they can determine whether the system is ready for production. It will help to show the progress of testing to clients when they have doubts.

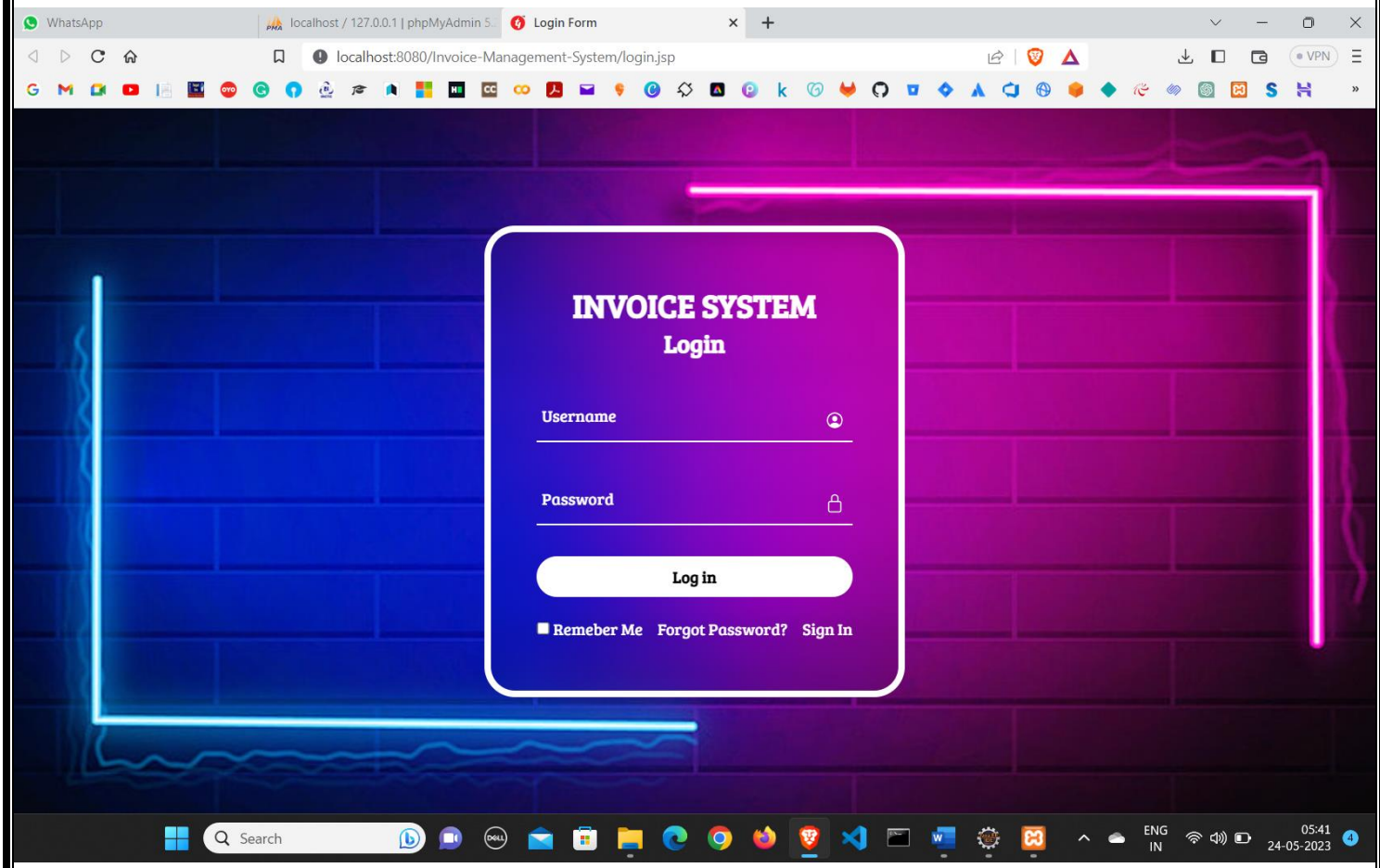
Long Term Purpose:

- To trace problems in the event application of the main functions.

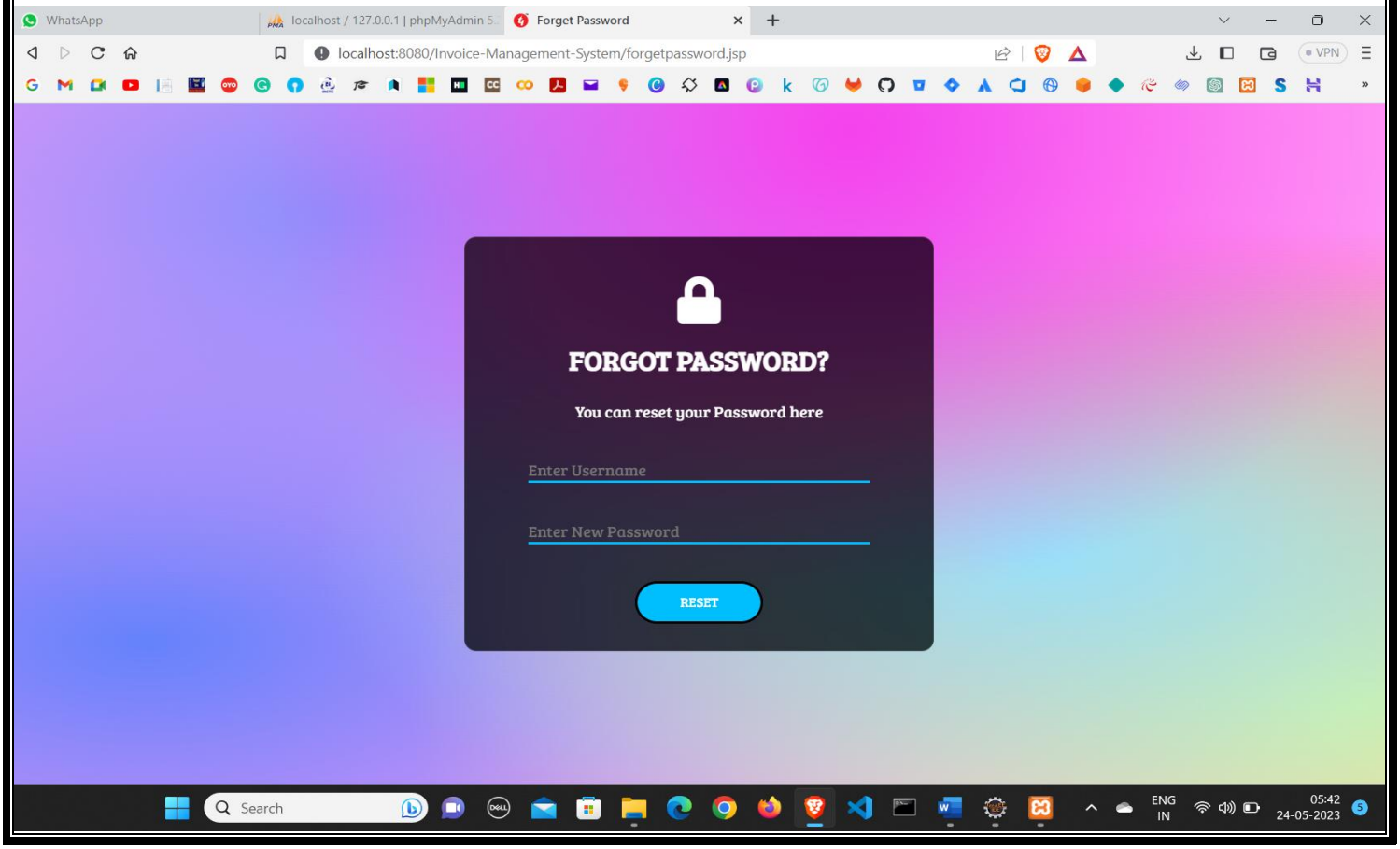
Codes & Screenshot

➢ Invoice System Portal

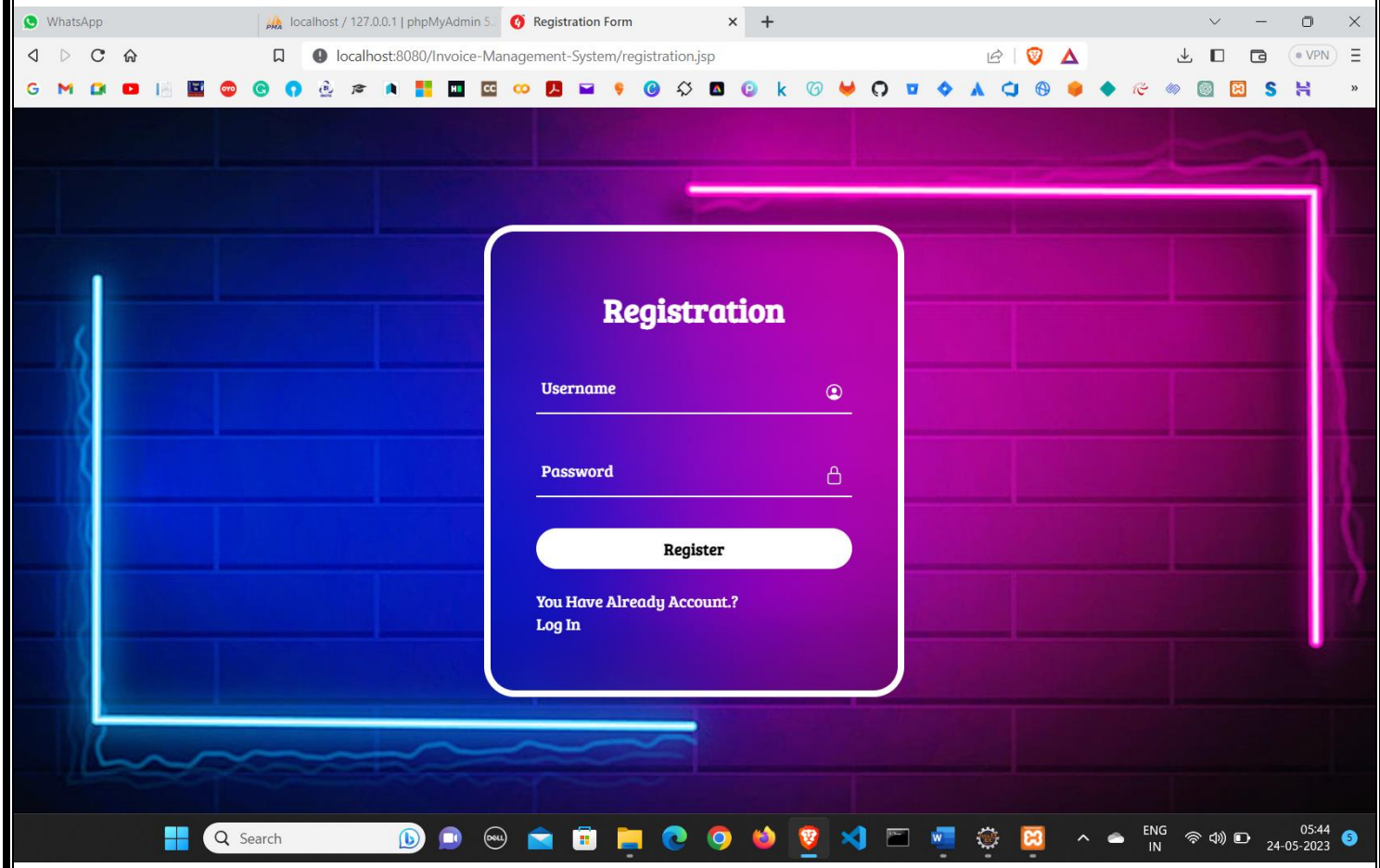
- **Login Page**



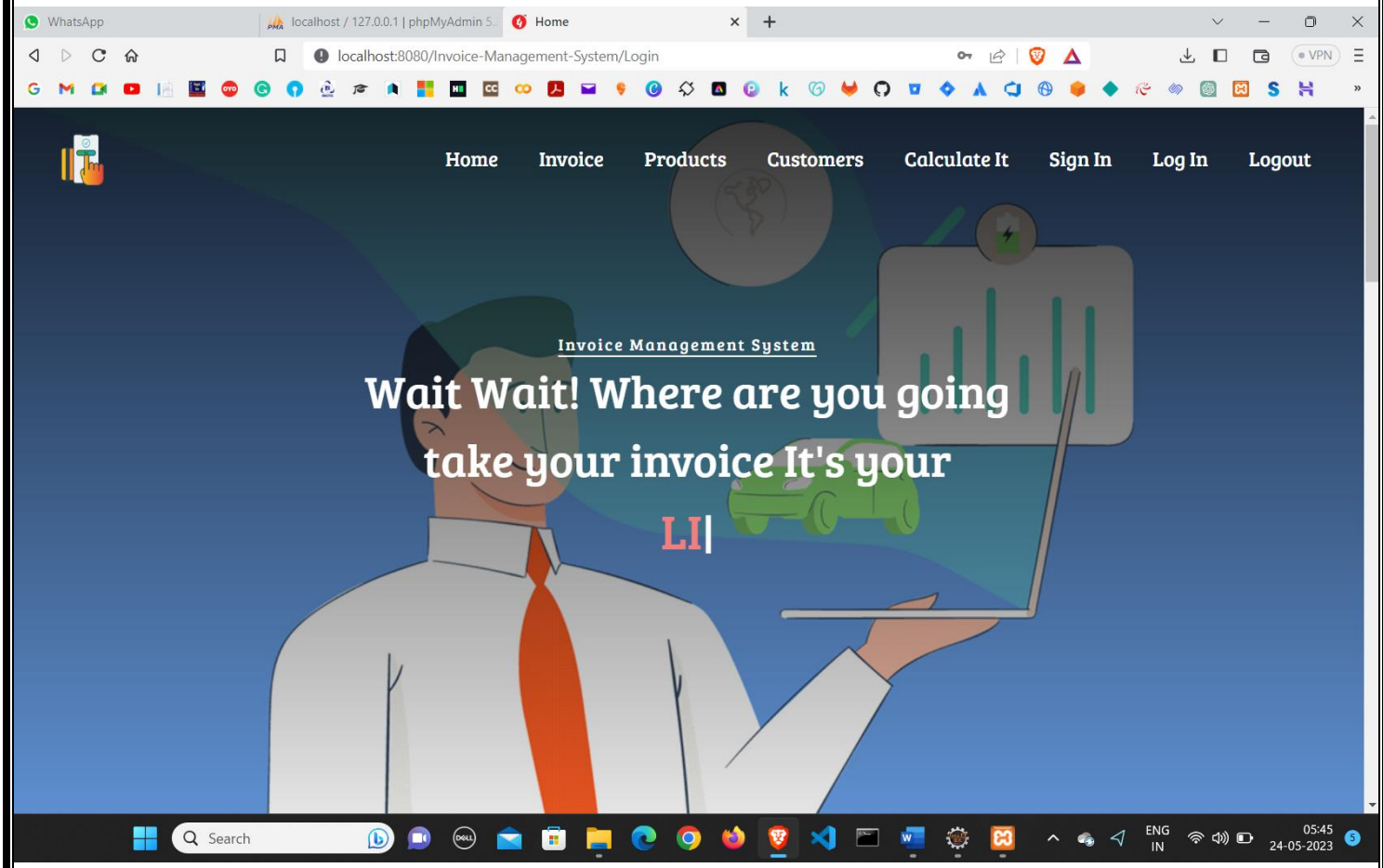
- **Forgot Password**

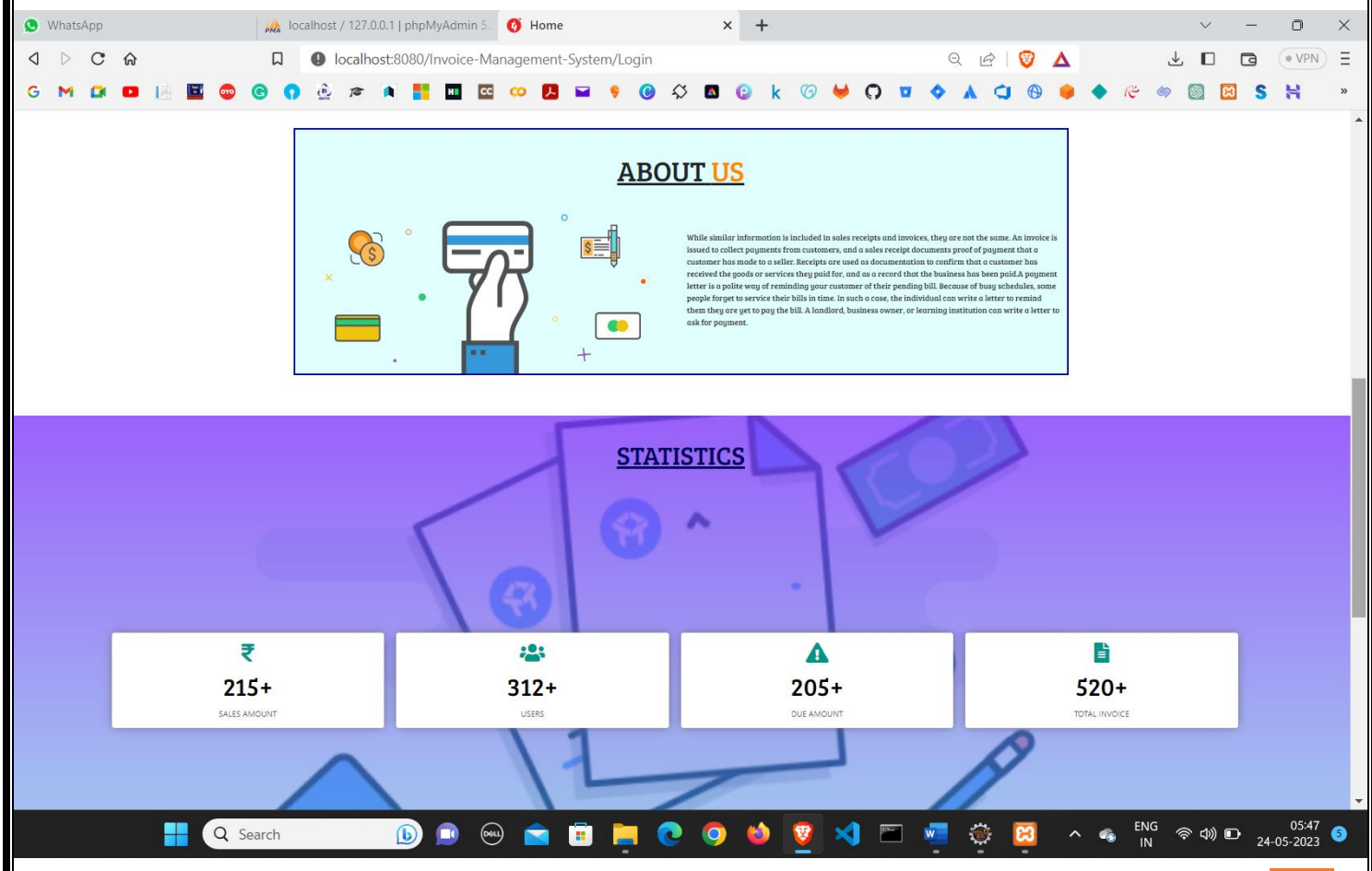


- **Sign In Page**

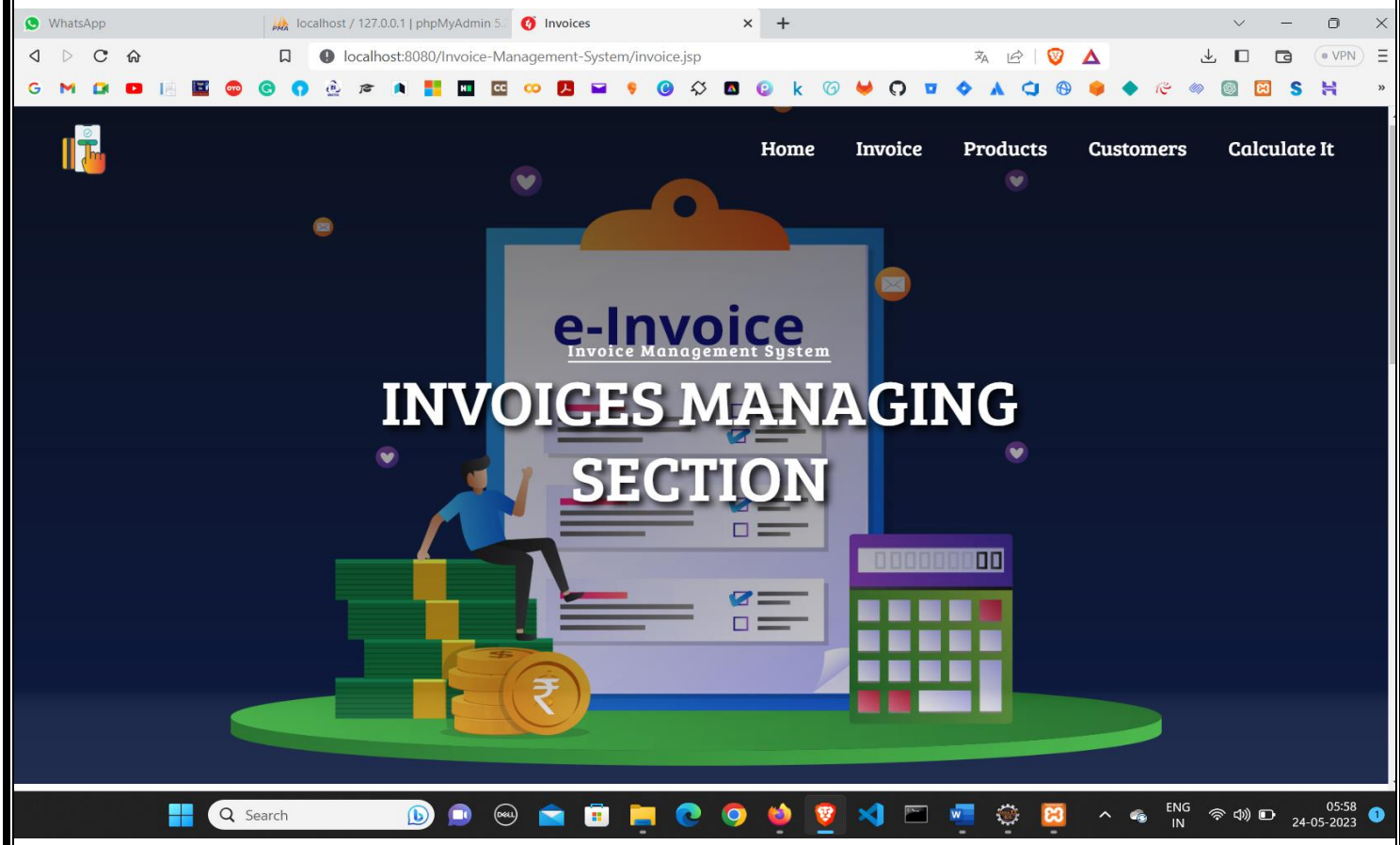

WhatsApp
localhost / 127.0.0.1 | phpMyAdmin 5...
Registration Form
localhost:8080/Invoice-Management-System/registration.jsp
Registration
Username
Password
Register
You Have Already Account.?
Log In
Search
ENG
IN
05:44
24-05-2023

