**BILLING APP**

**PROJECT REPORT**

Submitted to Bharathiar University in partial fulfilment of the requirements of the degree of Bachelor of Science in**Computer Science**

****

Submitted by:

**SRIVARSHNI M**

**(2022K0470)**

Under the Guidance of

**Mrs.J.POORNIMHA MCA., M.Phil., (Ph.D.)**

**(Assistant professor, Department of Computer Science)**

**DEPARTMENT OF COMPUTER SCIENCE**

**KG COLLEGE OF ARTS AND SCIENCE**

**Affiliated to Bharathiar University, Accredited by NAAC with A++**

**Saravanampatti, Coimbatore-641035.**

**APRIL-2023**

**CERTIFICATE**

This is certify that project entitled **“BILLING APP”** submitted to Bharathiar University fulfilment for the award of degree of Bachelor of Science in Computer Science is a record of original work done by **SRIVARSHNI M (2022K0470)** during the period of study in KG College of Arts and Science under the supervision of   
**Mrs. J. POORNIMHA** MCA., M.Phil., (Ph.D.) Assistant professor, Department of Computer Science.

**PLACE: Coimbatore**

**DATE:**

**Guide College Seal HoD**

**Viva-Voce examination held on**

**Internal Examiner** **External Examiner**

**DECLARATION**

I hereby declare that the project entitled **“BILLING APP”** by **SRIVARSHNI M (2022K0470)** in partial fulfilment of requirements for the award of degree of B.Sc. Computer Science at KG College of Arts and Sciencein anauthentic record of my own work carried out under the supervision of **Mrs.J.POORNIMHA MCA., M.Phil., (Ph.D.) Assistant professor, Department of Computer Science.**The content presented has not been submitted by me in any other University/Institution for the award of degree of **B.Sc. Computer Science.**

**PLACE: Coimbatore SRIVARSHNI M**

**DATE: (2022K0470)**

**ACKNOWLEGEMENT**

I express my sincere thanks to **Dr. ASHOK BAKTHAVATHSALAM,** Managing Trustee of KG College of Arts and Science for giving me an opportunity to do this course of study and to undertake this project work.

It is my pleasure to express my grateful thanks to **Dr. B. VANITHA,** Secretary, KG College of Arts and Science, for her valuable thoughts and guidance for completing this project successfully.

I express my humble thanks to **Dr. J. RATHINAMALA,** Principal, KG College of Arts and Science, for encouraging and helping me for the completion of this project.

I take this opportunity to convey my heartfelt wishes to **Dr. P. KRISHNAPRIYA,**Vice Principal, KG College of Arts and Science, for her moral support throughout the course.

I express my sincere thanks to **Dr. P. AJITHA,** Dean, Academics Science, KG College of Arts and Science, for her constant motivation and endless support.

With deep I express my profound thanks **to Dr.A.GOKILAVANI**, Head, Department of computer science,KG College of arts and science, for her constant encouragement and motivation throughout the project. I thank my guide **Ms.J.POORNIMHA** for her endless support and encouragement towards the completion of my project work.

**CONTENTS**

**DECLARATION**

**1. INTRODUCTION**

**1.1 ABOUT THE PROJECT**

**1.2 OBJECTIVES**

**1.3 SCOPE**

**1.4 ORGANISATION PROFILE**

**2. SYSTEM ANALYSIS AND FEASIBILITY STUDY**

**2.1 EXISTING SYSTEM**

**2.1.1 DRAWBACKS**

**2.2 PROPOSED SYSTEM**

**2.2.1 FEATURES**

**3. SYSTEM SPECIFICATION**

**3.1 HARDWARE CONFIGURATION**

**3.2 SOFTWARE CONFIGURATION**

**3.3 REQUIREMENT ANALYSIS**

**4. SYSTEM DESIGN**

**4.1 INPUT DESIGN**

**4.2 OUTPUT DESIGN**

**4.3 DATAFLOW DIAGRAM**

**4.5 DESCRIPTION MODULE**

**5. SYSTEM TESTING AND IMPLEMENTATION**

**5.1 SYSTEM TESTING**

**5.2 TESTING TYPES**

**5.3 SYSTEM IMPLEMENTATION**

**6. SYSTEM MAINTENANCE**

**7. CONCLUSION**

**8. APPENDICES**

**8.1 BIBLIOGRAPHY**

**8.2 SAMPLE CODING**

**8.3 SAMPLE INPUT**

**8.4SAMPLE OUTPUT**

# INTRODUCTION

**1.1 ABOUT THE PROJECT**

The billing application project aims to develop a software application to streamline and maintain the billing process for businesses. The application will allow users to input customer information, product details, and pricing information. It will generate invoices or bills for the customers, and maintain records of all transactions.