- **Home Page**

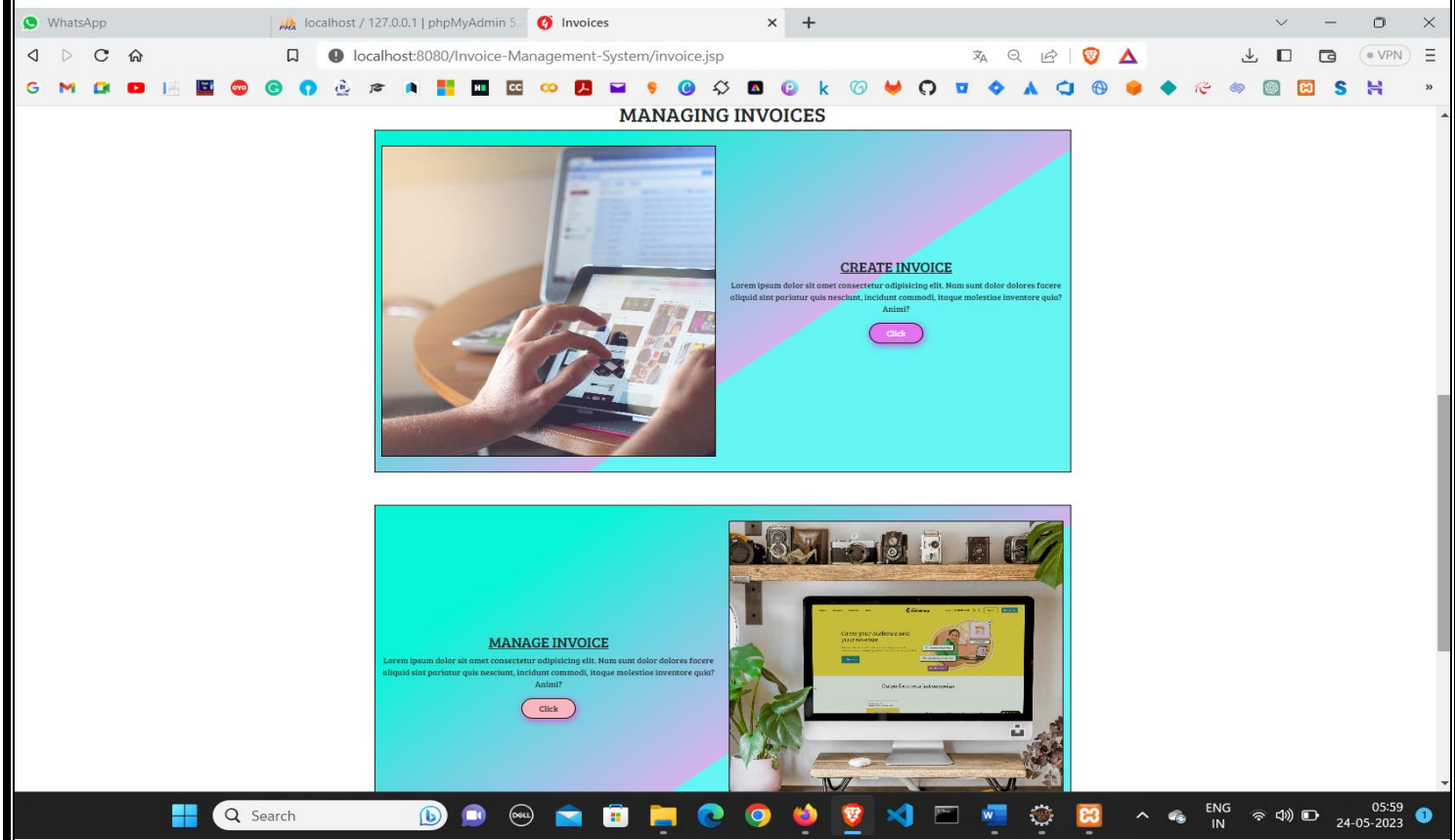

localhost:8080/Invoice-Management-System/Login
Home
Invoice
Products
Customers
Calculate It
Sign In
Log In
Logout
Invoice Management System
Wait Wait! Where are you going
take your invoice It's your
LI


localhost:8080/Invoice-Management-System/Login
ABOUT US
While similar information is included in sales receipts and invoices, they are not the same. An invoice is issued to collect payments from customers, and a sales receipt documents proof of payment that a customer has made to a seller. Receipts are used as documentation to confirm that a customer has received the goods or services they paid for, and as a record that the business has been paid.A payment letter is a polite way of reminding your customer of their pending bill. Because of busy schedules, some people forget to service their bills in time. In such a case, the individual can write a letter to remind them they are yet to pay the bill. A landlord, business owner, or learning institution can write a letter to ask for payment.
STATISTICS
215+
SALES AMOUNT
312+
USERS
205+
DUE AMOUNT
520+
TOTAL INVOICE