The billing application will offer a user-friendly interface, accessible through desktop or mobile devices, with easy navigation and intuitive design. It will ensure the accuracy of data entered and store invoices automatically to generate bills accurately.

The application will provide various functionalities such as generating reports, tracking bills, and generating receipts, etc, the project's goal is to help businesses save time and reduce human error in invoicing and billing. It will increase efficiency in managing accounts receivable and cash flow for businesses, leading to increased profitability and customer satisfaction.

Overall, the billing application project will address the need for a modern, comprehensive and efficient billing system. It will play a critical role in the success of businesses by improving their efficiency, accuracy, and security of the billing and payment process. It will also provide valuable insights into financial data, enabling businesses to make informed decisions to optimize their revenue and profitability.

With the increasing globalization and digitalization of businesses, the billing application will enable businesses to manage their billing processes smoothly and efficiently across different locations and time zones. The application will be scalable, flexible, and easy to use, allowing businesses to customize it to their unique requirements.

**1.2 OBJECTIVES**

**1. Accurately calculate and generate invoices:** The primary objective of a billing application is to calculate and generate accurate invoices for the products or services provided by a business.

**2. Manage customer data:** A billing application must be able to manage customer data and keep track of their billing details such as payment history, outstanding balances, and credit limits.

**3. Generate financial reports:** A billing application should be able to generate financial reports for various purposes, such as tracking revenue, analyzing customer behavior, and forecasting future income.

**4. Enhance security:** A billing application should have enhanced security features to protect sensitive customer data, prevent fraud, and ensure compliance with regulations such as GDPR and PCI.

**5. Offer a mobile application:** In today's mobile-centric world, a billing application should also offer a mobile application for customers to make payments, update their billing details, and access invoices on the go.

**1.3 SCOPE**

The scope of a billing application depends on the specific requirements of the organization that will use the software. In general, the scope of a billing application includes the following:

1. **Creating invoices -** The billing application should allow users to create invoices and store it for future reference.

**2. Managing customer data -** The application should be able to store customer information such as customer name, billing history such as time and date of purchase.

**4. Generating reports -** The application should be able to generate reports such as billing history.

Overall, the scope of the billing application should cover all aspects of the billing process from creating invoices and store reports for analysis, finally, it should have robust data backup and recovery features to ensure the security and integrity of valuable customer data.

**1.4 ORGANISATIONAL PROFILE**

The billing application provides an online platform for businesses to manage and process their invoicing. The application is designed to streamline billing processes while providing businesses with robust invoice management tools.

The application is used by small to medium-sized enterprises (SMEs) to manage their invoicing and receipts . The platform supports multiple inputs and offers a range of features including financial reporting. This integration offers businesses a seamless end-to-end solution for managing their financial management process.

The application is built on cloud-based technology, making it accessible to a global audience. The platform allows for easy invoice creation and management, offering billing team 24/7 support from a dedicated invoice management portal.

Overall, the billing application is a comprehensive financial management tool for SMEs, providing simple and efficient solutions for businesses to manage their billing activities, improving productivity, and reducing costs associated with manual billing processes.

1. **SYSTEM ANALYSIS AND FEASIBILITY STUDY** 
   1. **EXISTING SYSTEM**

The existing system for billing application is manually operated, and the process involves generating bills using paper invoices. The process is time-consuming and prone to errors because of its manual nature. Moreover, the system lacks a centralized database for storing customer information and payment details.

The feasibility study reveals that the existing system is inefficient and requires modernization to improve its processes. A computerized billing system can automate these processes and enhance operations, increase user efficiency and performance, reduce manual error, and store all data securely in the system.

To build an automated billing application, we must prioritize factors such as time, budget, human resources, and technological advancements. An automated billing system may increase overhead costs, require additional staff, and make training mandatory.

We may also need to consider operational and technical feasibility, such as the compatibility of the billing system on various devices and its integration with other applications. Operational feasibility is the ability of the system to meet its business requirements, while technical factors involve its compatibility with existing hardware and software devices.

Based on the feasibility study, we recommend developing an automated billing system that is efficient, error-free, and customizable to meet the needs of the billing operation. With an online billing application in place, saving time and decreasing billing turnaround times.

* 1. **DRAWBACKS**

**1. Inadequate Data Security:** A billing application may contain sensitive information such as customer details and product information. If the billing system is not secure, it can result in a data breach and compromise the privacy of the customers.

**2. Technical Issues:** Billing applications can suffer from technical glitches and malfunctions that may cause disruption in the billing process, leading to problems for both the customers and the business. Technical inefficient can further aggravate these issues.

**3. Inflexibility:** Some billing applications lack flexibility and customization options, which means the business may need to compromise on its billing processes to work with the application. This can lead to inefficient processes and lower productivity.

**4. Limited Integration:** Some billing applications may be incompatible with other software systems in the organization, leading to a lack of integration and reduced automation options.

**5. Cost:** Developing or implementing a billing application can be a costly process. The cost of developing, maintaining and updating the application needs to be taken into consideration, and the ROI needs to be analyzed.

**6. User Adoption:** If employees are not trained adequately, they may not be able to use the billing application effectively. This could result in errors in the billing process.

**7. Lack of Features:** Some billing applications may be lacking in features that are crucial for the business. This could lead to using multiple applications and manual processes in combination, making the overall billing process more complex and difficult to manage.

* 1. **PROPOSED SYSTEM**

The proposed system for the billing application is aimed at creating an efficient billing process for a company that offers products and services to its customers. The system will streamline the billing process, making it quicker, more accurate, saving time and resources for the company. This system is critical to the success of the business, and hence this feasibility study has been conducted to identify the viability of the project.

Objectives of the Proposed System:

1. To automate the billing process and develop an efficient and user-friendly system.

2. To reduce the time taken to generate invoices and bills.

3. To minimize billing errors and discrepancies.

4. To provide timely and accurate bills and to store the invoices.

6. To ensure data security and integrity.

Analysis of the Proposed System:

The proposed system will consist of a user interface that will allow the user to enter the necessary data, such as customer details, product/service details, pricing details, etc. Once the relevant data has been entered, the system will automatically generate invoices and bills. The system will have the following features:

1. **User-friendly interface:** The system will be easy to use, with a simple and intuitive interface.
2. **Automation:** The system will automate the billing process, reducing the time taken to generate invoices and bills.

**3. Billing accuracy:** The system will minimize billing errors and discrepancies, ensuring accurate bills for customers.

**4. Flexibility:** The system will be flexible, allowing users to customize the billing process according to their needs.

**5. Security:** The system will ensure data security and integrity, preventing unauthorized access and maintaining the confidentiality of customer information.

Feasibility Analysis:

**1. Technical Feasibility:** The proposed system is technically feasible, as it can be developed using existing technologies and software tools.

**2. Economic Feasibility:** The proposed system is economically feasible, as it can save the company time and resources, resulting in cost savings in the long run.

**3. Operational Feasibility:** The proposed system is operationally feasible, as it can be easily integrated into the company's existing processes and procedures.

Conclusion:

Based on the feasibility analysis, the proposed system for the billing application is viable and practical. It offers a cost-effective solution to streamline the billing process and minimize errors and discrepancies. The system's user-friendly interface, flexibility, and reporting capabilities will provide significant benefits to the company. The proposed system will enhance the company's revenue, increase its profitability, and provide accurate and timely bills.

* 1. **FEATURES**

**1. User-friendly Interface:** The billing app needs to have a simple, intuitive interface that can be easily used by both the user and the billing staff.

**2. Customizable Templates:** The app should provide customizable billing templates that can be used to generate invoices, receipts, and other documents.

**5. Customer Management:** The app should offer customer management features such as customer data storage and easy access to billing information.

**6. Security Measures:** The app should have robust security measures to offer protection against unauthorized access and data breaches.

**7. Compatibility with Different Devices:** The billing app should be compatible with different devices, including desktops, laptops, tablets, and smartphones, to enable easy access and usage.

**8. Cost-effectiveness:** The app should be cost-effective to use, and monthly or yearly subscription fees should be reasonable.