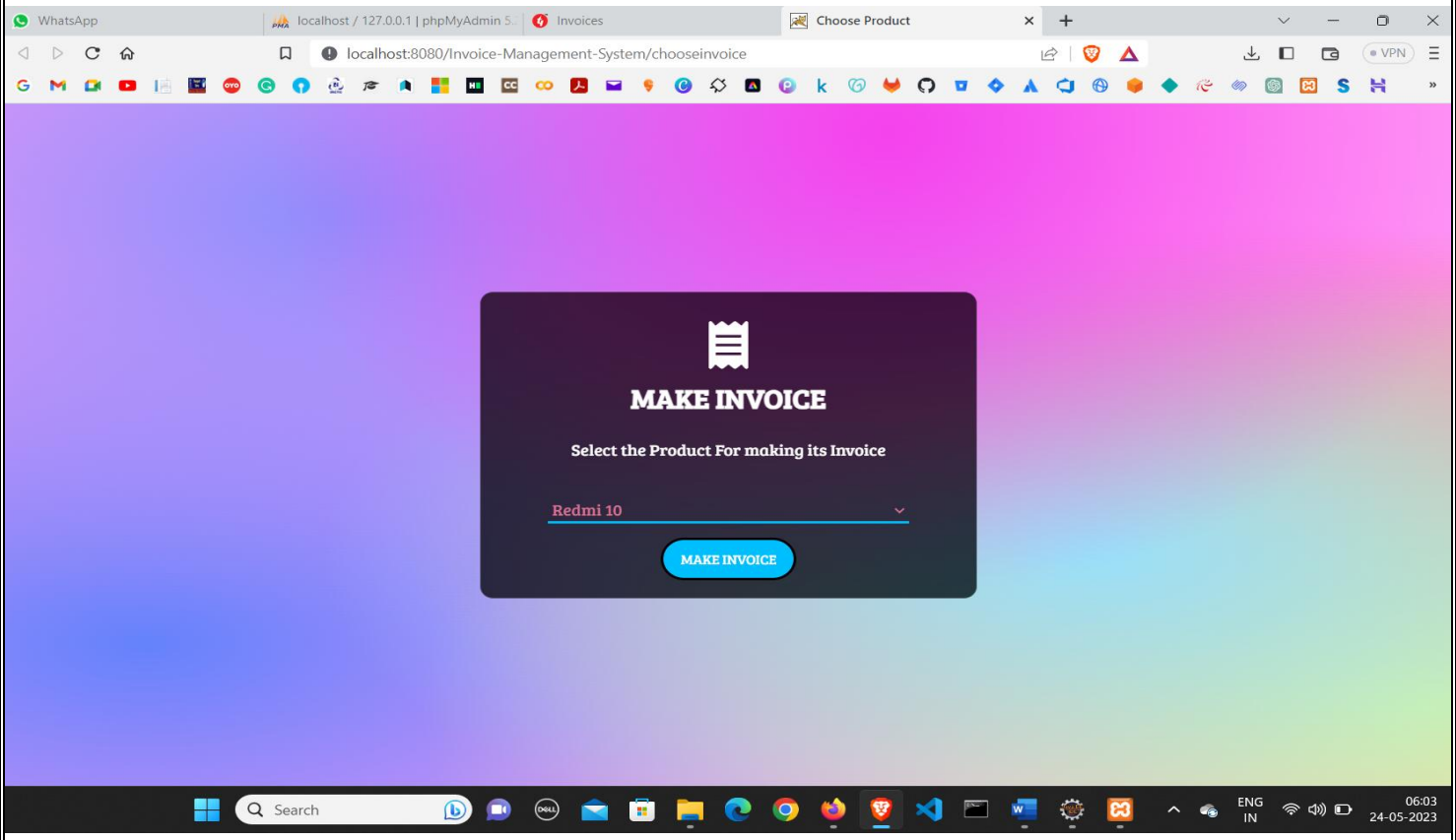
- **Invoice page**



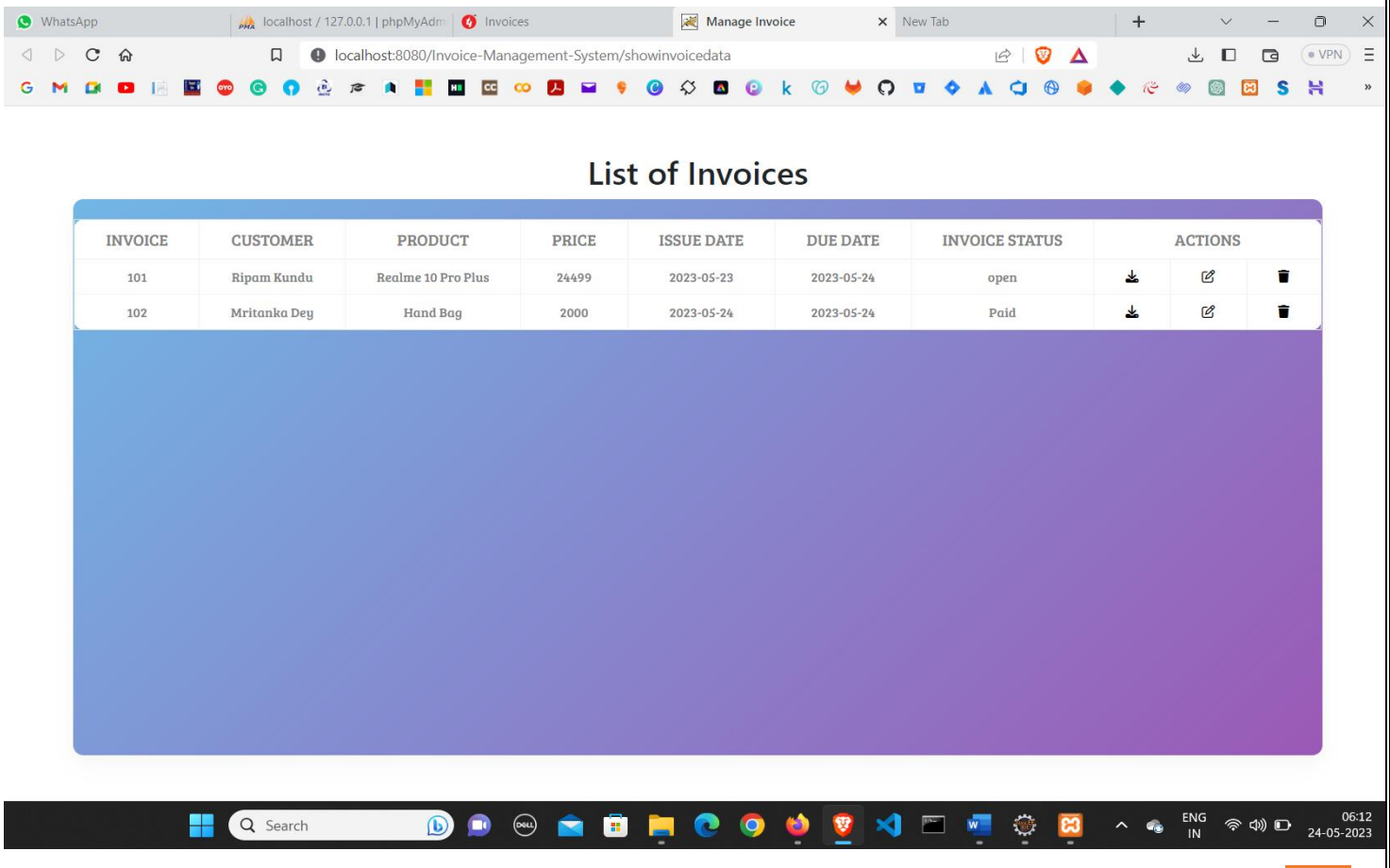
- **Managing Invoice**



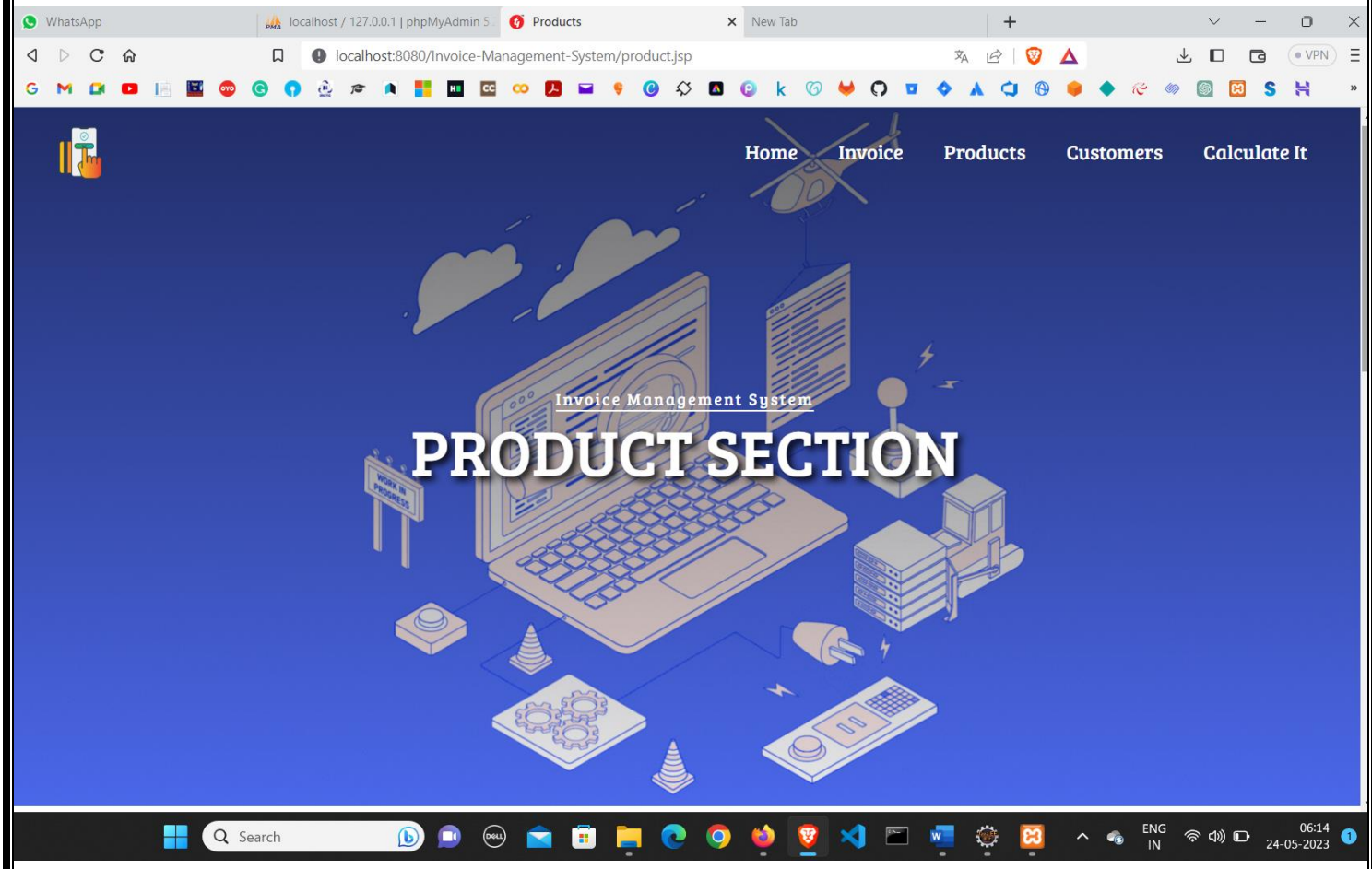
- **Create Invoice**



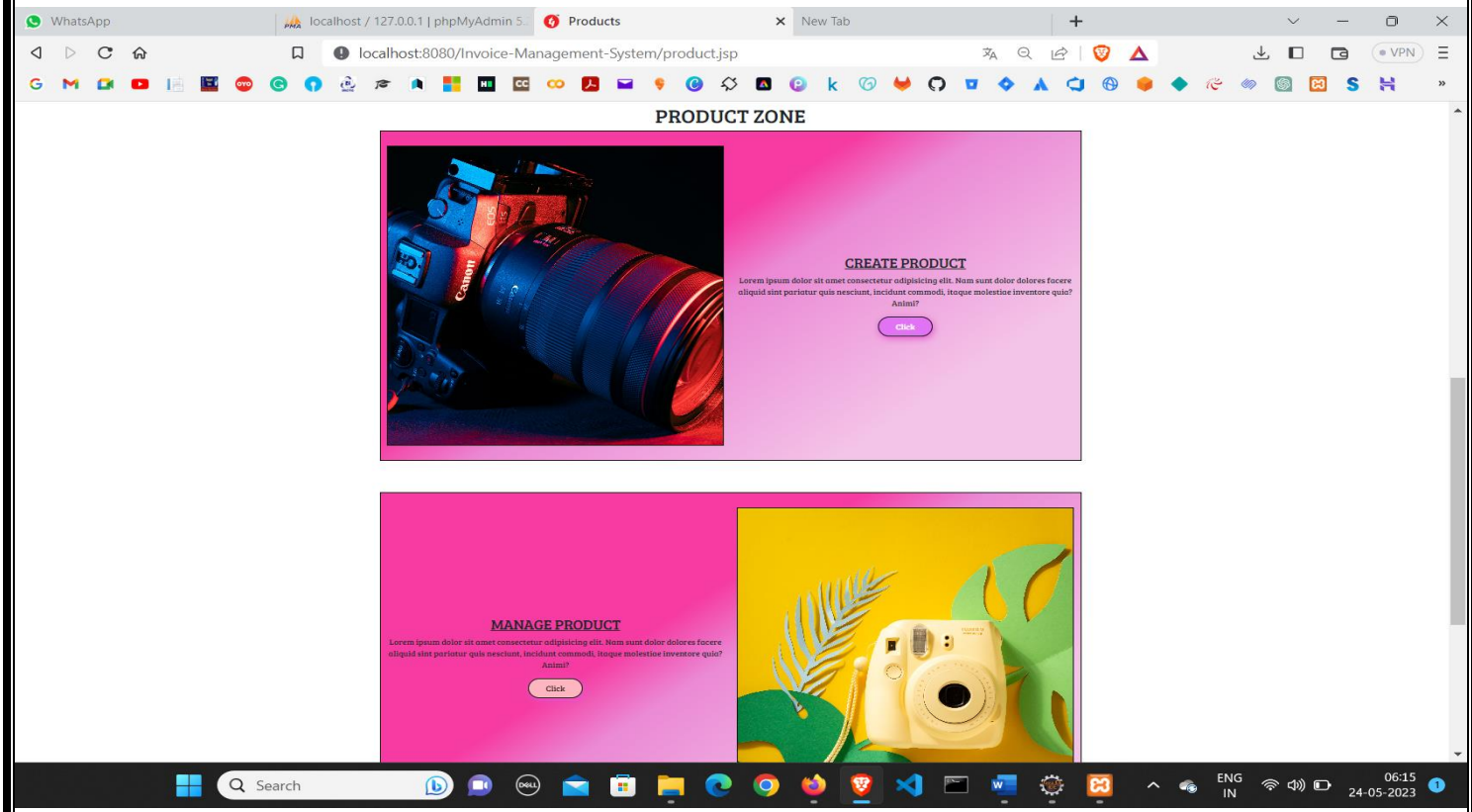
- **Manage Invoice**



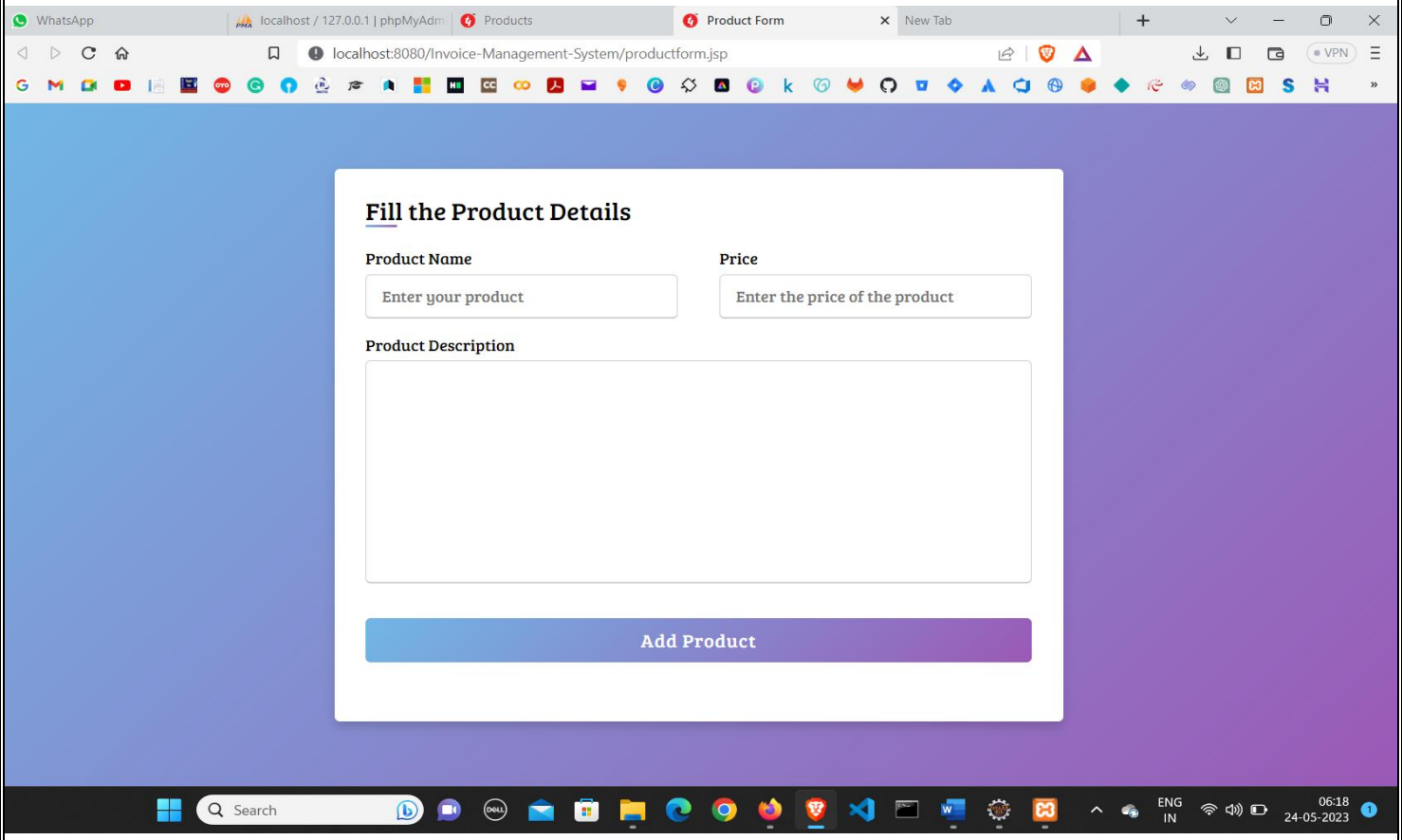
- **Product Page**



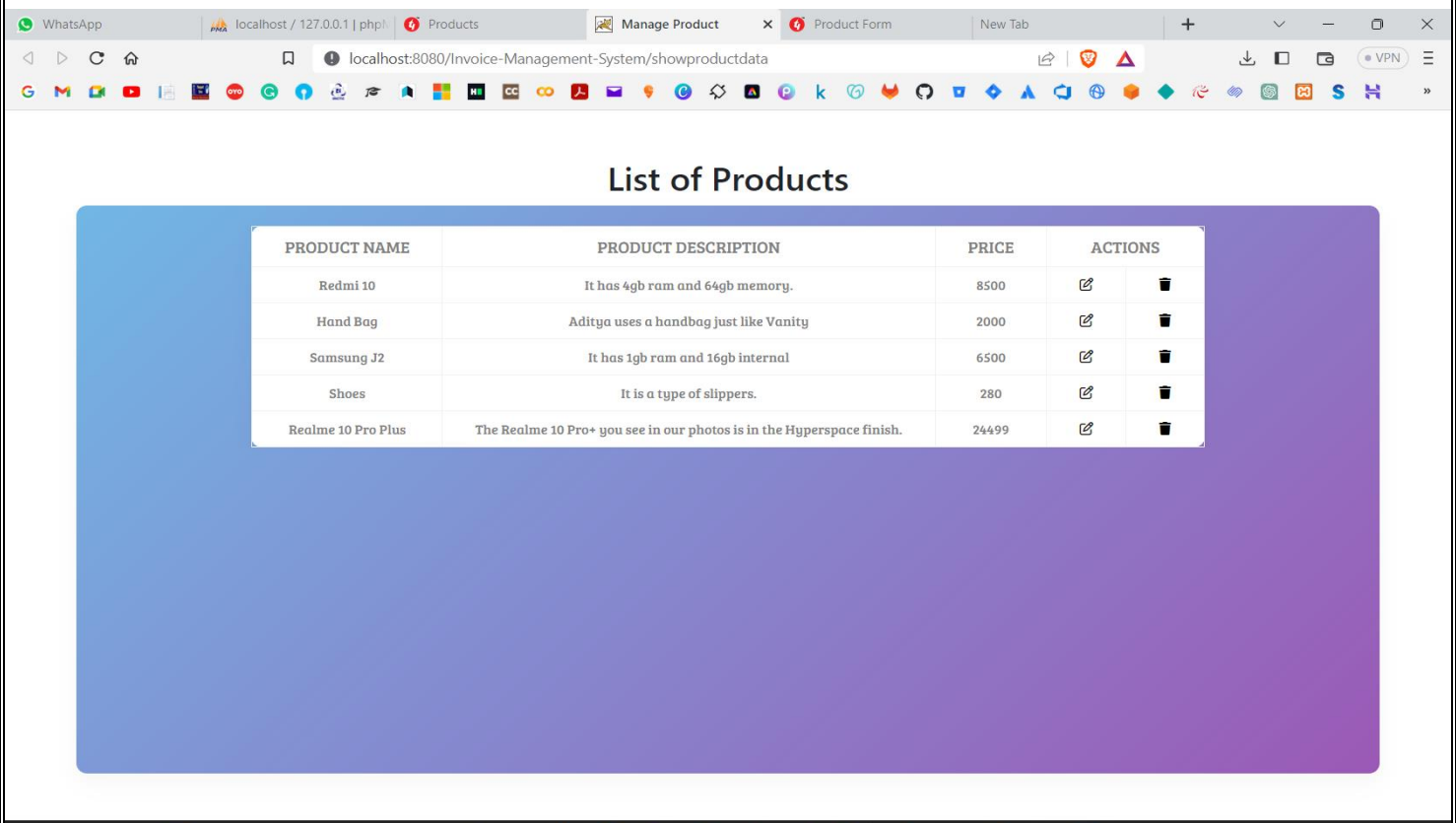
- **Managing Product**



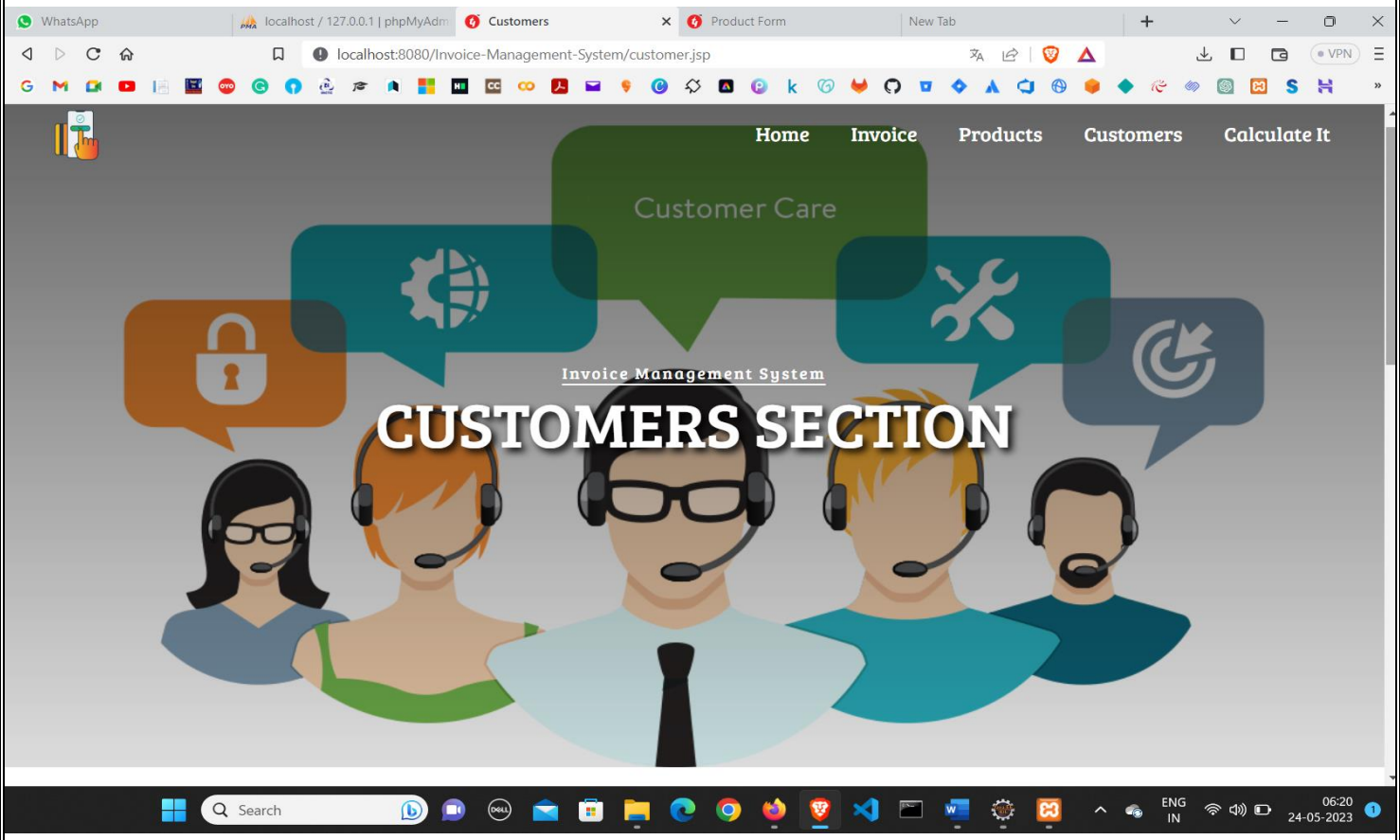
- **Create Product**



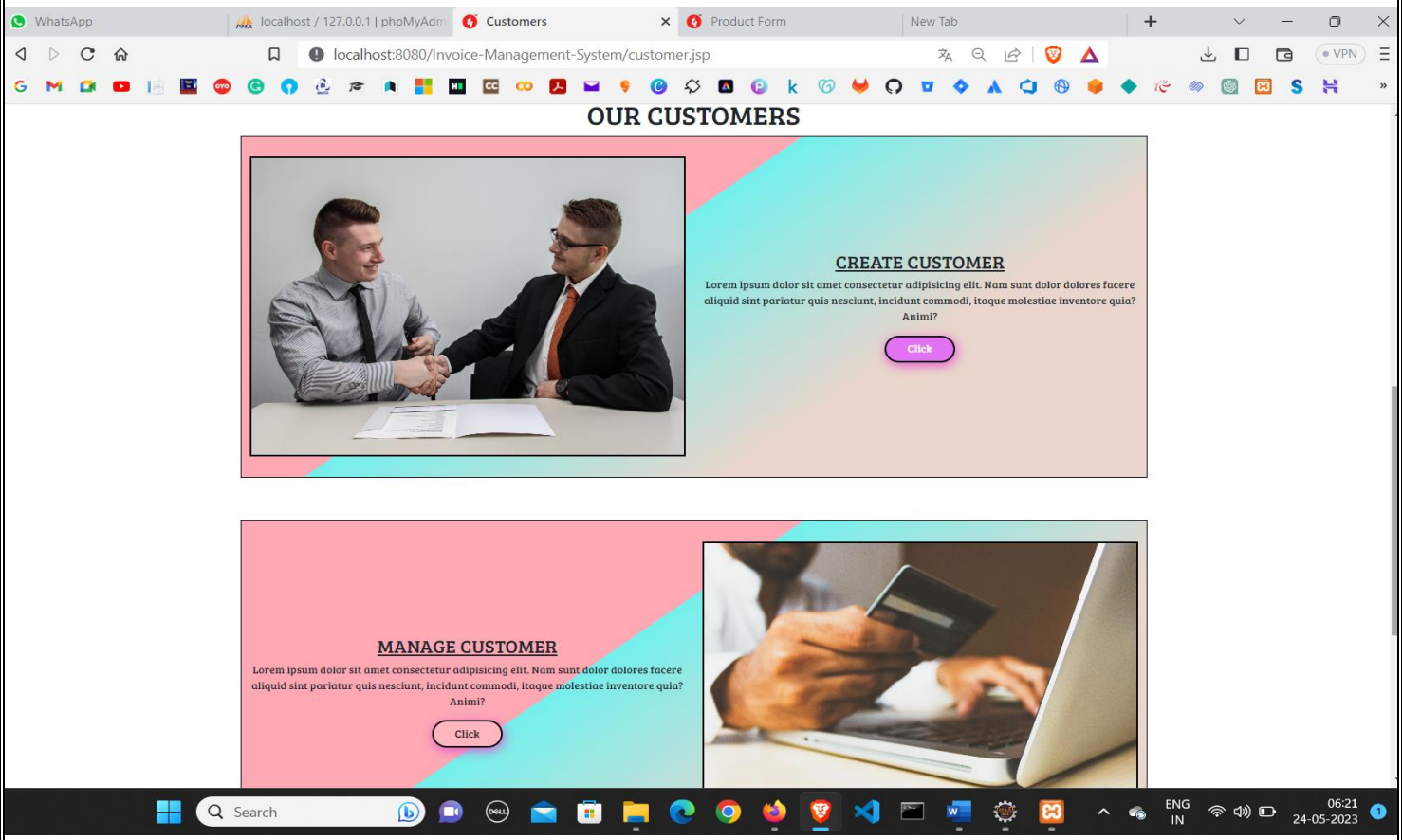
- **Manage Product**



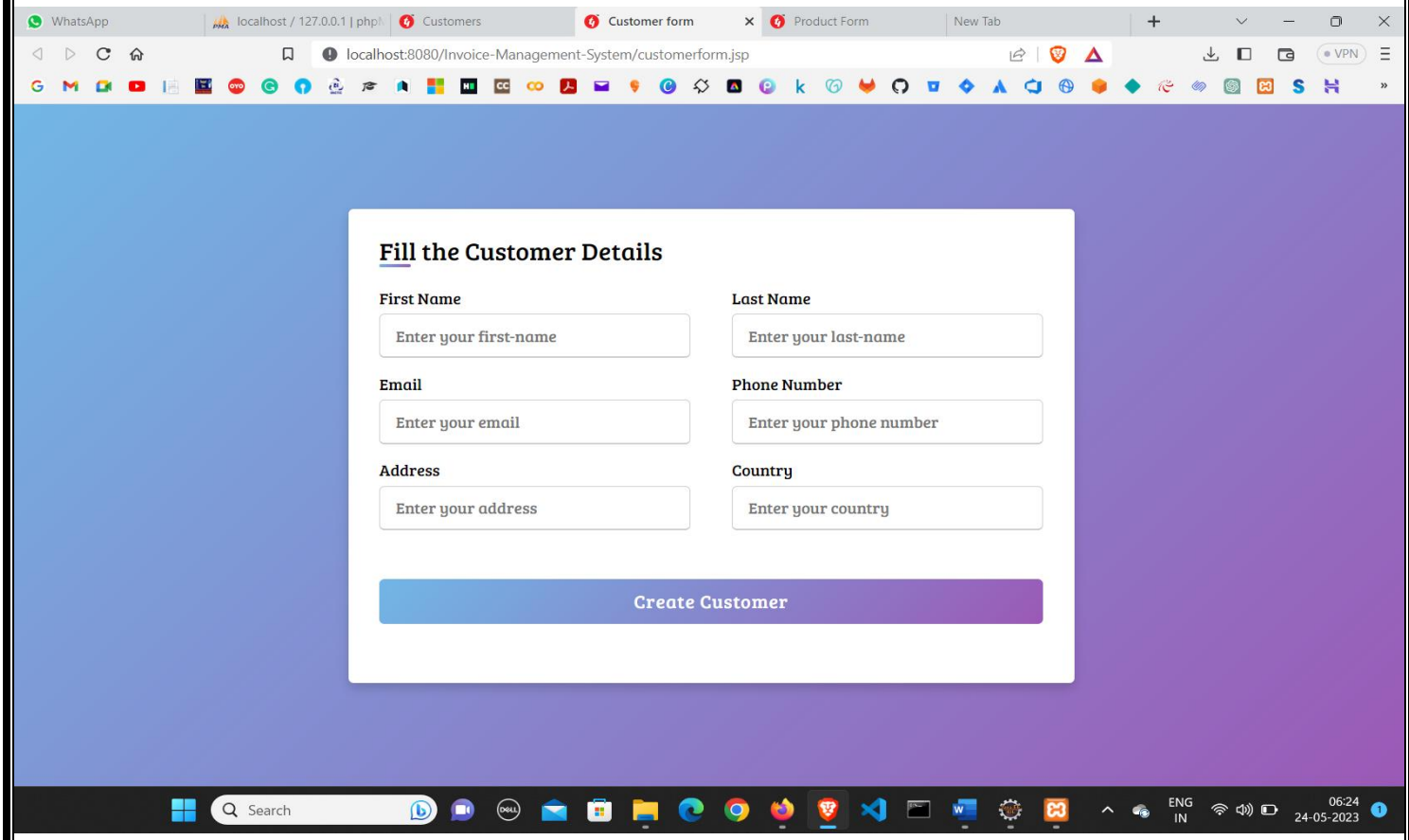
- **Coustomer Page**



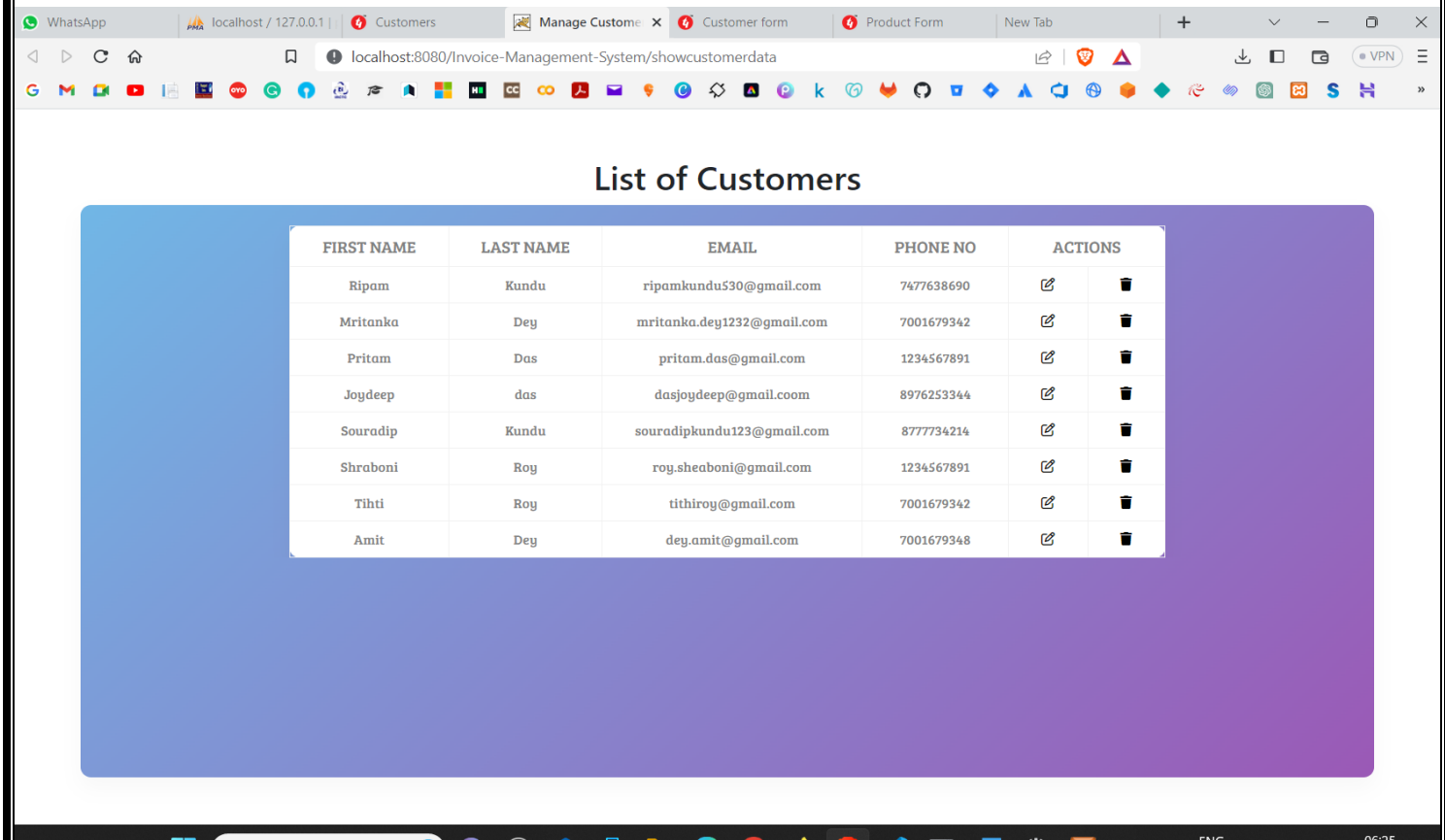
- **Manage Customer**