**3.SYSTEM SPECIFICATION**

**3.1 HARDWARE CONFIGURATION**

Processor : Intel i5

Speed : 2.4 GHz

Hard disk : 700 GB

Monitor : 15 VGA color

Mouse : Logitech.

RAM : 4 GB

Keyboard : 110 keys enhanced

**3.2 SOFTWARE CONFIGURATION**

Operating system : Windows 10 pro

Package Manager : NPM (Node Package Manager)

Coding Language : javascript

Front end Library : ReactJS

State Managemant : Redux

**3.3 REQUIREMENT ANALYSIS**

Introduction:

The billing application is a software tool that is responsible for recording, tracking, and billing transactions for goods and services. This type of application is commonly used by enterprises to automate routine billing processes and to ensure timely payment collection from clients. The billing application should be designed to fulfill some basic requirements to utilize an efficient system. This document outlines the specifications and requirements that the billing application needs to meet.

Functional Requirements:

1. Create and Manage invoice:

The billing system should allow for the creation of invoice including customer name. Additionally, the system should support adding and removing customers name, viewing customer information and update their invoices.

2. Generate and Manage Invoices:

The system should be able to create and manage invoices based on customer orders or generate automatically recurring invoices. Invoices shall be stored within the system and available for view in the dashboard.

3. Reporting and Analytics:

The billing application should allow for easy access to transaction history and generate reports showing transaction history, total earnings or revenue collected, breakdown of payments by customer and other basic analytics.

Non-Functional Requirements:

1. Security:

The billing application must ensure the safety and security of sensitive data like customer payment details, billing data and financial records to keep the system’s creditworthiness intact.

2. Scalability:

The application must be flexible with an option to scale-up as the business grows or if customer demands increase.

3. Performance:

The billing application must be responsive and user-friendly in a way that provides smooth navigation regardless of the volume of transactions processed.

4. Reliability:

The billing application must be stable and available to use throughout the system’s operational hours with minimal downtime for maintenance.

Conclusion:

In conclusion, designing a billing application that meets all the business requirements is essential. The application should have both functional and non-functional features to make it efficient and scalable. A highly functional billing application is an essential tool that every business needs to ensure they receive payment for their services fairly and reliably.

**4. SYSTEM DESIGN**

**4.1 INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**Input Stages**

The main input stages can be listed as below:

**Data recording**

Input data like product details are recorded or stored in order to process and deliver corresponding invoice details for future reference.

**Data conversion**

Input data is converted from text to binary by entering the values in the system.

**Data verification**

Input data is verified for approximation. If a person selects edit, add and delete will be verified, i.e. he or she will be checked whether they got updated information or not.

**Data validation**

Input data is validated for accuracy. If a person selects vehicle number 1, person will not only be checked whether he or she got vehicle details, specifications required by them i.e. details for selected vehicle must be checked for accuracy.

**Data correction**

In case of any errors in the input data like vehicle number or speed data will be corrected.

**4.2 OUTPUT DESIGN**

Output Design:

The billing app should have a simple, user-friendly interface that allows users to quickly and easily create bills and invoices. The system should generate clear, easy-to-read invoices that include all necessary information, such as the customer's name and time date of invoice generated, the items or services purchased, the quantity and price of each item.

System Design:

The billing app should be designed with scalability in mind, allowing it to handle large volumes of transactions and data over time. The system should be built using a modular, object-oriented approach, with clean, well-structured code that is easy to maintain and update.

To ensure maximum performance and reliability, the system should be hosted on a high-quality cloud-based infrastructure, with built-in redundancy and failover capabilities to minimize downtime and data loss. Finally, the system should be designed to be easily customizable, allowing users to tailor it to their specific business needs and requirements.

**4.3 DATAFLOW DIAGRAM**

The following is an example of a system design and dataflow diagram for a billing app:

**Virtual DOM**: Like an actual DOM, virtual DOM is also a node tree that lists the elements and their attributes and content as Objects and their properties. React’s render function creates a node tree out of the React components. Then, it updates this tree in response to the mutations in data model caused by various actions done either by the user or by the system.

System Design:



- The billing app will be a web-based application built using HTML, CSS, JavaScript, and Reactjs on the client side.

- The app will use a Redux state management to store customer information, invoices, and payment details.

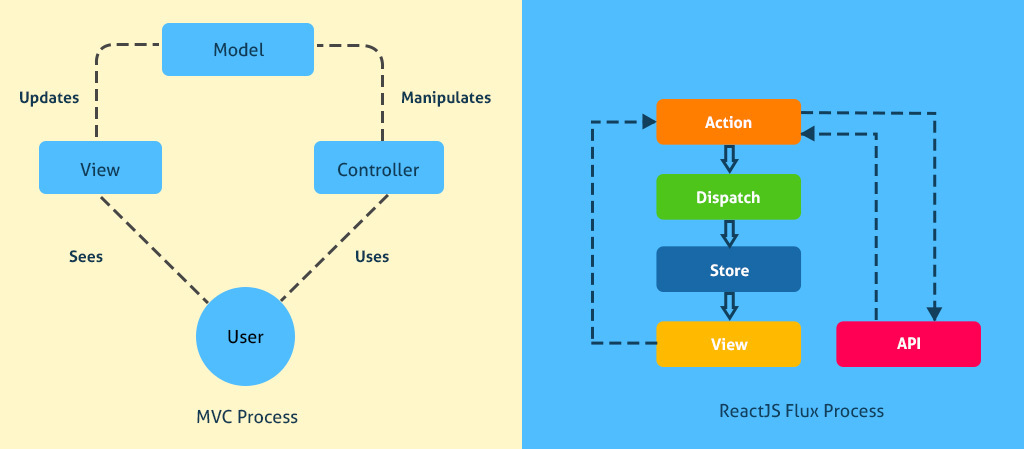
- The app will have three main sections: invoicing, invoice history and contact.

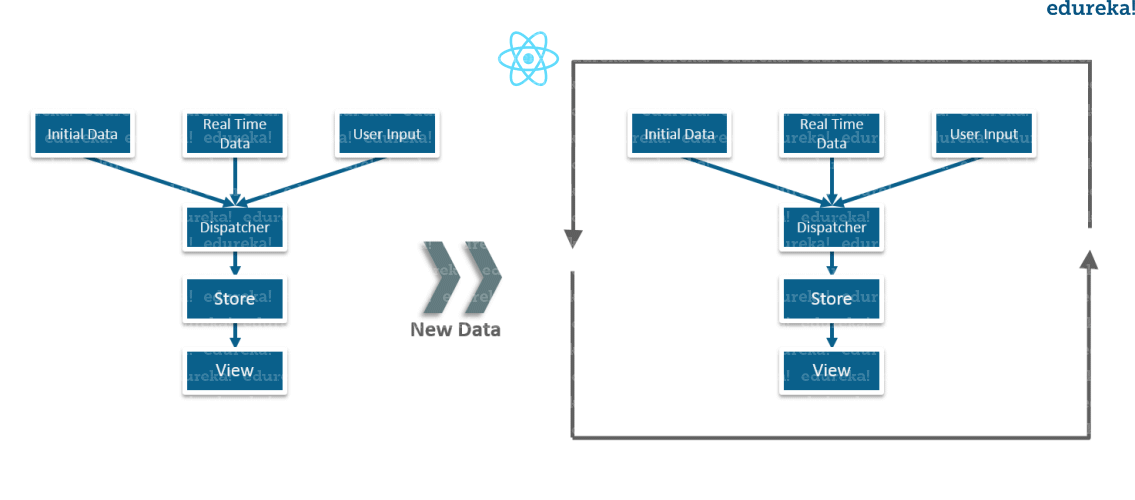
- The stock management section will allow users to add, edit, and delete stock information, including product name, price, quantity terms.

- The invoicing section will allow users to create new invoices, view existing invoices, and generate reports on outstanding invoices.

- The app will have built-in security features to protect user data and prevent unauthorized access, such as encryption of sensitive information, secure authentication, and role-based access control.

Dataflow Diagram:





The dataflow diagram for the billing app is as follows:

- The user interacts with the system through a web interface.

- The web server receives the user's request and sends it to the appropriate module based on the request type.

- The customer management module interacts with the database to add, edit or delete customer information. It sends responses about the success or failure of these operations back to the web server, which displays the appropriate message to the user.

- The invoicing module allows the user to create, view or edit invoices. It retrieves the necessary information from the database and generates PDF versions of the invoices for printing or email. It also sends data to the payment processing module when new invoices are created.