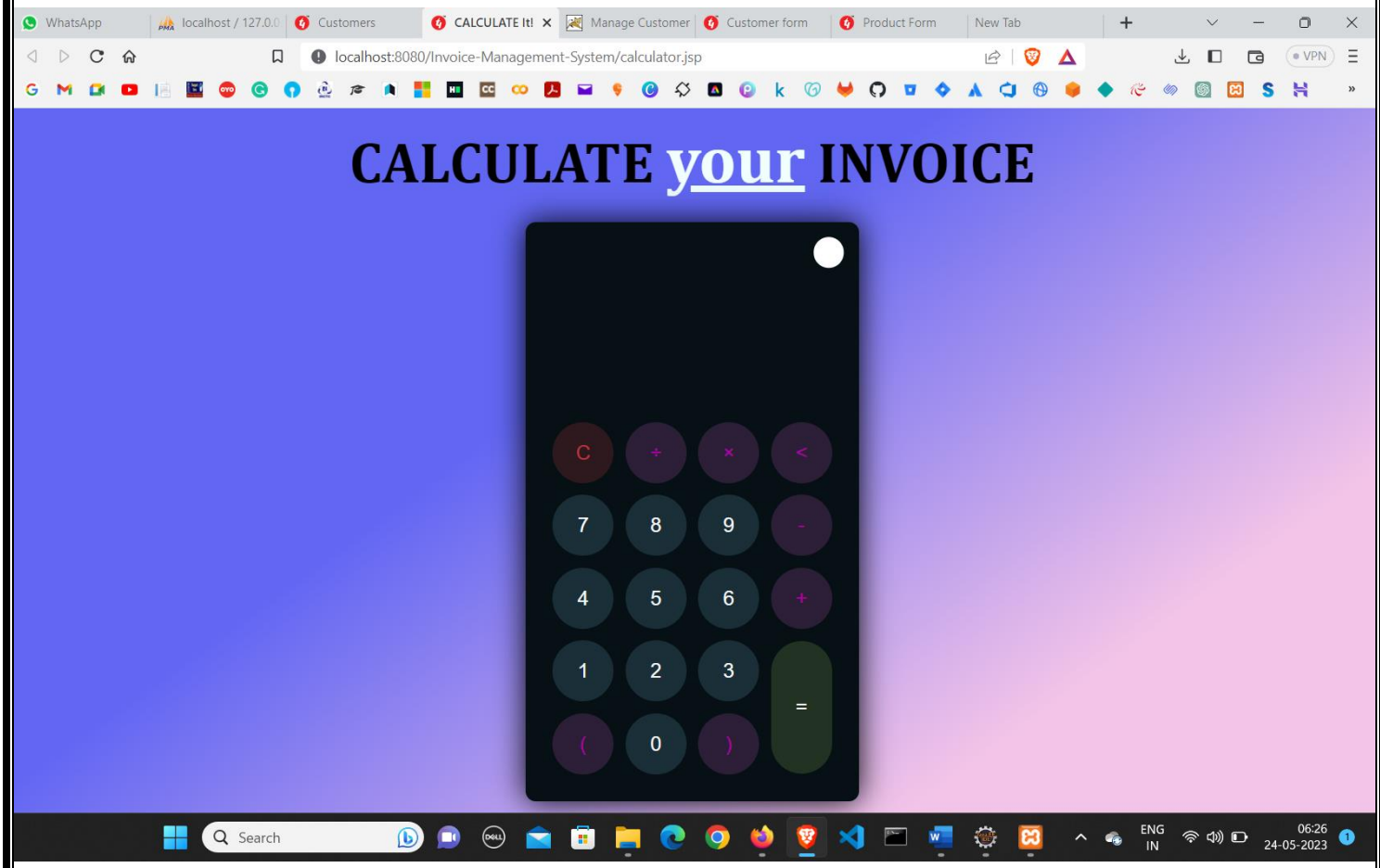
- **Create Customer**



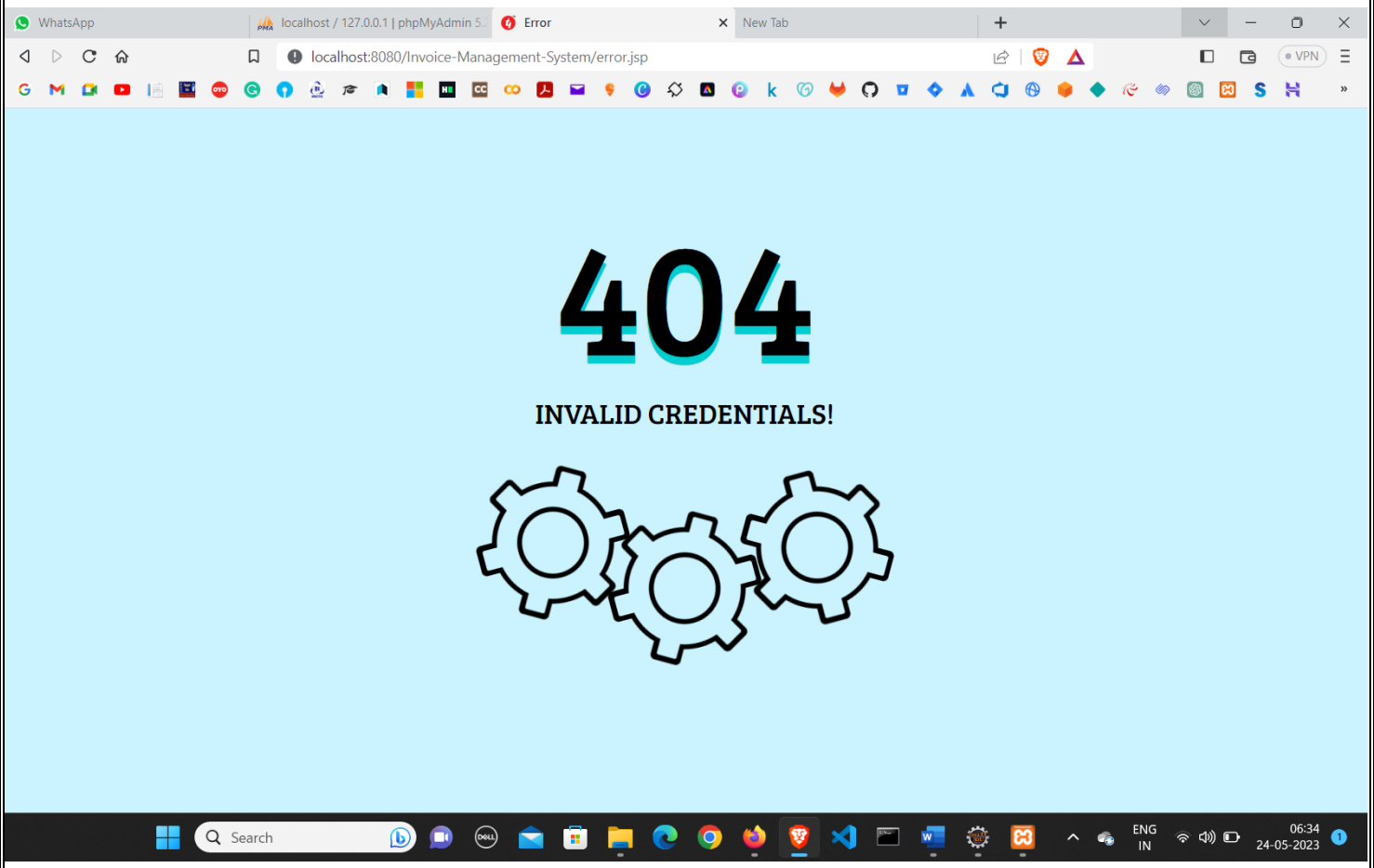
- **Manage Customer**



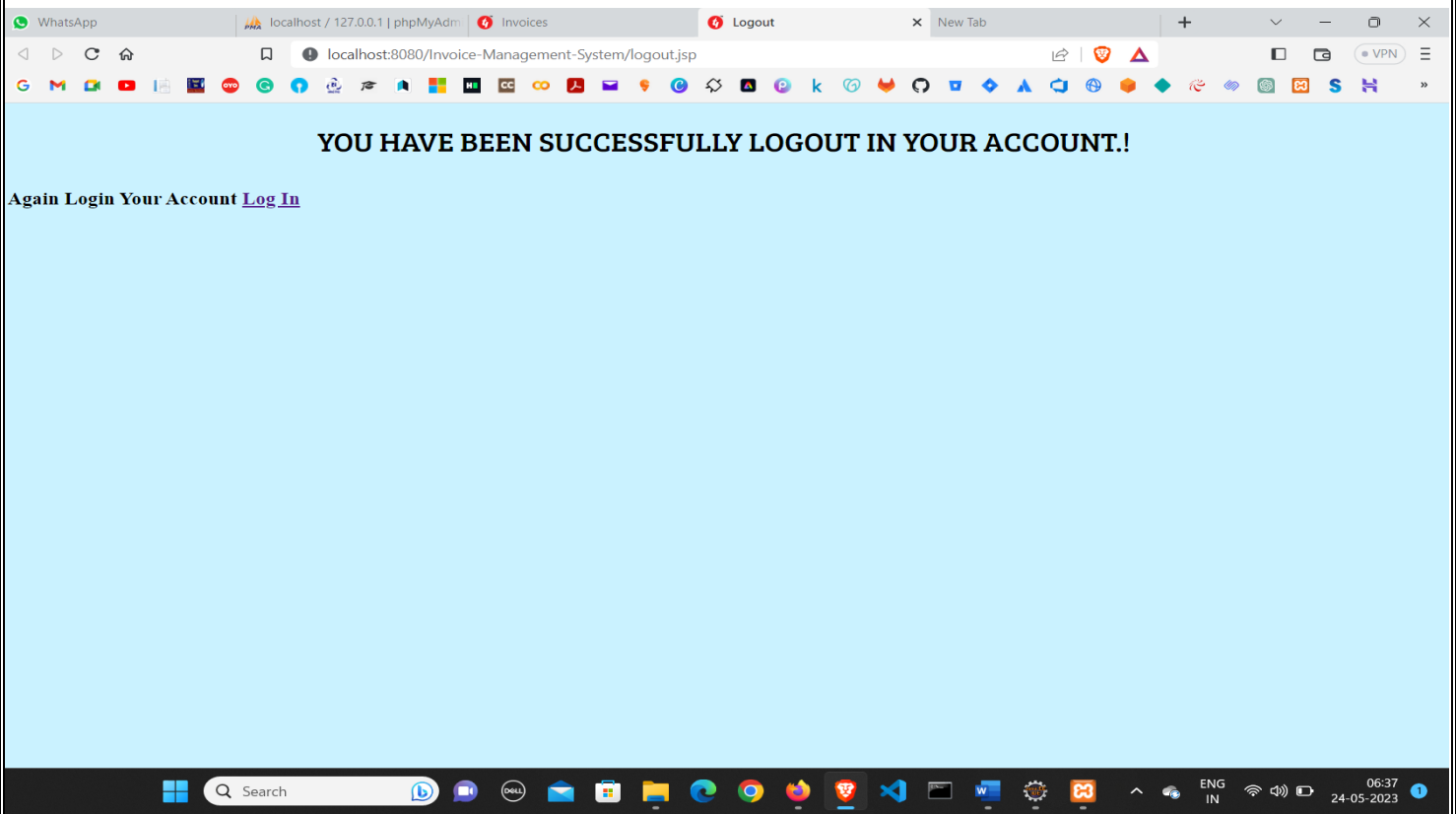
- **Calculate It**



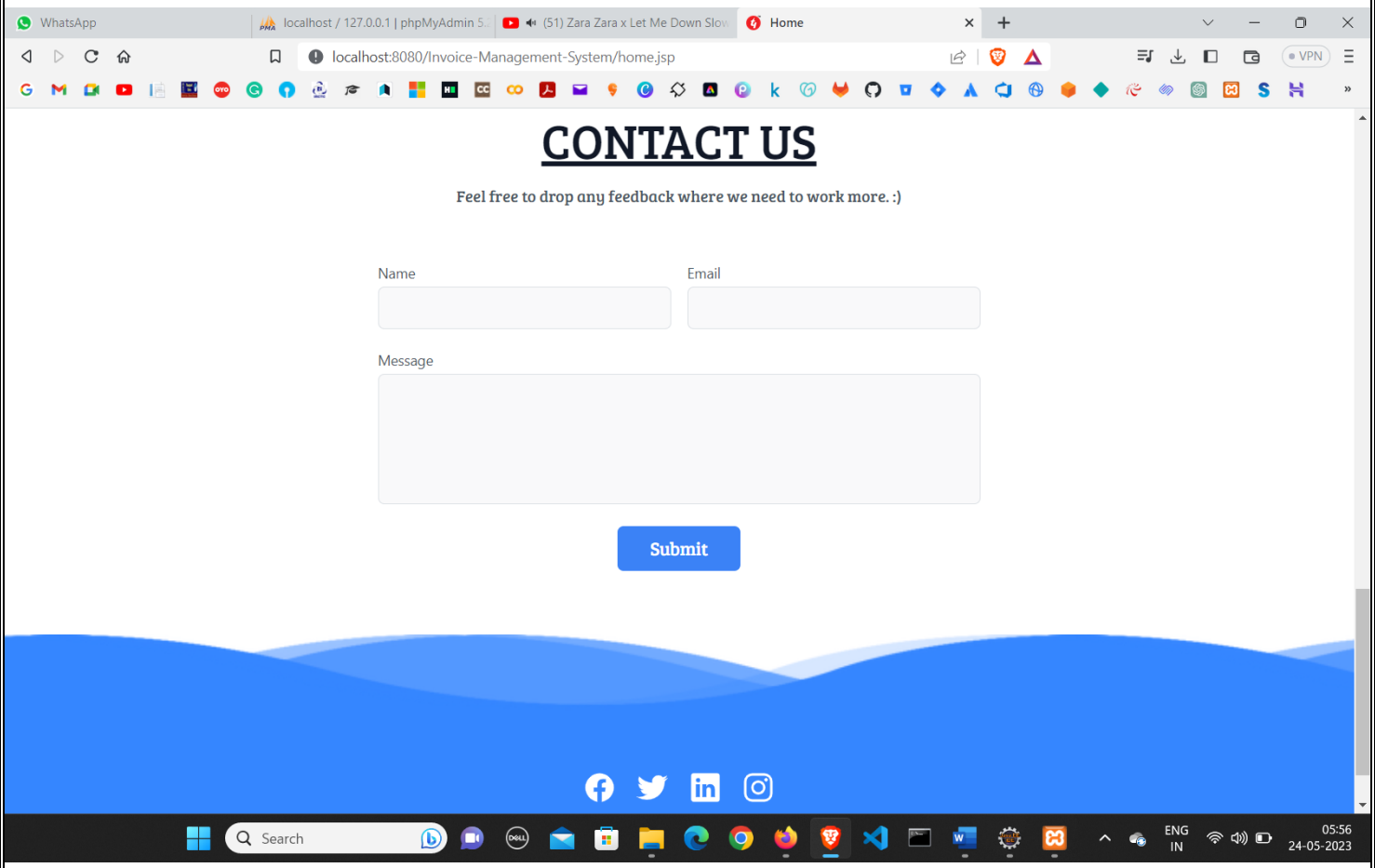
- **Error Page**



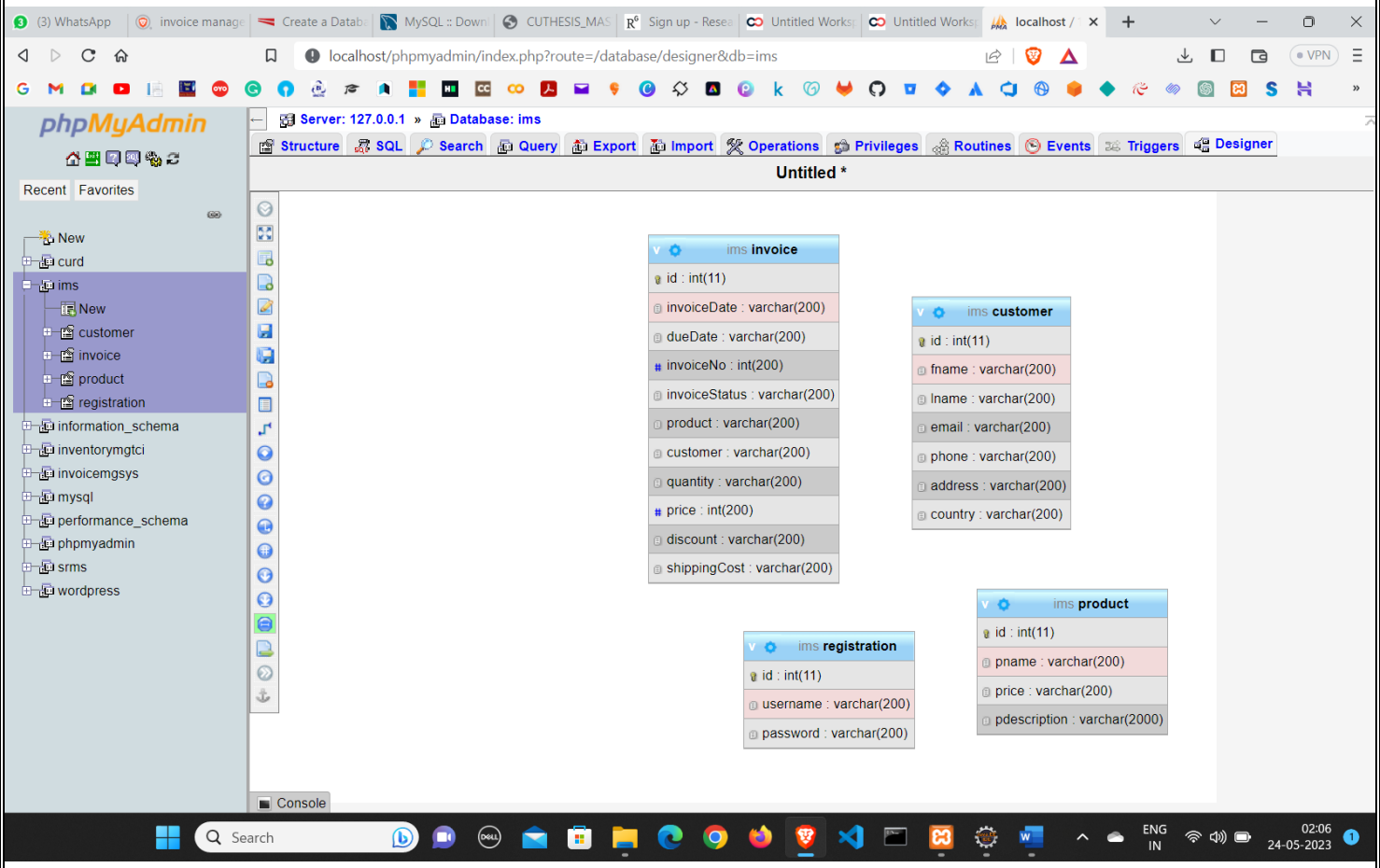
- **Logout**



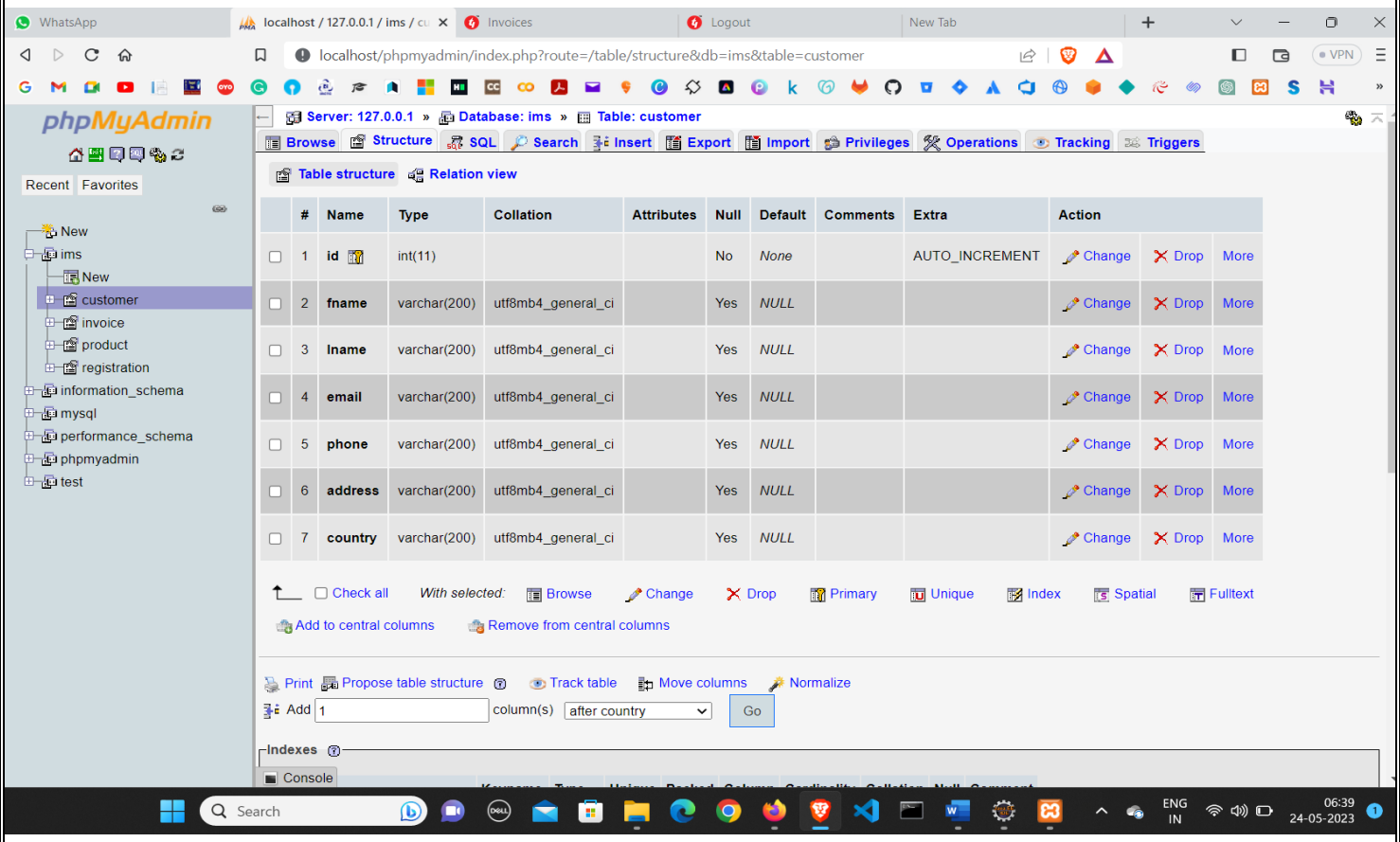
- **Contact Us**


WhatsApp
localhost / 127.0.0.1 | phpMyAdmin 5
(51) Zara Zara x Let Me Down Slow
Home
localhost:8080/Invoice-Management-System/home.jsp
CONTACT US
Feel free to drop any feedback where we need to work more. :)
Name
Email
Message
Submit
Search
ENG
IN
05:56
24-05-2023

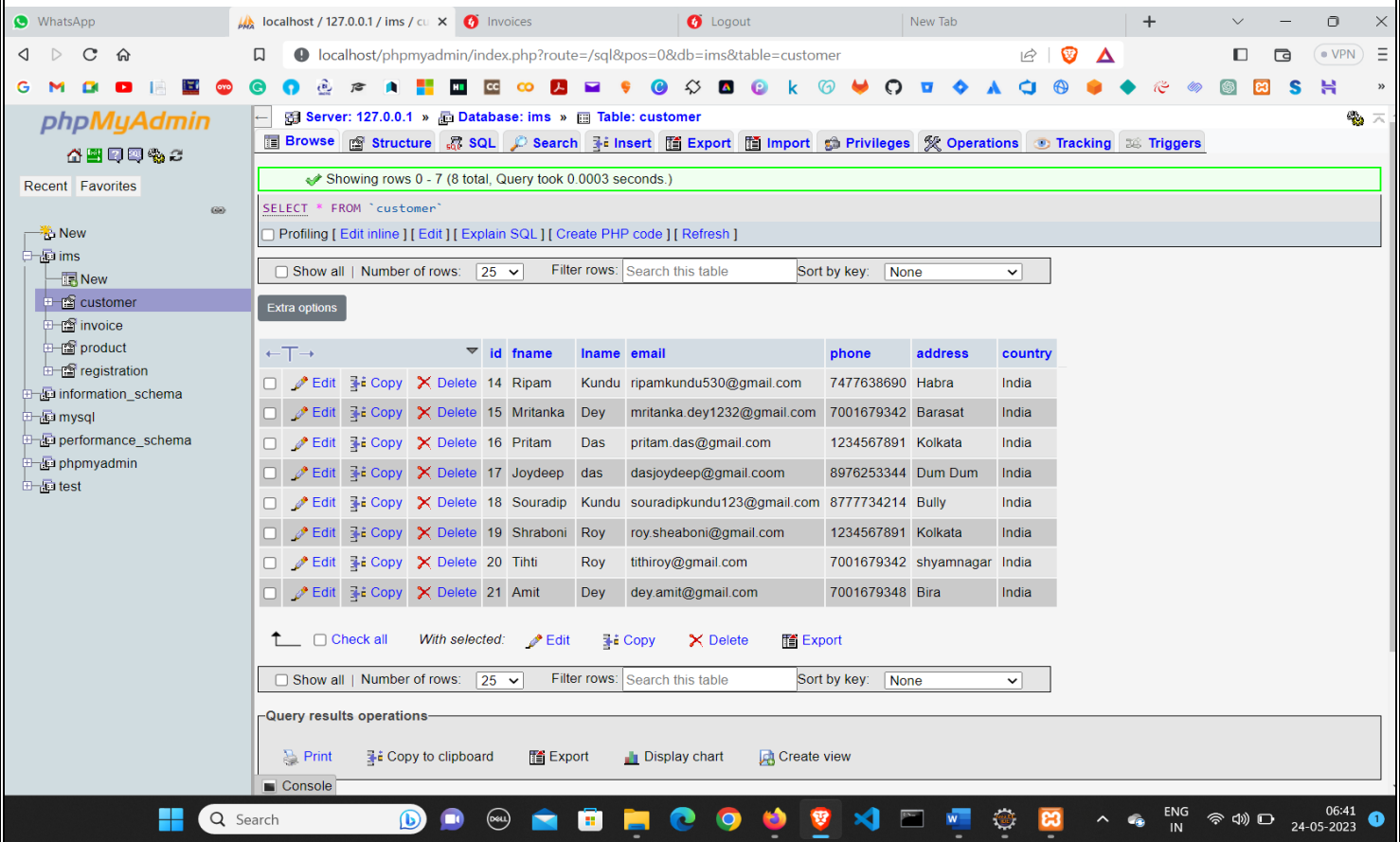
➢ Database Layout



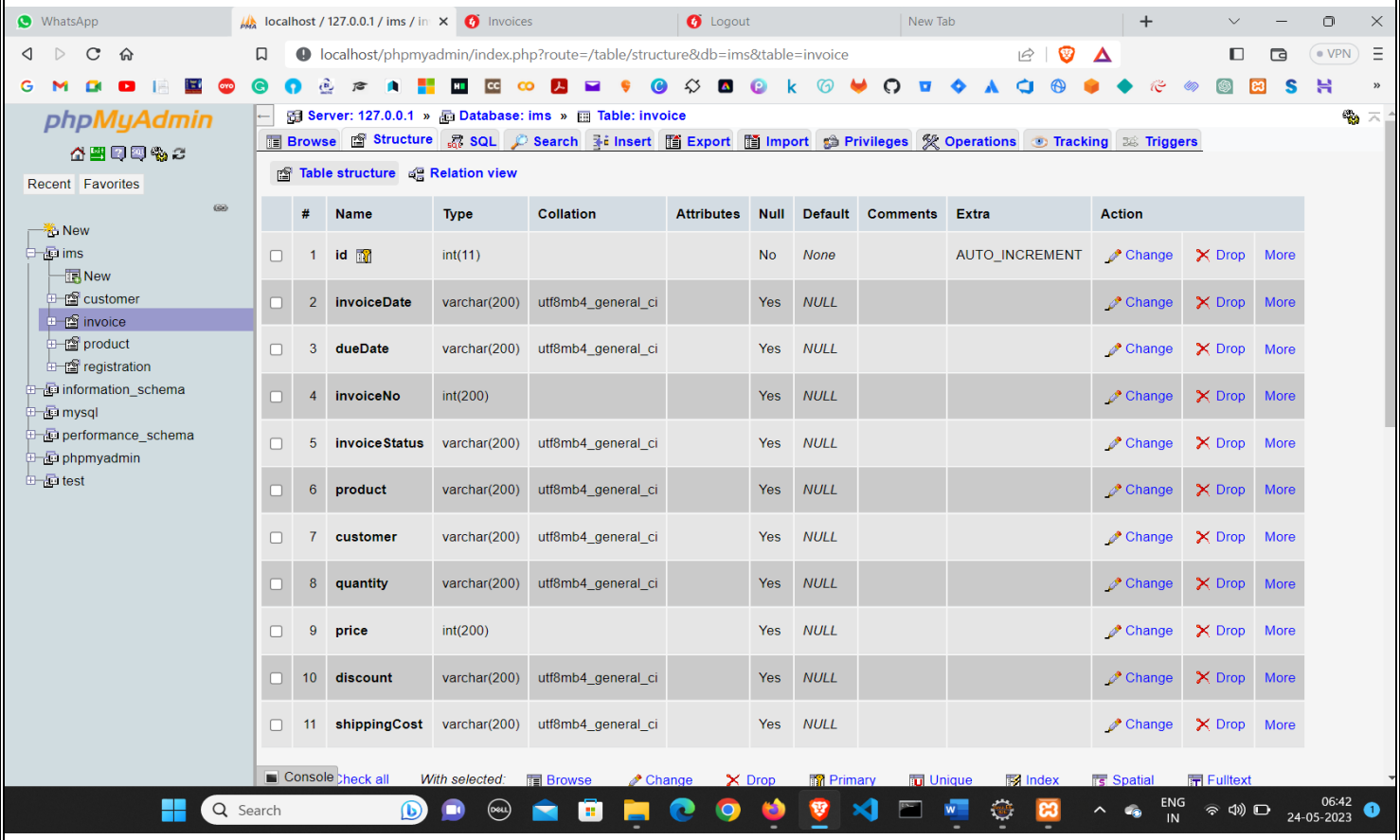
- **Customer Page**



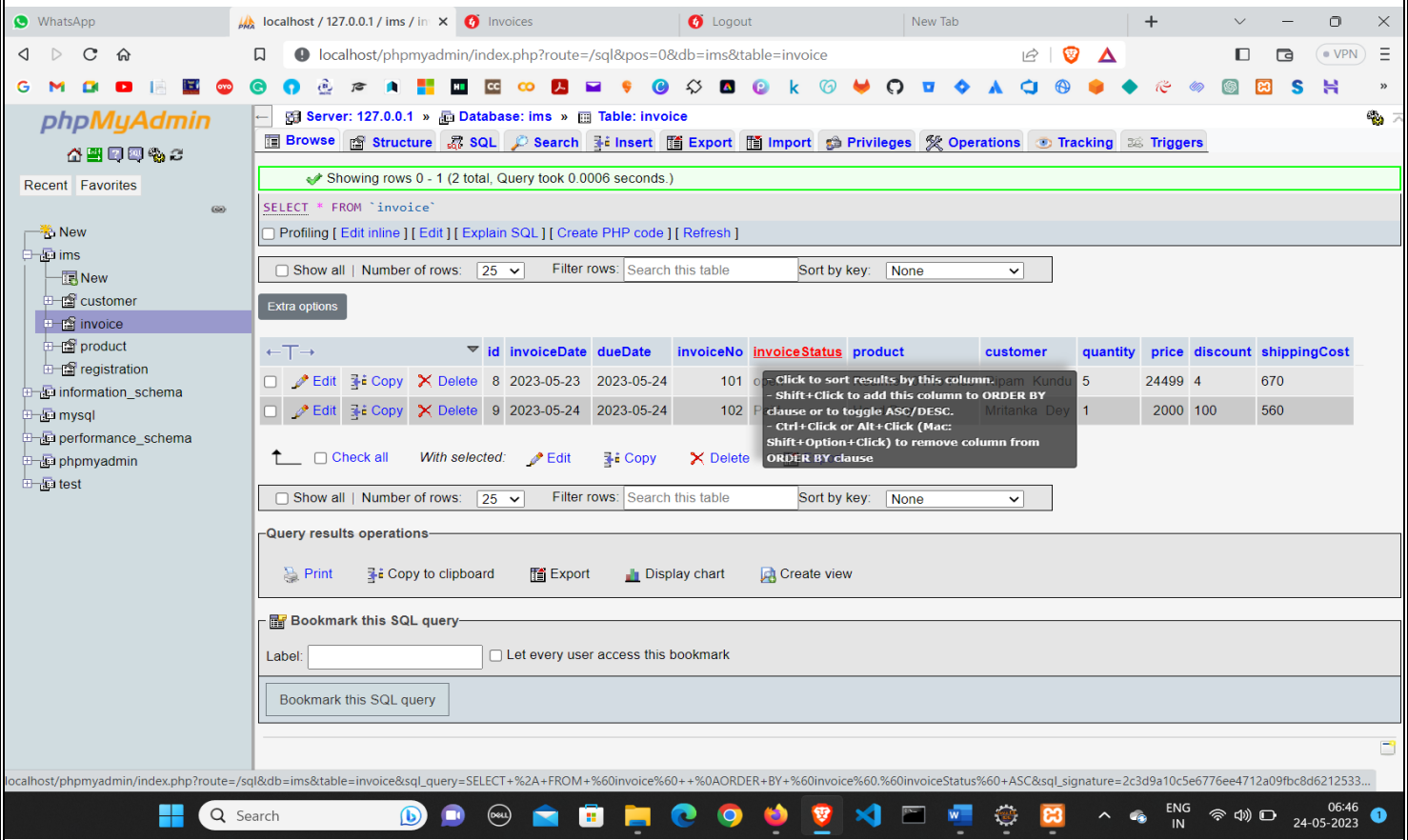
- **Customer User**



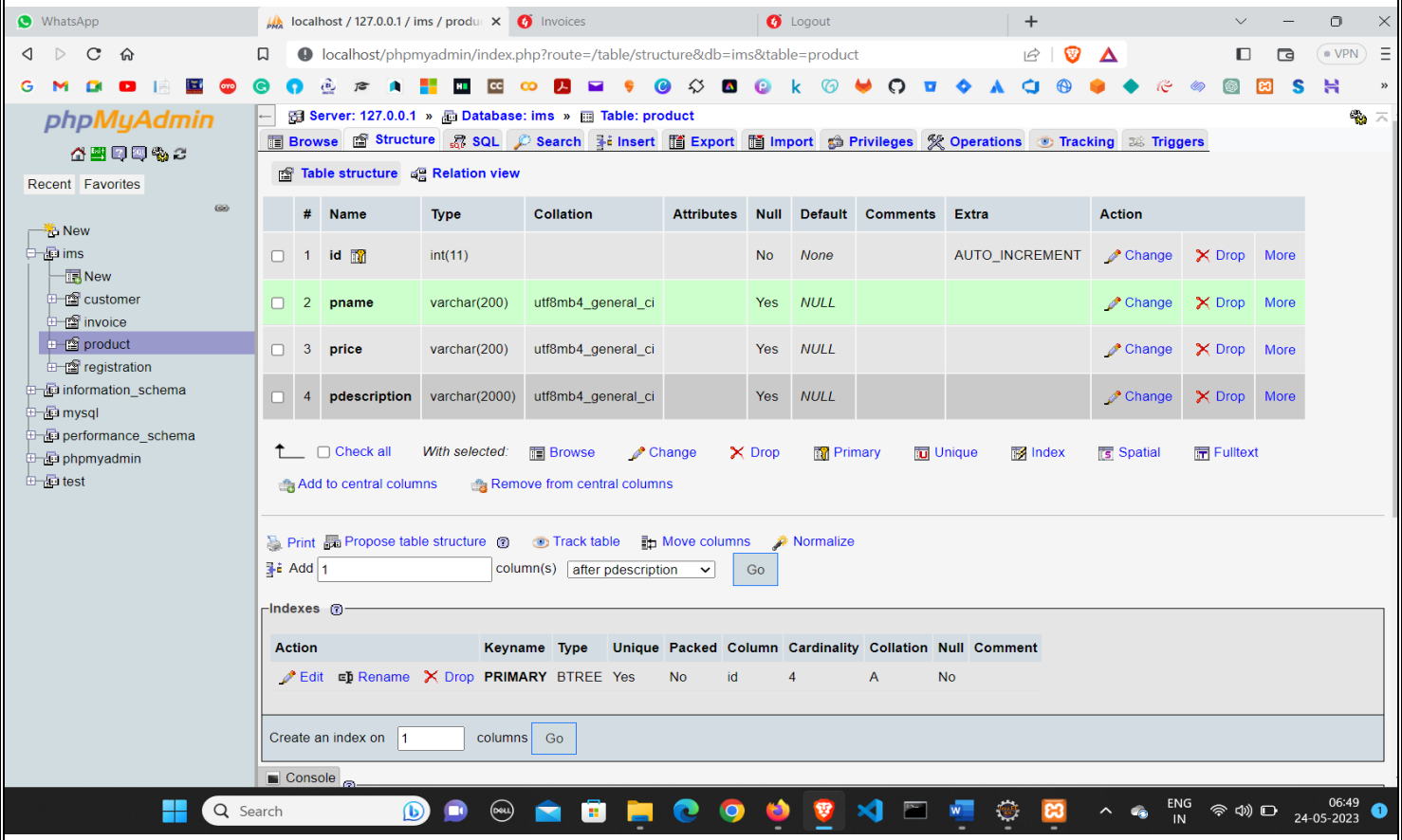
- **Invoice**



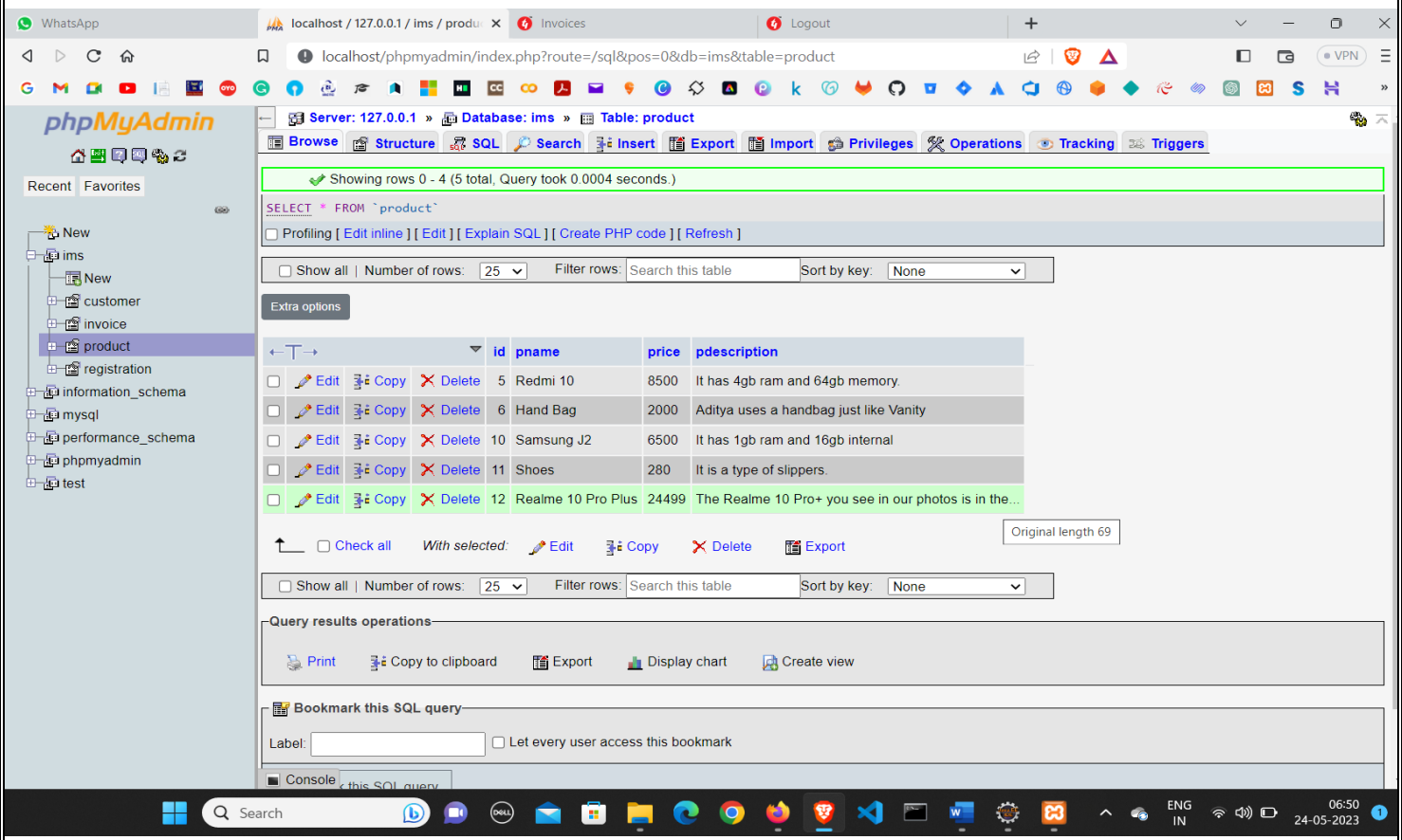
- **Invoice Table**



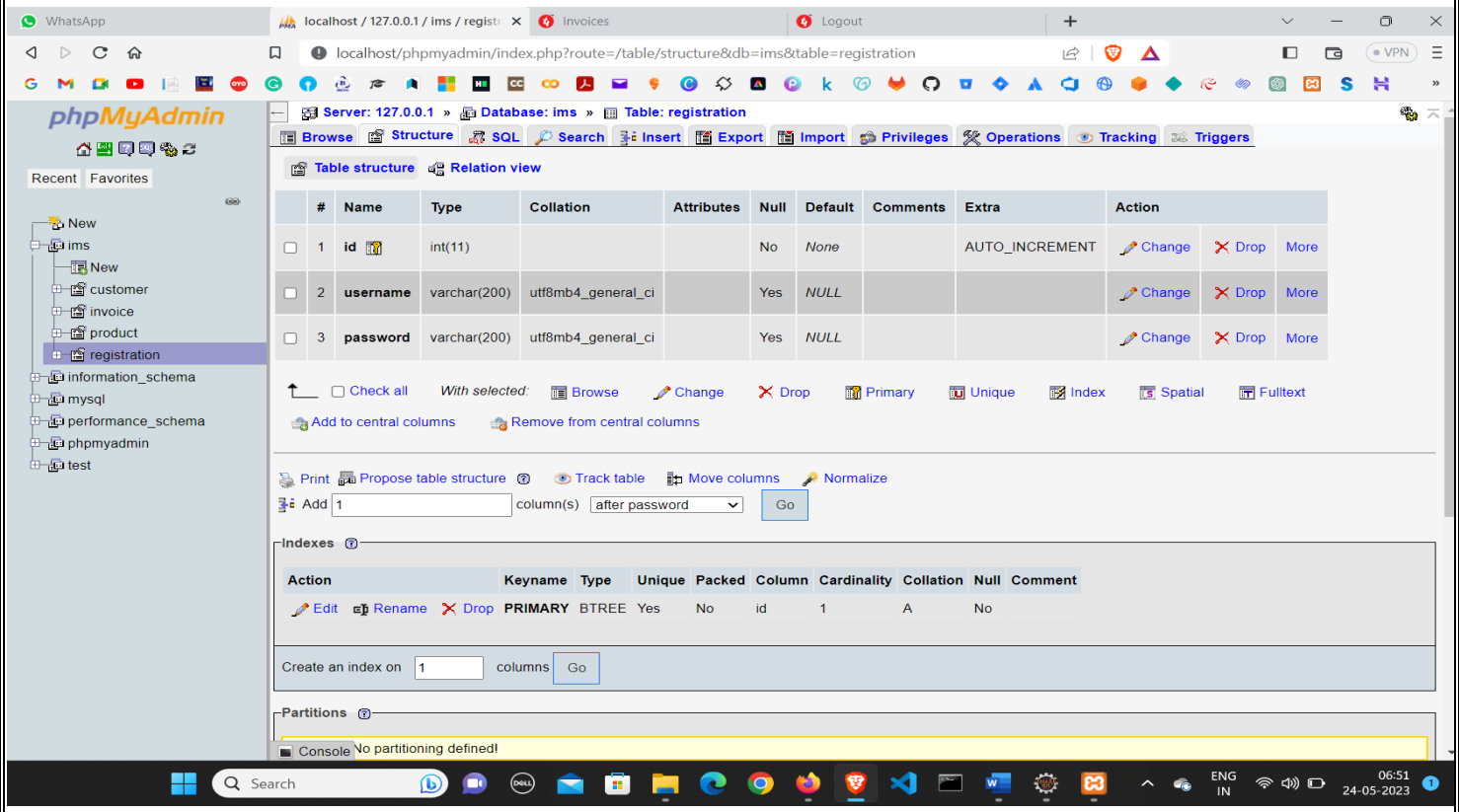
- **Product Structure**



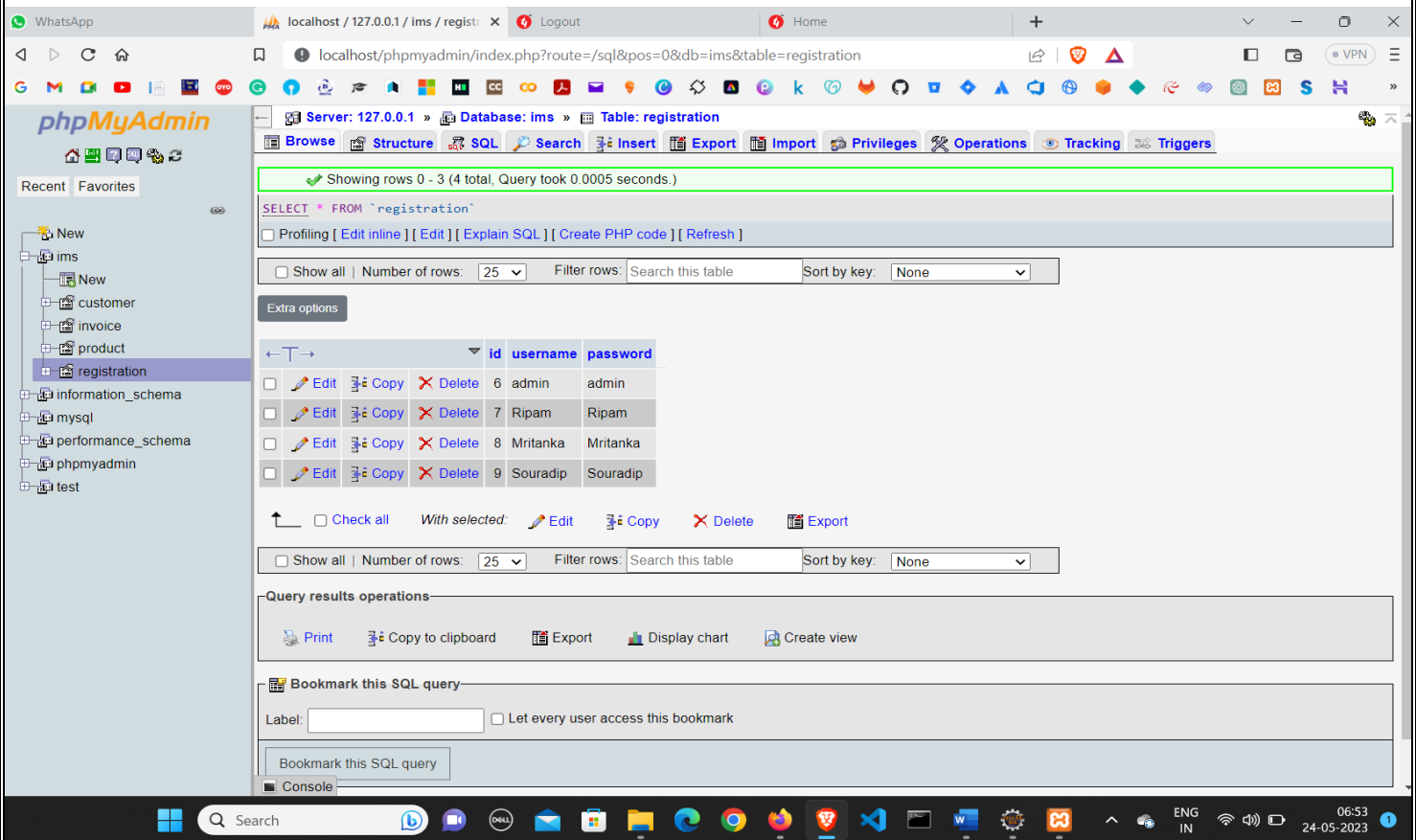
- **Product Table**



- **Registration Structure**



- **Registration User**



- **User Interface**
  - **Sign In**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<!--<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />-->
<title>Registration Form</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">

<link rel="stylesheet" href="./css/login.css">

</head>
<body>
  <div class="container">
          <div class="form-box">
                  <div class="form-value">
                          <form action="registrationServlet" method="POST">
                                  <h2>Registration</h2>
                                  <div class="inputbox">
                                          <ion-icon name="person-circle-outline"></ion-icon>
                                          <input type="text" name="username" required> <label
for="username">Username</label>
                                  </div>
                                  <div class="inputbox">
                                          <ion-icon name="lock-closed-outline"></ion-icon>
                                          <input type="password" name="password" required> <label
for="password">Password</label>
                                  </div>
                                  <button>Register</button>

                                  <div class="bottom">
                                          <div class="right">
                                                  <p>You Have Already Account.?</p>
                                                  <label><a href="login.jsp">Log In</a></label>

                                          </div>
                                  </div>
                          </form>
                  </div>
          </div>
  </div>
<script type="module"
  src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
<script nomodule=""
  src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
</body>
</html>
```

- **Log In**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<!-- - <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>-->
<title>Login Form</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">

<link rel="stylesheet" href="./css/login.css">
</head>
<body>
  <div class="container">
        <div class="form-box">
            <div class="form-value">
                <form action="Login" method="POST">
                    <h2>INVOICE SYSTEM</h2>
                    <h3>Login</h3>
                    <div class="inputbox">
                        <ion-icon name="person-circle-outline"></ion-icon>
                        <input type="text" name="username" required> <label
                            for="username">Username</label>
                    </div>
                    <div class="inputbox">
                        <ion-icon name="lock-closed-outline"></ion-icon>
                        <input type="password" name="password" required> <label
                            for="password">Password</label>
                    </div>
                    <button>Log in</button>
                    <div class="bottom">
                        <div class="left">
                            <input type="checkbox" id="check"> <label
for="check">Remeber
                                Me</label>
                        </div>
                        <div class="right">
                            <label><a href="forgetpassword.jsp">Forgot
Password?</a></label>
                        </div>
                        <div class="right">
                            <label><a href="registration.jsp">Sign In</a></label>
                        </div>
                    </div>
                </form>
            </div>
        </div>
  </div>
  <script type="module"
        src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
  <script nomodule=""
        src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
</body>
</html>
```

- **Logout**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Logout</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">

<link
  href="https://fonts.googleapis.com/css?family=Encode+Sans+Semi+Condensed:100,200,300,400"
  rel="stylesheet">
<link rel="stylesheet" href="./css/error.css">
</head>
<body class="loading">
<% session.invalidate(); %>
  <h2>You have been successfully logout In your Account.!</h2>
  <h3>Again Login Your Account <a href="./login.jsp" target="_blank">Log In </a></h3>
  <br>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="js/main.js" type="text/javascript"></script>

</body>
</html>
```

- **Forgot Password**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
	pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Forget Password</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">
<!-- Font Awesome Icons  -->
<link rel="stylesheet"
	href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css"
	integrity="sha512-
+4zCK9k+qNFUR5X+cKL9EIR+ZOhtIloNl9GIKS57V1MyNsYpYcUrUeQc9vNfzsWfV28IaLL3i96P9sdNyeRssA=="
	crossorigin="anonymous" />

<!-- Google Fonts  -->
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
	href="https://fonts.googleapis.com/css2?family=Poppins&display=swap"
	rel="stylesheet">
<link rel="stylesheet" href="./css/forget.css">
<style>
@media ( max-width :365px) {
	h2 {
		font-size: 1.2rem !important;
		text-align: center;
	}
	p {
		font-size: 0.8rem;
	}
}
</style>
</head>
<body>
	<form class="card" action="PasswordUpdate" method="POST">

		<p class="lock-icon">
			<i class="fas fa-lock"> </i>
		</p>
		<h2>Forgot Password?</h2>
		<br>
		<p>You can reset your Password here</p>
		<br> <input type="text" class="passInput" name="username"
			placeholder="Enter Username"> <br> <input type="text"
			name="password" class="passInput" placeholder="Enter New Password">
		<br>
		<button type="submit" name="submit">Reset</button>
	</form>
</body>
</html>
```

- **Home Page**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Home</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">

<link rel="stylesheet"href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" type="text/css" href="./css/home.css">
<link rel="stylesheet" type="text/css" href="./css/responsive.css">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ"
    crossorigin="anonymous">
<link href="/dist/output.css" rel="stylesheet">
<link rel="stylesheet"
    href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap">
<!-- Font Awesome CDN -->
<link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.15.4/css/all.css">
<link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />

<%
String username = (String) session.getAttribute("username");
if(username==null)
{
    request.getRequestDispatcher("login.jsp").forward(request, response);
}
%>
</head>
<body>
    <!-- ------------------------------Header Section---------------------------- -->
    <header>
        <nav>
            <div class="logo">
                <img src="./image/logo.png" alt="No-Pic">
            </div>
            <div class="menu">
                <a href="./home.jsp">Home</a>
                <a href="./invoice.jsp">Invoice</a>
                <a href="./product.jsp">Products</a>
                <a href="./customer.jsp">Customers</a>
                <a href="./calculator.jsp" target="_blank">Calculate It</a>
                <a href="./registration.jsp" target="_blank">Sign In</a>
                <a href="./login.jsp" target="_blank">Log In</a>
                <a href="./logout.jsp" target="_blank">Logout</a>

            </div>
            <!-- <div class="login">
        <a href="#">Login</a>
    </div> -->
```

```
            </nav>
            <section class="header_text">
                <span class="something">Invoice Management System</span>
                <h1>
                    Wait Wait! Where are you going take your invoice It's your <br>
                    <span class="auto-type"></span>
                </h1>
                <br>
            </section>

    </header>
    <!-- ------------------------------About Section---------------------------- -->
    <main data-aos="fade-up">
        <div class="container" style="margin-top: 4%;">
            <h1 style="margin-top: 3%;">
                ABOUT <span>US</span>
            </h1>
            <div class="row">
                <div class="col-md-6">
                    <img src="./image/about.png" alt="No-Pic">
                </div>
                <div class="col-md-6">
                    <p class="para">While similar information is included in sales
                        receipts and invoices, they are not the same. An invoice is issued
                        to collect payments from customers, and a sales receipt documents
                        proof of payment that a customer has made to a seller. Receipts
                        are used as documentation to confirm that a customer has received
                        the goods or services they paid for, and as a record that the
                        business has been paid.A payment letter is a polite way of
                        reminding your customer of their pending bill. Because of busy
                        schedules, some people forget to service their bills in time. In
                        such a case, the individual can write a letter to remind them they
                        are yet to pay the bill. A landlord, business owner, or learning
                        institution can write a letter to ask for payment.</p>
                </div>
            </div>
        </div>
    </main>
    <!-- ------------------------------Statistics Section---------------------------- -->
    <section style="margin-top: 3%;" data-aos="fade-up">
        <div class="hero">
            <div class="title">
                <h1 style="margin-top: 3%;">STATISTICS</h1>
                <!-- <p>Duis vulputate et nulla ac dapibus. Nullam feugiat massa elit, at scelerisque
urna facilisis id. Suspendisse commodo scelerisque sem sit amet aliquam. Curabitur nulla lectus, pretium ac arcu
sed, laoreet eleifend nunc.Duis vulputate et nulla ac dapibus. Nullam feugiat massa elit.</p> -->
            </div>

            <div class="row">
                <div class="col">
                    <div class="counter-box">
                        <i class="fa fa-inr" aria-hidden="true"></i>
                        <h2 class="counter" data-number="215">0+</h2>
                        <h4>SALES AMOUNT</h4>
                    </div>
                </div>
                <div class="col">
```

```
                                    <div class="counter-box">
                                        <i class="fa fa-users"></i>
                                        <h2 class="counter" data-number="312">0+</h2>
                                        <h4>USERS</h4>
                                    </div>
                                </div>
                                <div class="col">
                                    <div class="counter-box">
                                        <i class="fa fa-exclamation-triangle" aria-hidden="true"></i>
                                        <h2 class="counter" data-number="205">0+</h2>
                                        <h4>DUE AMOUNT</h4>
                                    </div>
                                </div>
                                <div class="col">
                                    <div class="counter-box">
                                        <i class="fa fa-file-text" aria-hidden="true"></i>
                                        <h2 class="counter" data-number="519">0+</h2>
                                        <h4>TOTAL INVOICE</h4>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </section>
                    <!-- ------------------------------Contact Form---------------------------- -->
                    <section class="text-gray-600 body-font relative">
                        <div class="container px-5 py-24 mx-auto">
                            <div class="flex flex-col text-center w-full mb-12">
                                <h1
                                    class="sm:text-3xl text-2xl font-medium title-font mb-4 text-gray-
900">CONTACT
                                    US</h1>
                                <p class="lg:w-2/3 mx-auto leading-relaxed text-base">Feel free
                                    to drop any feedback where we need to work more. :)</p>
                            </div>
                            <div class="lg:w-1/2 md:w-2/3 mx-auto">
                                <div class="flex flex-wrap -m-2">
                                    <div class="p-2 w-1/2">
                                        <div class="relative">
                                            <label for="name" class="leading-7 text-sm text-gray-
600">Name</label>
                                            <input type="text" id="name" name="name"
                                                class="w-full bg-gray-100 bg-opacity-50 rounded
border border-gray-300 focus:border-blue-500 focus:bg-white focus:ring-2 focus:ring-blue-200 text-base outline-
none text-gray-700 py-1 px-3 leading-8 transition-colors duration-200 ease-in-out">
                                        </div>
                                    </div>
                                    <div class="p-2 w-1/2">
                                        <div class="relative">
                                            <label for="email" class="leading-7 text-sm text-gray-
600">Email</label>
                                            <input type="email" id="email" name="email"
                                                class="w-full bg-gray-100 bg-opacity-50 rounded
border border-gray-300 focus:border-blue-500 focus:bg-white focus:ring-2 focus:ring-blue-200 text-base outline-
none text-gray-700 py-1 px-3 leading-8 transition-colors duration-200 ease-in-out">
                                        </div>
                                    </div>
                                    <div class="p-2 w-full">
```

```
                                        <div class="relative">
                                            <label for="message" class="leading-7 text-sm text-gray-600">Message</label>
                                            <textarea id="message" name="message"
                                                class="w-full bg-gray-100 bg-opacity-50 rounded border border-gray-300 focus:border-blue-500 focus:bg-white focus:ring-2 focus:ring-blue-200 h-32 text-base outline-none text-gray-700 py-1 px-3 resize-none leading-6 transition-colors duration-200 ease-in-out"></textarea>
                                        </div>
                                    </div>
                                    <div class="p-2 w-full">
                                        <button
                                            class="flex mx-auto text-white bg-blue-500 border-0 py-2 px-8 focus:outline-none hover:bg-blue-600 rounded text-lg"
                                            style="font-family: 'Bree Serif', sans-serif;">Submit</button>
                                    </div>

                                </div>
                            </div>
                        </div>
                    </section>
                    <!-- ----------------------------Footer Section---------------------------- -->
                    <section>
                        <footer class="footer" style="margin-top: 5%;">
                            <div class="waves">
                                <div class="wave" id="wave1"></div>
                                <div class="wave" id="wave2"></div>
                                <div class="wave" id="wave3"></div>
                                <div class="wave" id="wave4"></div>
                            </div>
                            <ul class="social-icon">
                                <li class="social-icon__item"><a class="social-icon__link"
                                        href="https://www.facebook.com/rk.kundu26/"> <ion-icon name="logo-facebook"></ion-icon>
                                </a></li>
                                <li class="social-icon__item"><a class="social-icon__link"