**4.5 DESCRIPTION MODULE**

The description module of the billing app will include all the necessary information about the products or services that the business offers. This module will include details such as product name, description, price, quantity, and any other relevant information that may be needed for the creation of invoices. This module will also allow for the addition, deletion, and modification of products or services within the app.

System Design:

The billing app will have a user-friendly interface that will allow for easy navigation and use. The system design will include the following features:

1. Dashboard: The dashboard will provide an overview of all the invoices created, invoice history, and stocks.

2. Product Management: The app will allow for the addition, deletion, and modification of products or services offered by the business.

3. Invoice Creation: The app will allow for the creation of invoices which will include the customer name, product details, price, quantity, and total amount due.

4. Payment Management: The app will allow for the management of payments received, pending payments, and payment reminders for overdue invoices.

5. Reporting and Analytics: The app will also provide reporting and analytics features that will allow the business owner to track sales, revenue, and other relevant metrics to make informed decisions.

6. Integration with Payment Gateways: The app will integrate with popular payment gateways to facilitate easy and secure payments.

Overall, the billing app will be a comprehensive solution to help businesses manage their billing and invoicing needs efficiently.

1. **SYSTEM TESTING AND IMPLEMENTATION**

System testing is a process of testing an entire system or software application in order to evaluate its overall compliance to specified functional and non-functional requirements. Implementation is the process of delivering the software product/system to the end-users or clients and making sure that it meets their requirements and is functional.

**5.1SYSTEM TESTING**

During system testing, various types of testing are performed, including functional testing, performance testing, security testing, usability testing, and so on. The purpose of system testing is to ensure that the software application/system is error-free, meets customer requirements, and performs optimally under different conditions.

Once the software application/system has been tested and verified to meet the requirements, it is implemented. Implementation involves transferring the system from the development team to the end-users. This may involve training the end-users, ensuring that the software is deployed correctly, and addressing any issues that may arise during the implementation process.

The implementation process can be complex and involves many steps, including:

1. Planning the implementation process

2. Preparing the system for implementation

3. Deploying the system to the production environment

4. Testing the system in the production environment

5. Resolving any issues that arise during implementation

In conclusion, system testing and implementation are critical stages in the software development lifecycle. They ensure that the software application/system is error-free, meets customer requirements, and performs optimally. A successful implementation ensures that the end-users get the expected benefits from the software application/system.

**5.2TESTING TYPES**

1. Functional Testing: Testing the individual functions of the billing application to ensure they work as expected. This includes testing of billing process, invoice generation, payment processing, and account management.

2. Integration Testing: Testing the interaction between different modules and components of the billing application to ensure they work together seamlessly. This includes testing of database integration, third-party payment gateway integration, and communication between system components.

3. Performance Testing: Testing the billing application's ability to function under varying workload conditions, ensuring systems are able to handle expected traffic loads and still maintain acceptable response times.

4. User Acceptance Testing: Testing the billing application to ensure it meets the needs and requirements of the end users. This includes testing the user interface, user experience, and overall usability of the system.

5. Security Testing: Testing the billing application to ensure it is secure from all risks such as unauthorized access, data breaches, and other security concerns. This includes testing of authentication and authorization features, encryption of data in transit and data at rest, and third-party security compliance.

6. Regression Testing: Testing the billing application each time a new feature or functionality is added, updated, or modified to ensure that the new changes do not affect existing functionality.

7. Database Testing: Testing the database behind the billing application to ensure data integrity, consistency, and accuracy of data input and output.

8. Implementation Testing: Testing the billing application during and after implementation to ensure all features and functionalities are working as intended. This includes testing that data has been migrated accurately, user accounts are set up correctly, and system configurations are correct.

**5.3 SYSTEM IMPLEMENTATION**

1. UNIT TESTING: This involves testing each component of the billing application individually to ensure that it works as expected. This includes testing database connections, data input validation, data storage and retrieval, and other basic functionality.

2. INTEGRATION TESTING: This involves testing how the different components of the billing application work together. It focuses on the interactions between multiple components such as the website, database server, and billing system.

3. ACCEPTANCE TESTING: This involves testing the entire billing application as a whole to ensure that it meets the required specifications and performs as expected. This testing is conducted by end-users to ensure that the system meets their needs and operates in accordance with their requirements.