                                    href="https://twitter.com/RipamKundu?fbclid=IwAR025OdxOKMLPmm_Z6uHQeimXUKbj9s9elwNoj7wFwpQvPCQHRAFhwo3hEQ"> <ion-icon name="logo-twitter"></ion-icon>
                                </a></li>
                                <li class="social-icon__item"><a class="social-icon__link"
                                        href="https://www.linkedin.com/in/rk-ripam-kundu/"> <ion-icon name="logo-linkedin"></ion-icon>
                                </a></li>
                                <li class="social-icon__item"><a class="social-icon__link"
                                        href="https://www.instagram.com/rk_ripam_kundu/?hl=en"> <ion-icon name="logo-instagram"></ion-icon>
                                </a></li>
                            </ul>
                            <p>&copy; MadeBy MCA Student 2k22 - 23 | All Rights Reserved</p>
                        </footer>
                    </section>

                    <script src="https://unpkg.com/typed.js@2.0.15/dist/typed.umd.js"></script>
                    <script
                        src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"
```

```
            integrity="sha384-
ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe"
            crossorigin="anonymous"></script>
    <script src="https://cdn.tailwindcss.com"></script>
    <script src="https://unpkg.com/aos@next/dist/aos.js"></script>
    <script nomodule
            src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
    <script type="module"
            src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
    <script src="./js/custom.js"></script>
</body>
</html>
```

- **Invoice**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Invoices</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">

<link rel="stylesheet" href="./css/invoice.css">
<link rel="stylesheet" href="./css/responsive.css">
<link rel="stylesheet" href="./css/responsive1.css">
<link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css">
<link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />
<%
String username = (String) session.getAttribute("username");
if(username==null)
{
    request.getRequestDispatcher("login.jsp").forward(request, response);
}
%>
</head>
<body>
    <header>
        <nav>
            <div class="logo">
                <img src="./image/logo.png" alt="No-Pic">
            </div>
            <div class="menu">
                <a href="./home.jsp">Home</a> <a href="./invoice.jsp">Invoice</a>
                <a href="./product.jsp">Products</a> <a
href="./customer.jsp">Customers</a>
                <a href="./calculator.jsp" target="_blank">Calculate It</a>
                <a href="./logout.jsp" target="_blank">Logout</a>

            </div>
            <!-- <div class="login">
          <a href="#">Login</a>
        </div> -->
        </nav>
        <section class="header_text">
            <span class="something">Invoice Management System</span>
            <h3>INVOICES MANAGING SECTION</h3>
            <br>
        </section>
    </header>
    <section>
        <div class="container" style="margin-top: 3%;">
```

```
                    <h1>MANAGING INVOICES</h1>
                    <div class="row">
                        <div class="col-md-6">

                            <div class="pic">
                                <!-- <img src="./image/invoice2.jpg" alt="No-Pic"> -->

                            </div>
                        </div>
                        <div class="col-md-6">
                            <h3 style="text-decoration: underline;">CREATE INVOICE</h3>

                            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam
                                sunt dolor dolores facere aliquid sint pariatur quis nesciunt,
                                incidunt commodi, itaque molestiae inventore quia?
Animi?</p>
                            <!-- <button href = "./invoiceform.html"  type="button" class="kuch-
bhi" > Click</button> -->
                            <a href="chooseinvoice" target="_blank" class="kuch-bhi">
                                Click</a>

                        </div>
                    </div>
                    <div class="row" style="margin-top: 5%;">
                        <div class="col-md-6">
                            <h3 style="text-decoration: underline;">MANAGE INVOICE</h3>
                            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam
                                sunt dolor dolores facere aliquid sint pariatur quis nesciunt,
                                incidunt commodi, itaque molestiae inventore quia?
Animi?</p>
                            <!-- <button href = "./manageinvoice.html"  type="button"
class="kuch" > Click</button> -->
                            <a href="showinvoicedata" target="_blank" class="kuch">
                                Click</a>
                        </div>
                        <div class="col-md-6">
                            <!-- <img src="./image/invoice3.jpg" alt="No-Pic"> -->
                            <div class="pic1">
                                <!-- <img src="./image/invoice2.jpg" alt="No-Pic"> -->
                            </div>
                        </div>
                    </div>
                </div>
            </section>
            <!-- ----------------------------------------------Footer ---------------------------------- -->
            <section>
                <footer class="footer" style="margin-top: 15%;">
                    <div class="waves">
                        <div class="wave" id="wave1"></div>
                        <div class="wave" id="wave2"></div>
                        <div class="wave" id="wave3"></div>
                        <div class="wave" id="wave4"></div>
                    </div>
```

```
                    <ul class="social-icon">
                        <li class="social-icon__item"><a class="social-icon__link"
                                href="#"> <ion-icon name="logo-facebook"></ion-icon>
                        </a></li>
                        <li class="social-icon__item"><a class="social-icon__link"
                                href="#"> <ion-icon name="logo-twitter"></ion-icon>
                        </a></li>
                        <li class="social-icon__item"><a class="social-icon__link"
                                href="#"> <ion-icon name="logo-linkedin"></ion-icon>
                        </a></li>
                        <li class="social-icon__item"><a class="social-icon__link"
                                href="#"> <ion-icon name="logo-instagram"></ion-icon>
                        </a></li>
                    </ul>
                    <p>&copy; MadeByRenegades | All Rights Reserved</p>
                </footer>
            </section>



            <script
                src="shttps://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js"></script>

            <script type="module"
                src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
            <script nomodule
                src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
            <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
            <script src="https://unpkg.com/aos@next/dist/aos.js"></script>
</body>
</html>
```

- **Invoice Form**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Invoice Form</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">
<style>
@import
    url('https://fonts.googleapis.com/css2?family=Bree+Serif&display=swap')
    ;

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Bree Serif', serif;
}

body {
    height: auto;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 10px;
    background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

.container {
    max-width: 700px;
    width: 100%;
    background-color: #fff;
    padding: 25px 30px;
    border-radius: 5px;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.15);
}

.container .title {
    font-size: 25px;
    font-weight: 500;
    position: relative;
}

.container .title::before {
    content: "";
    position: absolute;
    left: 0;
```

```
        bottom: 0;
        height: 3px;
        width: 30px;
        border-radius: 5px;
        background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

.content form .user-details {
        display: flex;
        flex-wrap: wrap;
        justify-content: space-between;
        margin: 20px 0 12px 0;
}

form .user-details .input-box {
        margin-bottom: 15px;
        width: calc(100%/ 2 - 20px);
}

form .input-box span.details {
        display: block;
        font-weight: 500;
        margin-bottom: 5px;
}

.user-details .input-box input, .user-details .input-box select {
        height: 45px;
        width: 100%;
        outline: none;
        font-size: 16px;
        border-radius: 5px;
        padding-left: 15px;
        border: 1px solid #ccc;
        border-bottom-width: 2px;
        transition: all 0.3s ease;
}

.user-details .input-box input:focus, .user-details .input-box input:valid
        {
        border-color: #9b59b6;
}

.user-details .input-box select:focus, .user-details .input-box select:valid
        {
        border-color: #9b59b6;
}

form .gender-details .gender-title {
        font-size: 20px;
        font-weight: 500;
}

form .category {
```

```
        display: flex;
        width: 80%;
        margin: 14px 0;
        justify-content: space-between;
}

form .category label {
        display: flex;
        align-items: center;
        cursor: pointer;
}

form .category label .dot {
        height: 18px;
        width: 18px;
        border-radius: 50%;
        margin-right: 10px;
        background: #d9d9d9;
        border: 5px solid transparent;
        transition: all 0.3s ease;
}

#dot-1:checked ~ .category label .one, #dot-2:checked ~ .category label .two,
        #dot-3:checked ~ .category label .three {
        background: #9b59b6;
        border-color: #d9d9d9;
}

form input[type="radio"] {
        display: none;
}

form .button {
        width: 100%;
        height: 45px;
        margin: 35px 0
}

form .button input {
        height: 100%;
        width: 100%;
        border-radius: 5px;
        border: none;
        color: #fff;
        font-size: 18px;
        font-weight: 500;
        letter-spacing: 1px;
        cursor: pointer;
        transition: all 0.3s ease;
        background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

form .button input:hover {
```

```
        /* transform: scale(0.99); */
        background: linear-gradient(-135deg, #71b7e6, #9b59b6);
}

@media ( max-width : 584px) {
   body{
      height: 100vh;
   }
        .container {
                max-width: 100%;
        }
        form .user-details .input-box {
                margin-bottom: 15px;
                width: 100%;
        }
        form .category {
                width: 100%;
        }
        .content form .user-details {
                max-height: 300px;
                overflow-y: scroll;
        }
        .user-details::-webkit-scrollbar {
                width: 5px;
        }
}

@media ( max-width : 459px) {
        .container .content .category {
                flex-direction: column;
        }
        .user-details .input-box input {
   font-size: 14px;
  }
}

.subtotal {
        display: flex;
        flex-direction: column;
}

.sub {
        display: flex;
}
</style>
</head>
<body>
        <div class="container">
                <div class="title">Fill the Invoice Details</div>
                <div class="content">
                        <form action="#">
                                <div class="user-details">
                                        <div class="input-box">
```

```
                    <span class="details">Invoice Date</span>
                    <input type="date" name="invoiceDate" placeholder="Enter
the invoice date" required>
                </div>
                <div class="input-box">
                    <span class="details">Due Date</span>
                    <input type="date" name="dueDate" placeholder="Enter the
due date" required>
                </div>
                <div class="input-box">
                    <span class="details">Invoice Number</span>
                    <input type="text" name="invoiceNo" placeholder="Enter the
invoice number" required>
                </div>
                <div class="input-box">
                    <span class="details">Invoice Status</span>
                    <select type="text" name="invoiceStatus" required>
                        <option value="open">Open</option>
                        <option value="paid">Paid</option>
                    </select>
                </div>
                <div class="input-box">
                    <span class="details">Select a Product</span>
                    <select type="text" name="product" required>
                        <option value=""></option>
                    </select>
                </div>
                <div class="input-box">
                    <span class="details">Select the customer</span> <select
name="customer"
                        type="text" required>
                        <option value=""></option>
                    </select>
                </div>
                <div class="input-box">
                    <span class="details">Quantity</span>
                    <input type="number" name="quantity" placeholder="Enter
the quantity" required>
                </div>
                <div class="input-box">
                    <span class="details">Price</span> <input type="number"
name="price"
                        placeholder="Price of the product" required>
                </div>
                <div class="input-box">
                    <span class="details">Discount</span> <input type="number"
name="discount"
                        placeholder="Enter the discount percentage" required>
                </div>
                <div class="input-box">
                    <span class="details">Shipping Cost</span> <input
type="number" name="shippingCost"
                        placeholder="Enter the shipping cost" required>
```

```
                                    </div>
                                    <div class="subtotal" style="margin-top: 1%;">
                                        <div class="sub">
                                            <h4>TAX & VAT :</h4>
                                            <p>0 Rupees</p>
                                        </div>
                                        <div class="sub">
                                            <h4>SUB TOTAL :</h4>
                                            <p>0 Rupees</p>
                                        </div>
                                    </div>
                                </div>
                                <div class="button">
                                    <input type="submit" value="Create Invoice" name="submit">
                                </div>
                            </form>
                        </div>
                    </div>
                </body>
                </html>
```

- **Product**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Products</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">

<link rel="stylesheet" href="./css/product.css">
<link rel="stylesheet" href="./css/responsive.css">
<link rel="stylesheet" href="./css/responsive1.css">
<link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css">
<link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />
<%
String username = (String) session.getAttribute("username");
if(username==null)
{
    request.getRequestDispatcher("login.jsp").forward(request, response);
}
%>
</head>
<body>
    <header>
        <nav>
            <div class="logo">
                <img src="./image/logo.png" alt="No-Pic">
            </div>
            <div class="menu">
                <a href="./home.jsp">Home</a> <a href="./invoice.jsp">Invoice</a>
                <a href="./product.jsp">Products</a> <a
href="./customer.jsp">Customers</a>
                <a href="./calculator.jsp" target="_blank">Calculate It</a>
                <a href="./logout.jsp" target="_blank">Logout</a>

            </div>
            <!-- <div class="login">
      <a href="#">Login</a>
    </div> -->
        </nav>
        <section class="header_text">
            <span class="something">Invoice Management System</span>
            <h3>PRODUCT SECTION</h3>
            <br>
        </section>
    </header>
    <section>
```

```
            <div class="container" style="margin-top: 3%;">
                <h1>PRODUCT ZONE</h1>
                <div class="row">
                    <div class="col-md-6">
                        <!-- <img src="./image/invoice2.jpg" alt="No-Pic"> -->
                        <div class="pic"></div>
                    </div>
                    <div class="col-md-6">
                        <h3 style="text-decoration: underline;">CREATE PRODUCT</h3>

                        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam
                            sunt dolor dolores facere aliquid sint pariatur quis nesciunt,
                            incidunt commodi, itaque molestiae inventore quia?
Animi?</p>
                        <!-- <button href = "# " type="button" class="kuch-bhi">
Click</button> -->
                        <a href="./productform.jsp" target="_blank" class="kuch-bhi">
                            Click</a>

                    </div>
                </div>
                <div class="row" style="margin-top: 5%;">
                    <div class="col-md-6">
                        <h3 style="text-decoration: underline;">MANAGE PRODUCT</h3>

                        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam
                            sunt dolor dolores facere aliquid sint pariatur quis nesciunt,
                            incidunt commodi, itaque molestiae inventore quia?
Animi?</p>
                        <!-- <button href = "# " type="button" class="kuch"> Click</button> -
->
                        <a href="showproductdata" target="_blank" class="kuch">
                            Click</a>

                    </div>
                    <div class="col-md-6">
                        <!-- <img src="./image/invoice3.jpg" alt="No-Pic"> -->
                        <div class="pic1"></div>
                    </div>
                </div>
            </div>
        </section>
        <!-- ----------------------------------------------Footer ------------------------------------ -->
        <section>
            <footer class="footer" style="margin-top: 15%;">
                <div class="waves">
                    <div class="wave" id="wave1"></div>
                    <div class="wave" id="wave2"></div>
                    <div class="wave" id="wave3"></div>
                    <div class="wave" id="wave4"></div>
                </div>
                <ul class="social-icon">
                    <li class="social-icon__item"><a class="social-icon__link"
```

```
                        href="#"> <ion-icon name="logo-facebook"></ion-icon>
                    </a></li>
                    <li class="social-icon__item"><a class="social-icon__link"
                        href="#"> <ion-icon name="logo-twitter"></ion-icon>
                    </a></li>
                    <li class="social-icon__item"><a class="social-icon__link"
                        href="#"> <ion-icon name="logo-linkedin"></ion-icon>
                    </a></li>
                    <li class="social-icon__item"><a class="social-icon__link"
                        href="#"> <ion-icon name="logo-instagram"></ion-icon>
                    </a></li>
                </ul>
                <p>&copy;MadeByRenegades | All Rights Reserved</p>
            </footer>
        </section>


        <script type="module"
            src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
        <script nomodule
            src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
</body>
</html>
```

- **Product From**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Product Form</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">

<style>
@import
    url('https://fonts.googleapis.com/css2?family=Bree+Serif&display=swap')
    ;

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Bree Serif', serif;
}

body {
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 10px;
    background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

.container {
    max-width: 700px;
    width: 100%;
    background-color: #fff;
    padding: 25px 30px;
    border-radius: 5px;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.15);
}

.container .title {
    font-size: 25px;
    font-weight: 500;
    position: relative;
}

.container .title::before {
    content: "";
    position: absolute;
    left: 0;
```

```
        bottom: 0;
        height: 3px;
        width: 30px;
        border-radius: 5px;
        background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

.content form .user-details {
        display: flex;
        flex-wrap: wrap;
        justify-content: space-between;
        margin: 20px 0 12px 0;
}

form .user-details .input-box {
        margin-bottom: 15px;
        width: calc(100%/ 2 - 20px);
}

form .input-box span.details {
        display: block;
        font-weight: 500;
        margin-bottom: 5px;
}

.user-details .input-box input {
        height: 45px;
        width: 100%;
        outline: none;
        font-size: 16px;
        border-radius: 5px;
        padding-left: 15px;
        border: 1px solid #ccc;
        border-bottom-width: 2px;
        transition: all 0.3s ease;
}

.user-details .input-box input:focus, .user-details .input-box input:valid
        {
        border-color: #9b59b6;
}

form .gender-details .gender-title {
        font-size: 20px;
        font-weight: 500;
}

form .category {
        display: flex;
        width: 80%;
        margin: 14px 0;
        justify-content: space-between;
}
```

```
form .category label {
        display: flex;
        align-items: center;
        cursor: pointer;
}

form .category label .dot {
        height: 18px;
        width: 18px;
        border-radius: 50%;
        margin-right: 10px;
        background: #d9d9d9;
        border: 5px solid transparent;
        transition: all 0.3s ease;
}

#dot-1:checked ~ .category label .one, #dot-2:checked ~ .category label .two,
        #dot-3:checked ~ .category label .three {
        background: #9b59b6;
        border-color: #d9d9d9;
}

form input[type="radio"] {
        display: none;
}

form .button {
        height: 45px;
        margin: 35px 0
}

form .button {
        width: 100%;
}

form .button input {
        height: 100%;
        width: 100%;
        border-radius: 5px;
        border: none;
        color: #fff;
        font-size: 18px;
        font-weight: 500;
        letter-spacing: 1px;
        cursor: pointer;
        transition: all 0.3s ease;
        background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

form .button input:hover {
        /* transform: scale(0.99); */
        background: linear-gradient(-135deg, #71b7e6, #9b59b6);
```

```
}

@media ( max-width : 584px) {
   body{
      height: 100vh;
   }
        .container {
                max-width: 100%;
        }
        @media ( max-width : 584px) {
        }
        form .user-details .input-box {
                margin-bottom: 15px;
                width: 100%;
        }
        form .category {
                width: 100%;
        }
        .content form .user-details {
                max-height: 300px;
                overflow-y: scroll;
        }
        .user-details::-webkit-scrollbar {
                width: 5px;
        }
}

@media ( max-width : 459px) {
        .container .content .category {
                flex-direction: column;
        }
        .user-details .input-box input {
   font-size: 14px;
  }
}

.input-box1 {
        display: flex;
        flex-direction: column;
        width: 100%;
}

.input-box1 .details {
        display: block;
        font-weight: 500;
        margin-bottom: 5px;
}

.input-box1 textarea {
        width: 100%;
        resize: none;
        outline: none;
        font-size: 16px;
```

```
        border-radius: 5px;
        padding-left: 15px;
        border: 1px solid #ccc;
        border-bottom-width: 2px;
        transition: all 0.3s ease;
}

.user-details .input-box1 textarea:focus, .user-details .input-box1 textarea:valid
        {
        border-color: #9b59b6;
}
</style>
</head>
<body>
        <div class="container">
                <div class="title">Fill the Product Details</div>
                <div class="content">
                        <form action="productInsert" method="POST">
                                <div class="user-details">
                                        <div class="input-box">
                                                <span class="details">Product Name</span> <input
type="text"
                                                        name="pname" placeholder="Enter your product"
required>
                                        </div>
                                        <div class="input-box">
                                                <span class="details">Price</span> <input type="number"
                                                        name="price" placeholder="Enter the price of the
product"
                                                        required>
                                        </div>
                                        <div class="input-box1">
                                                <span class="details">Product Description</span>
                                                <textarea id="" cols="30" rows="10" name="pdescription"
required></textarea>
                                        </div>
                                </div>

                                <div class="button">
                                        <input type="submit" value="Add Product" name="submit">
                                </div>
                        </form>
                </div>
        </div>
</body>
</html>
```

- # Customer

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Customer form</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">
<style>
@import
    url('https://fonts.googleapis.com/css2?family=Bree+Serif&display=swap')
    ;

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Bree Serif', serif;
}

body {
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 10px;
    background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

.container {
    max-width: 700px;
    width: 100%;
    background-color: #fff;
    padding: 25px 30px;
    border-radius: 5px;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.15);
}

.container .title {
    font-size: 25px;
    font-weight: 500;
    position: relative;
}

.container .title::before {
    content: "";
    position: absolute;
    left: 0;
    bottom: 0;
```

```
    height: 3px;
    width: 30px;
    border-radius: 5px;
    background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

.content form .user-details {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    margin: 20px 0 12px 0;
}

form .user-details .input-box {
    margin-bottom: 15px;
    width: calc(100%/ 2 - 20px);
}

form .input-box span.details {
    display: block;
    font-weight: 500;
    margin-bottom: 5px;
}

.user-details .input-box input {
    height: 45px;
    width: 100%;
    outline: none;
    font-size: 16px;
    border-radius: 5px;
    padding-left: 15px;
    border: 1px solid #ccc;
    border-bottom-width: 2px;
    transition: all 0.3s ease;
}

.user-details .input-box input:focus, .user-details .input-box input:valid
    {
    border-color: #9b59b6;
}

form .gender-details .gender-title {
    font-size: 20px;
    font-weight: 500;
}

form .category {
    display: flex;
    width: 80%;
    margin: 14px 0;
    justify-content: space-between;
}
```

```
form .category label {
    display: flex;
    align-items: center;
    cursor: pointer;
}

form .category label .dot {
    height: 18px;
    width: 18px;
    border-radius: 50%;
    margin-right: 10px;
    background: #d9d9d9;
    border: 5px solid transparent;
    transition: all 0.3s ease;
}

#dot-1:checked ~ .category label .one, #dot-2:checked ~ .category label .two,
    #dot-3:checked ~ .category label .three {
    background: #9b59b6;
    border-color: #d9d9d9;
}

form input[type="radio"] {
    display: none;
}

form .button {
    width: 100%;
    height: 45px;
    margin: 35px 0
}

form .button input {
    height: 100%;
    width: 100%;
    border-radius: 5px;
    border: none;
    color: #fff;
    font-size: 18px;
    font-weight: 500;
    letter-spacing: 1px;
    cursor: pointer;
    transition: all 0.3s ease;
    background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

form .button input:hover {
    /* transform: scale(0.99); */
    background: linear-gradient(-135deg, #71b7e6, #9b59b6);
}

@media ( max-width : 584px) {
    .container {
```

```
            max-width: 100%;
        }
        form .user-details .input-box {
            margin-bottom: 15px;
            width: 100%;
        }
        form .category {
            width: 100%;
        }
        .content form .user-details {
            max-height: 300px;
            overflow-y: scroll;
        }
        .user-details::-webkit-scrollbar {
            width: 5px;
        }
    }

    @media ( max-width : 459px) {
        .container .content .category {
            flex-direction: column;
        }
        .user-details .input-box input {
            font-size: 14px;
        }
    }
    </style>
    </head>
    <body>
        <div class="container">
            <div class="title">Fill the Customer Details</div>
            <div class="content">
                <form action="customerInsert" method="POST">
                    <div class="user-details">
                        <div class="input-box">
                            <span class="details">First Name</span> <input type="text"
                                name="fname" placeholder="Enter your first-name"
    required>
                        </div>
                        <div class="input-box">
                            <span class="details">Last Name</span> <input type="text"
                                name="lname" placeholder="Enter your last-name"
    required>
                        </div>
                        <div class="input-box">
                            <span class="details">Email</span> <input type="email"
                                name="email" placeholder="Enter your email"
    required>
                        </div>
                        <div class="input-box">
                            <span class="details">Phone Number</span> <input
    type="text"
```

```
                                        name="phone" placeholder="Enter your phone
number" required>
                                </div>
                                <div class="input-box">
                                        <span class="details">Address</span> <input type="text"
                                        name="address" placeholder="Enter your address"
required>
                                </div>
                                <div class="input-box">
                                        <span class="details">Country</span> <input type="text"
                                        name="country" placeholder="Enter your country"
required>
                                </div>
                        </div>
                        <div class="button">
                                <input type="submit" value="Create Customer" name="submit">
                        </div>
                </form>
        </div>
    </div>
</body>
</html>
```

- **Customer From**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Customer form</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">
<style>
@import
    url('https://fonts.googleapis.com/css2?family=Bree+Serif&display=swap');
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Bree Serif', serif;
}

body {
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 10px;
    background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

.container {
    max-width: 700px;
    width: 100%;
    background-color: #fff;
    padding: 25px 30px;
    border-radius: 5px;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.15);
}

.container .title {
    font-size: 25px;
    font-weight: 500;
    position: relative;
}

.container .title::before {
    content: "";
    position: absolute;
    left: 0;
    bottom: 0;
```

```
    height: 3px;
    width: 30px;
    border-radius: 5px;
    background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

.content form .user-details {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    margin: 20px 0 12px 0;
}

form .user-details .input-box {
    margin-bottom: 15px;
    width: calc(100%/ 2 - 20px);
}

form .input-box span.details {
    display: block;
    font-weight: 500;
    margin-bottom: 5px;
}

.user-details .input-box input {
    height: 45px;
    width: 100%;
    outline: none;
    font-size: 16px;
    border-radius: 5px;
    padding-left: 15px;
    border: 1px solid #ccc;
    border-bottom-width: 2px;
    transition: all 0.3s ease;
}

.user-details .input-box input:focus, .user-details .input-box input:valid
    {
    border-color: #9b59b6;
}

form .gender-details .gender-title {
    font-size: 20px;
    font-weight: 500;
}

form .category {
    display: flex;
    width: 80%;
    margin: 14px 0;
    justify-content: space-between;
}
```

```
form .category label {
    display: flex;
    align-items: center;
    cursor: pointer;
}

form .category label .dot {
    height: 18px;
    width: 18px;
    border-radius: 50%;
    margin-right: 10px;
    background: #d9d9d9;
    border: 5px solid transparent;
    transition: all 0.3s ease;
}

#dot-1:checked ~ .category label .one, #dot-2:checked ~ .category label .two,
    #dot-3:checked ~ .category label .three {
    background: #9b59b6;
    border-color: #d9d9d9;
}

form input[type="radio"] {
    display: none;
}

form .button {
    width: 100%;
    height: 45px;
    margin: 35px 0
}

form .button input {
    height: 100%;
    width: 100%;
    border-radius: 5px;
    border: none;
    color: #fff;
    font-size: 18px;
    font-weight: 500;
    letter-spacing: 1px;
    cursor: pointer;
    transition: all 0.3s ease;
    background: linear-gradient(135deg, #71b7e6, #9b59b6);
}

form .button input:hover {
    /* transform: scale(0.99); */
    background: linear-gradient(-135deg, #71b7e6, #9b59b6);
}

@media ( max-width : 584px) {
    .container {
```

```
            max-width: 100%;
        }
        form .user-details .input-box {
            margin-bottom: 15px;
            width: 100%;
        }
        form .category {
            width: 100%;
        }
        .content form .user-details {
            max-height: 300px;
            overflow-y: scroll;
        }
        .user-details::-webkit-scrollbar {
            width: 5px;
        }
    }

    @media ( max-width : 459px) {
        .container .content .category {
            flex-direction: column;
        }
        .user-details .input-box input {
            font-size: 14px;
        }
    }
    </style>
    </head>
    <body>
        <div class="container">
            <div class="title">Fill the Customer Details</div>
            <div class="content">
                <form action="customerInsert" method="POST">
                    <div class="user-details">
                        <div class="input-box">
                            <span class="details">First Name</span> <input type="text"
                                name="fname" placeholder="Enter your first-name"
    required>
                        </div>
                        <div class="input-box">
                            <span class="details">Last Name</span> <input type="text"
                                name="lname" placeholder="Enter your last-name"
    required>
                        </div>
                        <div class="input-box">
                            <span class="details">Email</span> <input type="email"
                                name="email" placeholder="Enter your email"
    required>
                        </div>
                        <div class="input-box">
                            <span class="details">Phone Number</span> <input
    type="text"
```

```
                                        name="phone" placeholder="Enter your phone
number" required>
                                </div>
                                <div class="input-box">
                                        <span class="details">Address</span> <input type="text"
                                        name="address" placeholder="Enter your address"
required>
                                </div>
                                <div class="input-box">
                                        <span class="details">Country</span> <input type="text"
                                        name="country" placeholder="Enter your country"
required>
                                </div>
                        </div>
                        <div class="button">
                                <input type="submit" value="Create Customer" name="submit">
                        </div>
                </form>
        </div>
    </div>
</body>
</html>
```

- **Calculator**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Error</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">
<link
    href="https://fonts.googleapis.com/css?family=Encode+Sans+Semi+Condensed:100,200,300,400"
    rel="stylesheet">
<link rel="stylesheet" href="./css/error.css">
</head>
<body class="loading">
    <h1>404</h1>
    <h2>Invalid Credentials!</h2>
    <br>
    <div class="gears">
        <div class="gear one" id="oone">
            <div class="bar"></div>
            <div class="bar"></div>
            <div class="bar"></div>
        </div>
        <div class="gear two" id="twoo">
            <div class="bar"></div>
            <div class="bar"></div>
            <div class="bar"></div>
        </div>
        <div class="gear three" id="three">
            <div class="bar"></div>
            <div class="bar"></div>
            <div class="bar"></div>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="js/main.js" type="text/javascript"></script>

</body>
</html>
```

- **Error**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Error</title>
<link rel="icon" type="image/x-icon" href="./image/logo.ico">
<link
    href="https://fonts.googleapis.com/css?family=Encode+Sans+Semi+Condensed:100,200,300,400"
    rel="stylesheet">
<link rel="stylesheet" href="./css/error.css">
</head>
<body class="loading">
    <h1>404</h1>
    <h2>Invalid Credentials!</h2>
    <br>
    <div class="gears">
            <div class="gear one" id="oone">
                <div class="bar"></div>
                <div class="bar"></div>
                <div class="bar"></div>
            </div>
            <div class="gear two" id="twoo">
                <div class="bar"></div>
                <div class="bar"></div>
                <div class="bar"></div>
            </div>
            <div class="gear three" id="three">
                <div class="bar"></div>
                <div class="bar"></div>
                <div class="bar"></div>
            </div>
    </div>
    <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="js/main.js" type="text/javascript"></script>

</body>
</html>
```

➢ JAVA Servlet Controller

• Login.java

```
package controller;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import daoImpl.DaoImpl;
import vo.ResultVO;
import vo.UserVO;
//import javax.servlet.RequestDispatcher;
//import daoImpl.DaoImpl;
//import vo.ResultVO;
//import vo.UserVO;
/**
 * Servlet implementation class Login
 */
@WebServlet("/Login")
public class Login extends HttpServlet {
	DaoImpl daoImpl=new DaoImpl();
//	ResultVO rvo = new ResultVO();
	private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
	public Login() {
		super();
		// TODO Auto-generated constructor stub
	}
	// From login.jsp, as a post method only the credentials are passed
	// Hence the parameters should match both in jsp and servlet and
	// then only values are retrieved properly
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		// We can able to get the form data by means of the below ways.
		// Form arguments should be matched and then only they are recognised
		// login.jsp component names should match and then only
		// by using request.getParameter, it is matched
		String username = request.getParameter("username");
		String password = request.getParameter("password");
		// To verify whether entered data is printing correctly or not
		System.out.println("username.." + username);
		System.out.println("password.." + password);
		UserVO uvo=new UserVO();
		uvo.setUsername(username);
		uvo.setPassword(password);
		String res=daoImpl.Login(uvo);
		if(res.equals("true"))
		{
			HttpSession httpSession = request.getSession(true);
//			 By setting the variable in session, it can be forwarded
			httpSession.setAttribute("username", username);
			request.getRequestDispatcher("home.jsp").forward(request, response);
		}
		else {
			request.getRequestDispatcher("error.jsp").forward(request, response);
		}
	}
}
```

- **Registration Servlet.java**

```
package controller;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.RequestDispatcher;
import daoImpl.DaoImpl;
import vo.ResultVO;
import vo.UserVO;
/**
 * Servlet implementation class RegistrationServlet
 */
@WebServlet("/registrationServlet")
public class RegistrationServlet extends HttpServlet {
  DaoImpl daoImpl=new DaoImpl();
  ResultVO rvo = new ResultVO();
  private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public RegistrationServlet() {
        super();
        // TODO Auto-generated constructor stub
    }
  /**
   * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
   */
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            response.getWriter().append("Served at: ").append(request.getContextPath());
  }
  /**
   * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
   */
  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
//          doGet(request, response);
            try {
                    String username=request.getParameter("username");
                    String password=request.getParameter("password");
                    UserVO uvo=new UserVO();
                    uvo.setUsername(username);
                    uvo.setPassword(password);
                    rvo = daoImpl.dataInsert(uvo);
                    RequestDispatcher rd= getServletContext().getRequestDispatcher("/login.jsp");
                    rd.include(request, response);
            }
            catch(Exception e)
            {
                    e.printStackTrace();
            }
  }

}
```

- **InvoiceFrom.java**

```
package controller;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import dbconnection.Dbconnection;

@WebServlet("/invoiceForm")
public class InvoiceForm extends HttpServlet {
        public Connection con = null;
        public PreparedStatement ps1 = null;
        public PreparedStatement ps2 = null;
        public PreparedStatement ps3 = null;
        Dbconnection dc = new Dbconnection();
        private final static String query1 = "select fname,lname from customer";
        private final static String query2 = "select pname,price from product where pname=?";
        private final static String query3 = "select count(*) from invoice";
   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
      //get PrintWriter
      PrintWriter pw = res.getWriter();
      //set content type
      res.setContentType("text/html");
      //get the id
      String pname = req.getParameter("pname");
      pw.println("<title>Invoice Form</title>");
      pw.println("<link rel='icon' type='image/x-icon' href='./image/logo.ico'>");
      pw.println("<link rel='stylesheet' href='css/invoiceform.css' />");
      //generate the connection
      try{
          con = dc.dbconnect();
                        ps1 = con.prepareStatement(query1);
                        ps2 = con.prepareStatement(query2);
                        ps3 = con.prepareStatement(query3);
          ps2.setString(1, pname);
         //resultSet
         ResultSet rs1 = ps1.executeQuery();
         ResultSet rs2 = ps2.executeQuery();
         ResultSet rs3 = ps3.executeQuery();
         pw.println("<div class='container'>");
         pw.println("<div class='title'>Fill the Invoice Details</div>");
         pw.println("<div class='content'>");
         pw.println("<form action='InvoiceInsert' method='POST'>");
```

```
        pw.println("<div class='user-details'>");
        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Invoice Date</span>");
        pw.println("<input id='invoiceDate' type='date' name='invoiceDate' placeholder='Enter the invoice date'
required>");
        pw.println("</div>");

        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Due Date</span>");
        pw.println("<input id='dueDate' type='date' name='dueDate' placeholder='Enter the due date' required>");
        pw.println("</div>");

        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Invoice Number</span>");
        rs3.next();
        pw.println("<input id='invoiceNo' type='text' name='invoiceNo' value='"+(rs3.getInt(1)+1)+"'
placeholder='Enter the invoice number' required>");
        pw.println("</div>");

        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Invoice Status</span>");
        pw.println("<select id='invoiceStatus' 'text' name='invoiceStatus' required>");
        pw.println("<option value='open'>Open</option>");
        pw.println("<option value='paid'>Paid</option>");
        pw.println("</select>");
        pw.println("</div>");

        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Product</span>");
        rs2.next();
        pw.println("<input id='product' type='text' name='product' placeholder='Enter the Product'
value='"+rs2.getString(1)+"' required>");
        pw.println("</div>");

        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Select a Customer</span>");
        pw.println("<select id='customer' type='text' name='customer' required>");
        while (rs1.next()) {
         pw.println("<option value='"+rs1.getString(1)+" "+rs1.getString(2)+"'>"+rs1.getString(1)+"
"+rs1.getString(2)+"</option>");
        }
        pw.println("</select>");
        pw.println("</div>");

        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Quantity</span>");
        pw.println("<input id='quantity' type='number' min='1' name='quantity' placeholder='Enter the quantity'
required>");
        pw.println("</div>");

        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Price</span>");
```

```
            pw.println("<input id='price' type='number' name='price' placeholder='Price of the Product'
value='"+rs2.getString(2)+"' required>");
            pw.println("</div>");

            pw.println("<div class='input-box'>");
            pw.println("<span class='details'>Discount</span>");
            pw.println("<input id='discount' type='number' name='discount' placeholder='Enter the discount percentage'
required>");
            pw.println("</div>");

            pw.println("<div class='input-box'>");
            pw.println("<span class='details'>Shipping Cost</span>");
            pw.println("<input id='shippingCost' type='number' name='shippingCost' placeholder='Enter the shipping cost'
required>");
            pw.println("</div>");

//            pw.println("<div class='subtotal' style='margin-top: 1%;'>");
//            pw.println("<div class='sub'>");
//            pw.println("<h4>TAX & VAT :</h4>");
//            pw.println("<p id='tax'></p>");
//            pw.println("</div>");
//            pw.println("<div class='sub'>");
//            pw.println("<h4>SUB TOTAL :</h4>");
//            pw.println("<p id='total'>0 Rupees</p>");
//            pw.println("</div>");
//            pw.println("</div>");

            pw.println("</div>");

            pw.println("<div class='button'>");
            pw.println("<input id='submit' type='submit' value='Create Invoice' name='submit'>");
            pw.println("</div>");
            pw.println("</form>");
        }catch(SQLException se) {
            pw.println("<h2 class='bg-danger text-light text-center'>"+se.getMessage()+"</h2>");
            se.printStackTrace();
        }catch(Exception e) {
            e.printStackTrace();
        }
        pw.println("</div>");
        pw.println("</div>");
        //close the stream
        pw.close();
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        doGet(req,res);
    }
}
```

- **Download Invoice.java**

```
package controller;
import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfWriter;
import java.io.*;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import dbconnection.Dbconnection;

/**
 * Servlet implementation class DownloadInvoice
 */
@WebServlet("/DownloadInvoice")
public class DownloadInvoice extends HttpServlet {
	public Connection con = null;
	public PreparedStatement ps = null;
	Dbconnection dc = new Dbconnection();
	private final static String query = "select
id,invoiceNo,customer,product,price,invoiceDate,dueDate,invoiceStatus,quantity,discount,shippingCost from invoice
where id=?";
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public DownloadInvoice() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
		// TODO Auto-generated method stub
//		response.getWriter().append("Served at: ").append(request.getContextPath());
		int id = Integer.parseInt(request.getParameter("id"));
		response.setContentType("application/pdf");
```

```
        response.setHeader(
           "Content-disposition",
           "inline; filename='Download Invoice.pdf'");
        try{
            con = dc.dbconnect();
                                ps = con.prepareStatement(query);
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            rs.next();
            int price=Integer.parseInt(rs.getString(5));
            int quantity=Integer.parseInt(rs.getString(9));
            int discount=Integer.parseInt(rs.getString(10));
            int shippingCost=Integer.parseInt(rs.getString(11));
            double total=((price*quantity)-((price*quantity)*discount)/100)+shippingCost;
           Document document = new Document();
           PdfWriter.getInstance(
              document, response.getOutputStream());
           document.open();
           document.add(new Paragraph(
                 "This is Your Invoice , Please Download it"));
           document.add(new Paragraph("Invoice Id :"+rs.getInt(1)));
           document.add(new Paragraph("Invoice Number :"+rs.getString(2)));
           document.add(new Paragraph("Customer Name :"+rs.getString(3)));
           document.add(new Paragraph("Product Name :"+rs.getString(4)));
           document.add(new Paragraph("Price of the Product :"+rs.getString(5)));
           document.add(new Paragraph("Invoice Date :"+rs.getString(6)));
           document.add(new Paragraph("Due Date :"+rs.getString(7)));
           document.add(new Paragraph("Invoice Status :"+rs.getString(8)));
           document.add(new Paragraph("Quantity :"+rs.getString(9)));
           document.add(new Paragraph("Discount :"+rs.getString(10)+"%"));
           document.add(new Paragraph("Shipping Cost :"+rs.getString(11)+" inr"));
           document.add(new Paragraph("You have to pay :"+total+" inr"));
           document.close();
        }
        catch (DocumentException de) {
           throw new IOException(de.getMessage());
        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
            }
            /**
             * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
             *
             */
            protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
                        // TODO Auto-generated method stub
                        doGet(request, response);
            }

}
```

- **Product From.java**

```
package controller;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import dbconnection.Dbconnection;

@WebServlet("/editproducturl")
public class EditProductForm extends HttpServlet {
        public Connection con = null;
        public PreparedStatement ps = null;
        Dbconnection dc = new Dbconnection();
        private final static String query = "select pname,pdescription,price from product where id=?";
   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
      //get PrintWriter
      PrintWriter pw = res.getWriter();
      //set content type
      res.setContentType("text/html");
      //get the id
      int id = Integer.parseInt(req.getParameter("id"));
      pw.println("<title>Product Update form</title>");
      pw.println("<link rel='stylesheet' href='css/productform.css' />");
      //generate the connection
      try{
          con = dc.dbconnect();
                        ps = con.prepareStatement(query);
          //set value
         ps.setInt(1, id);
         //resultSet
         ResultSet rs = ps.executeQuery();
         rs.next();
         pw.println("<div class='container'>");
         pw.println("<div class='title'>Fill the Product Details</div>");
         pw.println("<div class='content'>");
         pw.println("<form action='ProductUpdate?id="+id+"' method='POST'>");
         pw.println("<div class='user-details'>");
         pw.println("<div class='input-box'>");
         pw.println("<span class='details'>Product Name</span>");
         pw.println("<input type='text' name='pname' placeholder='Enter Product Name' value='"+rs.getString(1)+"'
required>");
         pw.println("</div>");
         pw.println("<div class='input-box'>");
```

```
            pw.println("<span class='details'>Price</span>");
            pw.println("<input type='text' name='price' placeholder='Enter the Price' value='"+rs.getString(3)+"'
required>");
            pw.println("</div>");
            pw.println("<div class='input-box1'>");
            pw.println("<span class='details'>Product Description</span>");
            pw.println("<textarea id= cols='30' rows='10' name='pdescription' required>"+rs.getString(2)+"</textarea>");
            pw.println("</div>");
            pw.println("</div>");
            pw.println("<div class='button'>");
            pw.println("<input type='submit' value='Update Product' name='submit'>");
            pw.println("</div>");
            pw.println("</form>");
        }catch(SQLException se) {
            pw.println("<h2 class='bg-danger text-light text-center'>"+se.getMessage()+"</h2>");
            se.printStackTrace();
        }catch(Exception e) {
            e.printStackTrace();
        }
        pw.println("</div>");
        pw.println("</div>");
        //close the stream
        pw.close();
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        doGet(req,res);
    }

}
```

- **Show Product.java**

```
package controller;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import dbconnection.Dbconnection;

@WebServlet("/showproductdata")
public class ShowProduct extends HttpServlet {
	public Connection con = null;
	public PreparedStatement ps = null;
	Dbconnection dc = new Dbconnection();
	private final static String query = "select id,pname,pdescription,price from product";

	@Override
	protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
		// get PrintWriter
		PrintWriter pw = res.getWriter();
		// set content type
		res.setContentType("text/html");
		// link the bootstrap
		pw.println("<title>Manage Product</title>");
		pw.println("<link rel='icon' type='image/x-icon' href='./image/logo.ico'>");
		pw.println(
				"<link rel=\"stylesheet\"
href=\"https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css\" integrity=\"sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm\" crossorigin=\"anonymous\">");
		pw.println(
				"<script src=\"https://code.jquery.com/jquery-3.2.1.slim.min.js\" integrity=\"sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN\"
crossorigin=\"anonymous\"></script>");
		pw.println(
				"<script src=\"https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js\"
integrity=\"sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q\"
crossorigin=\"anonymous\"></script>");
		pw.println(
				"<script src=\"https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js\"
integrity=\"sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl\"
crossorigin=\"anonymous\"></script>");
		pw.println(
				"<link rel=\"stylesheet\" href=\"https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.3.0/css/all.min.css\" />");
		pw.println("<link rel='stylesheet' href='css/manage.css' />");
		pw.println("<script src='js/tooltip.js'></script>");
//		pw.println("<marquee><h2 class='text-primary'>List of Products</h2></marquee>");


		// generate the connection
		try{
```

```
                        con = dc.dbconnect();
                        ps = con.prepareStatement(query);
                        // resultSet
                        ResultSet rs = ps.executeQuery();
                        pw.println("<div class='main-div'>");
                        pw.println("<h2>List of Products</h2>");
                        pw.println("<div class='center-div'>");
                        pw.println("<div class='table-responsive'>");
                        pw.println("<table>");
                        pw.println("<thead>");
                        pw.println("<tr>");
                        pw.println("<th>Product Name</th>");
                        pw.println("<th>Product Description</th>");
                        pw.println("<th>Price</th>");
                        pw.println("<th colspan='2'>Actions</th>");
                        pw.println("</tr>");
                        pw.println("</thead>");
                        while (rs.next()) {
                                pw.println("<tbody>");
                                pw.println("<tr>");
                                pw.println("<td>" + rs.getString(2) + "</td>");
                                pw.println("<td>" + rs.getString(3) + "</td>");
                                pw.println("<td>" + rs.getString(4) + "</td>");
                                pw.println(
                                                "<td><a style='color: black;' data-toggle='tooltip' data-placement='bottom'
title='UPDATE' href='editproducturl?id="
                                                                + rs.getInt(1) + "'><i class='fa-regular fa-pen-to-
square'></i></a></td>");
                                pw.println(
                                                "<td><a style='color: black;' data-toggle='tooltip' data-placement='bottom'
title='DELETE' href='ProductDelete?id="
                                                                + rs.getInt(1) + "'><i class='fa-solid fa-
trash'></i></a></td>");
                                pw.println("</tr>");
                                pw.println("</tbody>");
                        }
                        pw.println("</table>");
                } catch (SQLException se) {
                        pw.println("<h2 class='bg-danger text-light text-center'>" + se.getMessage() + "</h2>");
                        se.printStackTrace();
                } catch (Exception e) {
                        e.printStackTrace();
                }
                pw.println("</div>");
                pw.println("</div>");
                pw.println("</div>");
                // close the stream
                pw.close();
        }

        @Override
        protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
                doGet(req, res);
        }
}
```

- **Customer From.java**

```
package controller;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import dbconnection.Dbconnection;
@WebServlet("/editcustomerurl")
public class EditCustomerForm extends HttpServlet {
        public Connection con = null;
        public PreparedStatement ps = null;
        Dbconnection dc = new Dbconnection();
        private final static String query = "select fname,lname,email,phone,address,country from customer where id=?";
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
      //get PrintWriter
      PrintWriter pw = res.getWriter();
      //set content type
      res.setContentType("text/html");
      //get the id
      int id = Integer.parseInt(req.getParameter("id"));
      pw.println("<title>Customer Update form</title>");
      pw.println("<link rel='icon' type='image/x-icon' href='./image/logo.ico'>");
      pw.println("<link rel='stylesheet' href='css/customerform.css' />");

      //generate the connection
      try{
          con = dc.dbconnect();
                      ps = con.prepareStatement(query);
          //set value
        ps.setInt(1, id);
        //resultSet
        ResultSet rs = ps.executeQuery();
        rs.next();
        pw.println("<div class='container'>");
        pw.println("<div class='title'>Fill the Customer Details</div>");
        pw.println("<div class='content'>");
        pw.println("<form action='CustomerUpdate?id="+id+"' method='POST'>");
        pw.println("<div class='user-details'>");
        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>First Name</span>");
        pw.println("<input type='text' name='fname' placeholder='Enter your first-name' value='"+rs.getString(1)+"' required>");
        pw.println("</div>");
        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Last Name</span>");
        pw.println("<input type='text' name='lname' placeholder='Enter your last-name' value='"+rs.getString(2)+"' required>");
        pw.println("</div>");
        pw.println("<div class='input-box'>");
        pw.println("<span class='details'>Email</span>");
```

```
            pw.println("<input type='email' name='email' placeholder='Enter your email' value='"+rs.getString(3)+"' required>");
            pw.println("</div>");

            pw.println("<div class='input-box'>");
            pw.println("<span class='details'>Phone No.</span>");
            pw.println("<input type='text' name='phone' placeholder='Enter your phone number' value='"+rs.getString(4)+"'
required>");
            pw.println("</div>");
            pw.println("<div class='input-box'>");
            pw.println("<span class='details'>Address</span>");
            pw.println("<input type='text' name='address' placeholder='Enter your address' value='"+rs.getString(5)+"' required>");
            pw.println("</div>");

            pw.println("<div class='input-box'>");
            pw.println("<span class='details'>Country</span>");
            pw.println("<input type='text' name='country' placeholder='Enter your country' value='"+rs.getString(6)+"' required>");
            pw.println("</div>");

            pw.println("</div>");

            pw.println("<div class='button'>");
            pw.println("<input type='submit' value='Update Customer' name='submit'>");
            pw.println("</div>");
            pw.println("</form>");
        }catch(SQLException se) {
            pw.println("<h2 class='bg-danger text-light text-center'>"+se.getMessage()+"</h2>");
            se.printStackTrace();
        }catch(Exception e) {
            e.printStackTrace();
        }
        pw.println("</div>");
        pw.println("</div>");
        //close the stream
        pw.close();
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        doGet(req,res);
    }
}
```

- **Show Customer.java**