4. PERFORMANCE TESTING: This involves testing the system's performance and response time under varying loads of traffic to determine if it can handle large amounts of data and if there are any bottlenecks that need to be addressed.

5. SECURITY TESTING: This involves testing the security of the system to ensure that it is protected from unauthorized access, hacking and other breaches. This includes testing for encryption and other security measures that protect sensitive data.

6. USER ACCEPTANCE TESTING: This testing involves getting feedback from end-users on the ease of use, functionality, and experience of using the billing application. This feedback can be used to make improvements and updates to the system to enhance its user experience.

Once testing is complete, the billing application can be implemented into production. This involves deploying the application to the servers and ensuring that it is ready to support real-world transactions. The system should be monitored closely after implementation to ensure that it operates as expected and data is being processed correctly. Any issues that arise during this period should be addressed as quickly as possible to minimize any potential impact on the business.

1. **SYSTEM MAINTENANCE**

System maintenance for the billing application involves regular checks and updates to ensure its smooth and efficient operation. Here are some tasks that could be part of the maintenance process:

1. Back up the data: Regular backup of data is crucial to ensure that there is no data loss due to system crashes or other unforeseen circumstances.

2. Update software and patches: It's important to keep the billing application software up to date with the latest patches and upgrades to ensure that it remains secure and operates efficiently.

3. Monitor system performance: Monitoring system performance helps to detect any potential issues with the application and take corrective measures.

4. Check for errors: Regularly checking for errors within the billing application helps to identify issues that could compromise the accuracy of the billing process.

5. Restart the system: Restarting the system periodically can help to clear any issues that may arise due to prolonged usage.

6. Audit the system: It's essential to conduct regular audits to ensure that the billing application is functioning correctly and meeting the requirements of the organization.

By carrying out these maintenance tasks regularly, the billing application remains fully functional, provides accurate billing information, and minimizes downtime.

1. **CONCLUSION**

In conclusion, a billing application is the ideal solution for businesses that want to streamline their billing processes and improve their cash flow. With its user-friendly interface, robust features, and ability to automate billing processes, it can save time, reduce errors, and increase accuracy. Whether you're a small business owner, a freelancer, or a large enterprise, a billing application can simplify invoicing and payment collection, making it easier to manage your finances and improve your bottom line. By investing in a billing application, you're empowering your business to grow and succeed in today's competitive marketplace.

1. **APPENDICES**

1. Appendix A: Bill Format Template

This appendix includes a sample bill format template that the billing application can use to generate bills for customers. The template includes sections for customer information, billing period, itemized charges, and payment information.

2. Appendix B: Customer Information Form

This appendix includes a sample customer information form that the billing application can use to collect and store customer information. The form includes fields for customer name, address, phone number, and email address.

3. Appendix C: Payment Information Form

This appendix includes a sample payment information form that the billing application can use to collect and store payment information. The form includes fields for payment method, credit card information, and billing address.

4. Appendix D: Data Backup and Recovery Plan

This appendix outlines the data backup and recovery plan for the billing application. It includes details on how often data should be backed up, where backups should be stored, and how data should be recovered in the event of a system failure.

5. Appendix E: User Manual

This appendix includes a user manual for the billing application. The manual includes step-by-step instructions on how to use the application, including how to create bills, manage customer information, and process payments.

6. Appendix F: Training Materials

This appendix includes training materials for the billing application, such as instructional videos or guides. The materials can be used to train new users or to provide ongoing support for existing users.

7. Appendix G: Technical Support Information

This appendix includes technical support information for the billing application. It includes contact information for technical support staff, as well as details on how to report a bug or request a feature.

**8.1 BIBLIOGRAPHY**

**REFERENCEBOOKS**

**1. "Medical Billing and Coding Demystified" by Marilyn Burgos**

**2. "Medical Billing: The Bottom Line Revised 4th Edition" by Cynthia Stewart**

**3. "The Complete Idiot's Guide to Medical Billing and Coding" by Cynthia Stewart**

**4. "Understanding Health Insurance: A Guide to Billing and Reimbursement" by Michelle A. Green and JoAnn C. Rowell**

**5. "Medical Billing Certification Exam" by The National Healthcareer Association (NHA)**

**6. "ICD-10-CM Expert for Physicians 2022: The Complete Official Codebook" by American Medical Association**

**7. "CPT Professional 2022 (CPT / Current Procedural Terminology (Professional Edition))" by American Medical Association**