```
package controller;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import dbconnection.Dbconnection;
@WebServlet("/showcustomerdata")
public class ShowCustomer extends HttpServlet {
	public Connection con = null;
	public PreparedStatement ps = null;
	Dbconnection dc = new Dbconnection();
	private final static String query = "select id,fname,lname,email,phone from customer";
	@Override
	protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
		// get PrintWriter
		PrintWriter pw = res.getWriter();
		// set content type
		res.setContentType("text/html");
		// link the bootstrap
		pw.println("<title>Manage Customer</title>");
		pw.println("<link rel='icon' type='image/x-icon' href='./image/logo.ico'>");
		pw.println(
				"<link rel=\"stylesheet\"
href=\"https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css\" integrity=\"sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm\" crossorigin=\"anonymous\">");
		pw.println(
				"<script src=\"https://code.jquery.com/jquery-3.2.1.slim.min.js\" integrity=\"sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN\"
crossorigin=\"anonymous\"></script>");
		pw.println(
				"<script src=\"https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js\"
integrity=\"sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q\"
crossorigin=\"anonymous\"></script>");
		pw.println(
				"<script src=\"https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js\"
integrity=\"sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl\"
crossorigin=\"anonymous\"></script>");
		pw.println(
				"<link rel=\"stylesheet\" href=\"https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.3.0/css/all.min.css\" />");
		pw.println("<link rel='stylesheet' href='css/manage.css' />");
		pw.println("<script src='js/tooltip.js'></script>");
//		pw.println("<marquee><h2 class='text-primary'>List of Customers</h2></marquee>");

		try {
```

```
                        con = dc.dbconnect();
                        ps = con.prepareStatement(query);
                        // resultSet
                        ResultSet rs = ps.executeQuery();
                        pw.println("<div class='main-div'>");
                        pw.println("<h2>List of Customers</h2>");
                        pw.println("<div class='center-div'>");
                        pw.println("<div class='table-responsive'>");
                        pw.println("<table>");
                        pw.println("<thead>");
                        pw.println("<tr>");
                        pw.println("<th>First Name</th>");
                        pw.println("<th>Last Name</th>");
                        pw.println("<th>Email</th>");
                        pw.println("<th>Phone No</th>");
                        pw.println("<th colspan='2'>Actions</th>");
                        pw.println("</tr>");
                        pw.println("</thead>");
                        while (rs.next()) {
                                pw.println("<tbody>");
                                pw.println("<tr>");
                                pw.println("<td>" + rs.getString(2) + "</td>");
                                pw.println("<td>" + rs.getString(3) + "</td>");
                                pw.println("<td>" + rs.getString(4) + "</td>");
                                pw.println("<td>" + rs.getString(5) + "</td>");
                                pw.println(
                                                "<td><a style='color: black;' data-toggle='tooltip' data-
placement='bottom' title='UPDATE' href='editcustomerurl?id="
                                                                + rs.getInt(1) + "'><i class='fa-regular fa-pen-to-
square'></i></a></td>");
                                pw.println(
                                                "<td><a style='color: black;' data-toggle='tooltip' data-
placement='bottom' title='DELETE' href='customerDelete?id="
                                                                + rs.getInt(1) + "'><i class='fa-solid fa-
trash'></i></a></td>");
                                pw.println("</tr>");
                                pw.println("</tbody>");
                        }
                        pw.println("</table>");
                } catch (SQLException se) {
                        pw.println("<h2 class='bg-danger text-light text-center'>" + se.getMessage() + "</h2>");
                        se.printStackTrace();
                } catch (Exception e) {
                        e.printStackTrace();
                }
                pw.println("</div>");
                pw.println("</div>");
                pw.println("</div>");
                // close the stream
                pw.close();
        }
        @Override
        protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
                doGet(req, res);
        }
```

- **Daoimplimentation.java**

```
/**
 *
 */
package daoImpl;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import dbconnection.Dbconnection;
import vo.ResultVO;
import vo.UserVO;
import vo.CustomerVO;
import vo.ProductVO;
import vo.InvoiceVO;

import java.util.ArrayList;
import java.util.List;

/**
 * @author Rupayan Dirghangi
 *
 */
public class DaoImpl {
  public Connection con = null;
  public PreparedStatement ps = null;
  Dbconnection dc = new Dbconnection();
  ResultVO rvo = new ResultVO();
  boolean status = false;

  /**
   * @param args
   */
  public ResultVO dataInsert(UserVO uvo) {
            try {
                      String username = uvo.getUsername();
                      String password = uvo.getPassword();

                      con = dc.dbconnect();

                      String insertQuery = "insert into registration values(DEFAULT, ?, ?)";
                      ps = con.prepareStatement(insertQuery);
                      ps.setString(1, username);
                      ps.setString(2, password);

                      int i = ps.executeUpdate();
                      if (i > 0) {
                                System.out.println("Success");
                      } else {
                                System.out.println("Failed");
                      }

            } catch (Exception e) {
                      // TODO: handle exception
                      e.printStackTrace();
            }

            return rvo;
```

```
  }
//public ResultVO userDelete(UserVO uvo) {
//		try {
//			String email = uvo.getEmail();
//
//			con = dc.dbconnect();
//
//			String deleteQuery = "delete from user where email=?";
//			ps = con.prepareStatement(deleteQuery);
//			ps.setString(1, email);
//
//			int i = ps.executeUpdate();
//			if (i>0) {
//				System.out.println("Data Deleted");
//			}
//			else {
//				System.out.println("Failed");
//			}
//
//		} catch (Exception e) {
//			// TODO: handle exception
//			e.printStackTrace();
//		}
//
//		return rvo;
//
//}

  public ResultVO passwordUpdate(UserVO uvo) {
		try {
			String username = uvo.getUsername();
			String password = uvo.getPassword();

			con = dc.dbconnect();

			String updateQuery = "update registration set password=? where username=?";
			ps= con.prepareStatement(updateQuery);
			ps.setString(1, password);
			ps.setString(2, username);

			int i = ps.executeUpdate();
			if (i>0) {
				System.out.println("Data Updated");
			}
			else {
				System.out.println("Failed");
			}
		} catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
		}
		return rvo;
  }
  public String Login(UserVO uvo) {
		try {

			String username = uvo.getUsername();
			String password = uvo.getPassword();
```

```
                con = dc.dbconnect();

                String loginQuery = "select * from registration where username=? and password=?";
                ps = con.prepareStatement(loginQuery);
                ps.setString(1, username);
                ps.setString(2, password);

                ResultSet rs = ps.executeQuery();
                status = rs.next();
                if(status) {
                        return "true";
                }
                else {
                        return "false";
                }

        } catch (Exception e) {
                e.printStackTrace();
        }
        return "error";
  }

//Customer details
  public ResultVO customerInsert(CustomerVO cvo) {
        try {
                String fname = cvo.getFirstname();
                String lname = cvo.getLastname();
                String email = cvo.getEmail();
                String phone = cvo.getPhone();
                String address = cvo.getAddress();
                String country = cvo.getCountry();

                con = dc.dbconnect();

                String insertQuery = "insert into customer values(DEFAULT, ?, ?, ?, ?, ?, ?)";
                ps = con.prepareStatement(insertQuery);
                ps.setString(1, fname);
                ps.setString(2, lname);
                ps.setString(3, email);
                ps.setString(4, phone);
                ps.setString(5, address);
                ps.setString(6, country);

                int i = ps.executeUpdate();
                if (i > 0) {
                        System.out.println("Success");
                } else {
                        System.out.println("Failed");
                }

        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }

        return rvo;

  }
```

```
public ResultVO customerUpdate(CustomerVO cvo) {
	try {
		String fname = cvo.getFirstname();
		String lname = cvo.getLastname();
		String email = cvo.getEmail();
		String phone = cvo.getPhone();
		String address = cvo.getAddress();
		String country = cvo.getCountry();
		int id = cvo.getId();

		con = dc.dbconnect();

		String updateQuery = "update customer set fname=?,lname=?,email=?,phone=?,address=?,country=? where
id=?";
		ps= con.prepareStatement(updateQuery);
		ps.setString(1, fname);
		ps.setString(2, lname);
		ps.setString(3, email);
		ps.setString(4, phone);
		ps.setString(5, address);
		ps.setString(6, country);
		ps.setInt(7, id);

		int i = ps.executeUpdate();
		if (i>0) {
			System.out.println("Data Updated");
		}
		else {
			System.out.println("Failed");
		}
	} catch (Exception e) {
		// TODO: handle exception
		e.printStackTrace();
	}
	return rvo;
}

public ResultVO customerDelete(CustomerVO cvo) {
	try {
		int id = cvo.getId();

		con = dc.dbconnect();

		String deleteQuery = "delete from customer where id=?";
		ps = con.prepareStatement(deleteQuery);
		ps.setInt(1, id);

		int i = ps.executeUpdate();
		if (i>0) {
			System.out.println("Data Deleted");
		}
		else {
			System.out.println("Failed");
		}

	} catch (Exception e) {
		// TODO: handle exception
		e.printStackTrace();
```

```
		}

		return rvo;

	}


//Product details
	public ResultVO productInsert(ProductVO pvo) {
		try {
			String pname = pvo.getProductname();
			String price = pvo.getPrice();
			String pdescription = pvo.getPdescription();

			con = dc.dbconnect();

			String insertQuery = "insert into product values(DEFAULT, ?, ?, ?)";
			ps = con.prepareStatement(insertQuery);
			ps.setString(1, pname);
			ps.setString(2, price);
			ps.setString(3, pdescription);

			int i = ps.executeUpdate();
			if (i > 0) {
				System.out.println("Success");
			} else {
				System.out.println("Failed");
			}

		} catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
		}

		return rvo;

	}

	public ResultVO productUpdate(ProductVO pvo) {
		try {
			String pname = pvo.getProductname();
			String price = pvo.getPrice();
			String pdescription = pvo.getPdescription();
			int id = pvo.getId();

			con = dc.dbconnect();

			String updateQuery = "update product set pname=?,price=?,pdescription=? where id=?";
			ps= con.prepareStatement(updateQuery);
			ps.setString(1, pname);
			ps.setString(2, price);
			ps.setString(3, pdescription);
			ps.setInt(4, id);

			int i = ps.executeUpdate();
			if (i>0) {
				System.out.println("Data Updated");
			}
			else {
```

```
                        System.out.println("Failed");
                }
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        return rvo;
}

public ResultVO productDelete(ProductVO pvo) {
        try {
                int id = pvo.getId();

                con = dc.dbconnect();

                String deleteQuery = "delete from product where id=?";
                ps = con.prepareStatement(deleteQuery);
                ps.setInt(1, id);

                int i = ps.executeUpdate();
                if (i>0) {
                        System.out.println("Data Deleted");
                }
                else {
                        System.out.println("Failed");
                }

        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }

        return rvo;

}
//Invoice Section
public ResultVO invoiceInsert(InvoiceVO ivo) {
        try {
                String invoiceDate = ivo.getInvoiceDate();
                String dueDate = ivo.getDueDate();
                String invoiceNo = ivo.getInvoiceNo();
                String invoiceStatus = ivo.getInvoiceStatus();
                String product = ivo.getProduct();
                String customer = ivo.getCustomer();
                String quantity = ivo.getQuantity();
                String price = ivo.getPrice();
                String discount = ivo.getDiscount();
                String shippingCost = ivo.getShippingCost();

                con = dc.dbconnect();

                String insertQuery = "insert into invoice values(DEFAULT, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
                ps = con.prepareStatement(insertQuery);
                ps.setString(1, invoiceDate);
                ps.setString(2, dueDate);
                ps.setString(3, invoiceNo);
                ps.setString(4, invoiceStatus);
                ps.setString(5, product);
                ps.setString(6, customer);
```

```
                ps.setString(7, quantity);
                ps.setString(8, price);
                ps.setString(9, discount);
                ps.setString(10, shippingCost);

                int i = ps.executeUpdate();
                if (i > 0) {
                        System.out.println("Success");
                } else {
                        System.out.println("Failed");
                }

        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }

        return rvo;

 }

 public ResultVO invoiceUpdate(InvoiceVO ivo) {
        try {
                String invoiceDate = ivo.getInvoiceDate();
                String dueDate = ivo.getDueDate();
                String invoiceNo = ivo.getInvoiceNo();
                String invoiceStatus = ivo.getInvoiceStatus();
                String product = ivo.getProduct();
                String customer = ivo.getCustomer();
                String quantity = ivo.getQuantity();
                String price = ivo.getPrice();
                String discount = ivo.getDiscount();
                String shippingCost = ivo.getShippingCost();
                int id = ivo.getId();

                con = dc.dbconnect();

                String updateQuery = "update invoice set
invoiceDate=?,dueDate=?,invoiceNo=?,invoiceStatus=?,product=?,customer=?,quantity=?,price=?,discount=?,shippingCost=?
where id=?";
                ps= con.prepareStatement(updateQuery);
                ps.setString(1, invoiceDate);
                ps.setString(2, dueDate);
                ps.setString(3, invoiceNo);
                ps.setString(4, invoiceStatus);
                ps.setString(5, product);
                ps.setString(6, customer);
                ps.setString(7, quantity);
                ps.setString(8, price);
                ps.setString(9, discount);
                ps.setString(10, shippingCost);
                ps.setInt(11, id);

                int i = ps.executeUpdate();
                if (i>0) {
                        System.out.println("Data Updated");
                }
                else {
                        System.out.println("Failed");
```

```
			}
		} catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
		}
		return rvo;
	}


	public ResultVO invoiceDelete(InvoiceVO ivo) {
		try {
			int id = ivo.getId();

			con = dc.dbconnect();

			String deleteQuery = "delete from invoice where id=?";
			ps = con.prepareStatement(deleteQuery);
			ps.setInt(1, id);

			int i = ps.executeUpdate();
			if (i>0) {
				System.out.println("Data Deleted");
			}
			else {
				System.out.println("Failed");
			}

		} catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
		}

		return rvo;

	}
}
```

- **Database Connection**

```
/**
 *
 */
package dbconnection;

import java.sql.DriverManager;
import java.sql.Connection;

/**
 * @author Ripam Kundu
 *
 */
public class Dbconnection {
  public Connection con = null;

  /**
   * @param args
   */
  public static void main(String[] args) {
		// TODO Auto-generated method stub
		Dbconnection dc = new Dbconnection();
		dc.dbconnect();

  }
  public Connection dbconnect() {
		try {
			System.out.println("Connecting.....!");
			Class.forName("com.mysql.jdbc.Driver");
			con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ims","root","");
			System.out.println("Connected");
		}
		catch(Exception e) {
			System.out.println("CONNECTION ERROR");
			e.printStackTrace(); //printStackTrace is used to print the java sql exception
		}
		return con;
  }

}
```

Value Object (VO Implementation)

- **CustomerVO.java**

```
package vo;

public class CustomerVO {
	private int id;
	private String fname;
	private String lname;
	private String email;
	public CustomerVO() {

	}

	public CustomerVO(int id, String fname, String lname, String email, String phone, String
address, String country) {
		super();
		this.id = id;
		this.fname = fname;
		this.lname = lname;
		this.email = email;
		this.phone = phone;
		this.address = address;
		this.country = country;
	}
	public int getId() {
		return id;
	}


	public void setId(int id) {
		this.id = id;
	}
	public String getFirstname() {
		return fname;
	}


	public void setFirstname(String fname) {
		this.fname = fname;
	}


	public String getLastname() {
		return lname;
	}


	public void setLastname(String lname) {
		this.lname = lname;
	}


	public String getEmail() {
		return email;
	}


	public void setEmail(String email) {
		this.email = email;
```

```
	}


	public String getPhone()
	{
		return phone;
	}

	public void setPhone(String phone)
	{
		this.phone = phone;
	}


	public String getAddress()
	{
		return address;
	}


	public void setAddress(String address)
	{
		this.address = address;
	}


	public String getCountry()
	{
		return country;
	}


	public void setCountry(String country)
	{
		this.country = country;
	}

	private String phone;
	private String address;
	private String country;

}
```

- **InvoiceVO.java**

```
package vo;

public class InvoiceVO {
	private int id;
	private String invoiceDate;
	private String dueDate;
	private String invoiceNo;
	private String invoiceStatus;
	private String product;
	private String customer;
	private String quantity;
	private String price;
	private String discount;
	private String shippingCost;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getInvoiceDate() {
		return invoiceDate;
	}
	public void setInvoiceDate(String invoiceDate) {
		this.invoiceDate = invoiceDate;
	}
	public String getDueDate() {
		return dueDate;
	}
	public void setDueDate(String dueDate) {
		this.dueDate = dueDate;
	}
	public String getInvoiceNo() {
		return invoiceNo;
	}
	public void setInvoiceNo(String invoiceNo) {
		this.invoiceNo = invoiceNo;
	}
	public String getInvoiceStatus() {
		return invoiceStatus;
	}
	public void setInvoiceStatus(String invoiceStatus) {
		this.invoiceStatus = invoiceStatus;
	}
	public String getProduct() {
		return product;
	}
	public void setProduct(String product) {
		this.product = product;
	}
	public String getCustomer() {
		return customer;
	}
	public void setCustomer(String customer) {
		this.customer = customer;
	}
	public String getQuantity() {
		return quantity;
```

```
	}
	public void setQuantity(String quantity)
	{
		this.quantity = quantity;
	}
	public String getPrice()
	{
		return price;
	}
	public void setPrice(String price)
	{
		this.price = price;
	}
	public String getDiscount()
	{
		return discount;
	}
	public void setDiscount(String discount)
	{
		this.discount = discount;
	}
	public String getShippingCost()
	{
		return shippingCost;
	}
	public void setShippingCost(String shippingCost)
	{
		this.shippingCost = shippingCost;
	}

}
```

- **ProductVO.java**

```
package vo;

public class ProductVO
{
	private int id;
	private String pname;
	private String price;
	private String pdescription;

	public int getId()
	{
		return id;
	}
	public void setId(int id)
	{
		this.id = id;
	}
	public String getProductname()
	{
		return pname;
	}
	public void setProductname(String pname)
	{
		this.pname = pname;
	}
	public String getPrice()
	{
		return price;
	}
	public void setPrice(String price)
	{
		this.price = price;
	}
	public String getPdescription()
	{
		return pdescription;
	}
	public void setPdescription(String pdescription)
	{
		this.pdescription = pdescription;
	}
}
```

- **ResultVO.java**

```
/**
 *
 */
package vo;

/**
 * @author Ripam Kundu
 *
 */
public class ResultVO
{
	private boolean flag;

	public boolean isFlag()
	{
		return flag;
	}
	public void setFlag(boolean flag)
	{
		this.flag = flag;
	}
	/**
	 * @param args
	 */
	public static void main(String[] args)
	{
		// TODO Auto-generated method stub
	}
}
```

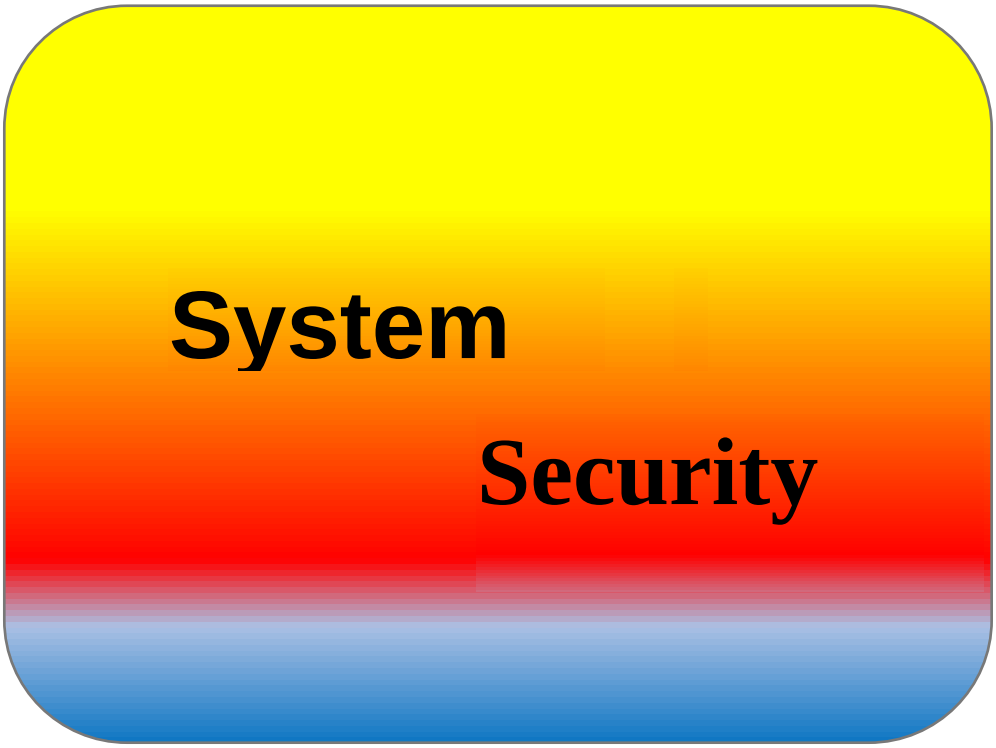
- **UserVO.java**

```
/**
 *
 */
package vo;

/**
 * @author Ripam Kundu
 *
 */
public class UserVO
{
//     Registration
       private String username;
       private String password;
       public String getUsername()
       {
              return username;
       }
       public void setUsername(String username)
       {
              this.username = username;
       }
       public String getPassword()
       {
              return password;
       }
       public void setPassword(String password)
       {
              this.password = password;
       }

//     Customer Details
//     Product Details

       /**
        * @param args
        */
       public static void main(String[] args)
       {
              // TODO Auto-generated method stub
       }
}
```


System
Security

8.1 DATABASE SECURITY

System security measure is meant to be provided to make your system reliable and secured from unauthorized user may create threats to the system. So you should follow some security measures. We have used security levels in database level at system level.

8.2 SYSTEM SECURITY

If we talk about the system security in our propose system we have implemented with the help of maintain the session throughout the system's use. Once has logged out than he/she will not be able to perform any task before signing back again.

A high level of authentic login is given to the system so this is a very tedious task to enter without authorization and authentication.

LIMITATION

- Excel export has not been developed for invoice system Item Category due to some criticality
- The transactions are executed in off-line mode, hence on-line data for Customer Order capture and modification is not possible.
- Off-line reports of Food Item. Confirm invoice. Customer cannot be generated due tobatch mode execution.
- Since it is an online project, Customers need internet connections to use it.
- People who are not familiar with computers can’t use this software.
- Customer must have debit card or credit card to book tickets.
- A description of the background and context of the project and its relation to workalready done in the area.
- Made statement of the aims and objectives of the project.
- The description of Purpose. Scope, and applicability
- We define the problem on which we are working in the project.
- We describe the requirement Specifications of the system and the actions thatcan be done on these things.
- We understand the problem domain and produce a model of the system, whichdescribes operations that can be performed on the system.
- We included features and operations in detail, including screen layouts
- We designed user interface and security issues related to system . Finally the system isimplemented and tested according to test cases

CONCULSION

This project has been appreciated by all the user in the organization. It is easy to use, since it uses the GUI provided in the user dialog. User friendly screens are provided. The usage of software increases the efficiency, decrease the effort. It has been efficiently employed as a Site management mechanism. It has been thoroughly tested and implemented. Although I have put my best efforts to make the software flexible, easy to operate but limitations cannot be ruled out even by me. Though the software presents a broad range of options to its users some intricate options could not be covered into it; partly because of logistic and partly due to lack of sophistication. Paucity of time was also major constraint; thus, it was not possible to make the software foolproof and dynamic Lack of time also compelled me to ignore some part such as storing old result of the candidate etc. Considerable efforts have made the software easy to operate even for the people not related to the field of computers but it is acknowledged that a layman may find it a bit problematic at the first instance. The user is provided help at each step for his convenience in working with the software.

FUTURE SCOPE AND FURTHER ENCHANCEMENT

The future scope and further enhancement of an invoice management system can be quite extensive, as technology continues to evolve and businesses seek more efficient ways to manage their invoicing processes. Here are some potential areas of improvement and future developments for an invoice management system:

Automation and Integration: One of the key areas of enhancement would be increasing automation and integrating the invoice management system with other systems and software used by businesses. This could include integrating with accounting software, customer relationship management (CRM) systems, and enterprise resource planning (ERP) systems to streamline the entire invoicing process and eliminate manual data entry.

Mobile Accessibility: Providing mobile accessibility to the invoice management system would enable users to access, create, and manage invoices on the go. Mobile apps can be developed to allow users to view invoice statuses, send reminders, and perform other invoice-related tasks conveniently from their smartphones or tablets.

Advanced Reporting and Analytics: Enhancing the reporting and analytics capabilities of the invoice management system can provide valuable insights to businesses. The system can generate customized reports and dashboards, allowing users to track invoice statuses, monitor payment trends, identify bottlenecks, and make data-driven decisions to optimize cash flow and financial operations.

Vendor Portals and Self-Service Features: Introducing vendor portals or self-service features within the invoice management system can empower suppliers and vendors to manage their own invoices. This includes functionalities such as invoice submission, status tracking, and communication channels, reducing the administrative burden on the business and improving collaboration with suppliers.

Enhanced Compliance and Regulation Features: Invoice management systems can be further enhanced to comply with evolving regulatory requirements, such as tax regulations, data privacy laws, and electronic invoicing standards. Integrating tax calculation engines, supporting digital signatures, and ensuring compliance with regional invoicing standards can provide businesses with a seamless and compliant invoicing process.

Enhanced Workflow and Approval Processes: Streamlining and automating workflow and approval processes can significantly improve efficiency and reduce processing time. Implementing customizable approval workflows, automated notifications, and alerts for pending actions can ensure timely processing and reduce the risk of invoice delays or errors.

These are just a few examples of the future scope and potential enhancements for an invoice management system. The specific improvements and features would depend on the needs and requirements of the businesses using the system, as well as the advancements in technology that emerge over time.

In future we would like to keep working on this project and make a new addition to provide users with more advanced features and more detailed information. We have set our sights on the following additions in future:

-

1. Forget password for admin and councilor.
2. Online payment process through debit and credit card.
3. Auto mail will be sent to the user’s email-id when new user creates their ID.

References

- www.wikipedia.com
- www.tutorialpoint.com
- www.geeksforgeeks.org
- www.nevonprojects.com

Project Upload Link (GitHub) :

- https://github.com/ripamkundu/Invoice-Management-System.git

Thank You