**8. "Healthcare Payment Systems: An Introduction" by Duane C. Abbey**

**9. "Guide to Health Claims Examining" by ICDC Publishing Inc.**

**10. "Medical Billing and Coding For Dummies" by Karen Smiley.**

**8.2 SAMPLE CODING**

**ADD FORM :-**

import React, { useState } from 'react';

import './AddForm.css'

function AddFrom({handleList}) {

    const [itemName, setItemName] = useState('')

    const [quantity, setQuantity] = useState('')

    const [price, setPrice] = useState('')

    const [name, setName] = useState('')

    const event = new Date();

    const date = event.toLocaleDateString()

    const time = event.toLocaleTimeString()

    // console.log(event)

    const handleChange = (e) => {

        const { name, value } = e.target

        if (name === 'ItemName') {

            setItemName(value)

        } else if (name === 'quantity') {

            setQuantity(value)

        } else if (name === 'price') {

            setPrice(value)

        }else if (name === 'name') {

            setName(value)

        }

    }

    const handleAdd = (e) => {

        e.preventDefault()

        handleList({ itemName, quantity, price})

        setItemName('')

        setQuantity('')

        setPrice('')

        setName('')

    }

    return (

        <form

        className='addform\_container'

        onSubmit={(e) => handleAdd(e)}

        >

            <div className='addform\_container\_two'>

            <input

                type='text'

                name='ItemName'

                className='addform\_input'

                placeholder='item Name'

                value={itemName}

                onChange={(e) => handleChange(e)}

                required

            />

            <input

                type='text'

                name='quantity'

                className='addform\_input'

                placeholder='Quantity'

                value={quantity}

                onChange={(e) => handleChange(e)}

                required

            />

            <input

                type='text'

                name='price'

                className='addform\_input'

                placeholder='Price'

                value={price}

                onChange={(e) => handleChange(e)}

                required

            />

            <button

                type='submit'

                className=' btn'

            >

                Add

            </button>

            </div>

        </form>

    )

}

export default AddFrom

{/\* <input

type='text'

name='name'

className='addform\_input'

placeholder='Customer Name'

value={name}

onChange={(e) => handleChange(e)}

/> \*/}

**LIST FORM :-**

import { render } from '@testing-library/react';

import React, { useState, useEffect } from 'react';

import { useDispatch } from 'react-redux';

import { AddNewList } from '../../Features/ListSlice';

import './ListForm.css';

import { fetchListData } from '../../Features/ListSlice';

function ListForm({ newList, handleDelete }) {

  const [list, setList] = useState([])

  const [tot, setTot] = useState(-1)

  // const [sendList, setSendList] = useState([])

  const dispatch = useDispatch()

  useEffect(() => {

    setList(newList)

  }, [newList, list]);

  // console.log(tot)

  useEffect(() => {

    setTot(-1)

    list.map((item, index) => {

      (list.length == 4 ? setTot(Number(item.price)) : setTot(tot + Number(item.price)))

    })

  }, [list]);

  // console.log(list)

  const total = <tbody>

    <tr>

      <td colSpan={2}>TOTAL</td>

      <td>{tot}</td>

    </tr>

  </tbody>

  const renderList = list.map((item, index) => {

    return (

      index > 2 ?

        <tbody key={index}>

          <tr>

            <td>{item.itemName}</td>

            <td>{item.quantity}</td>

            <td>{item.price}</td>

          </tr>

        </tbody>

        :

        null

    )

  })

  // console.log("jhgjh" ,sendList)

  const handlePrint = () => {

     const sendList = [tot, ...list]

      dispatch(AddNewList(sendList))

      // console.log("jhgjh", sendList)

    handleDelete()

  }

  return (

    <div className="ccontainer">

      <table className="table ">

        <thead className="table\_header">

          <tr>

            <th>Name</th>

            <th>Quantity</th>

            <th>Price</th>

          </tr>

        </thead>

        {(list.length > 1 ? renderList :

          <td colSpan={3}>Please add items & customer Name</td>)}

        {(list.length > 1 ? total : null)}

      </table>

      <div className='btn-container'>

        <button className='btn ' onClick={handlePrint}>Print</button>

        <button className='btn' onClick={handleDelete}>Delete</button>

      </div>

    </div>

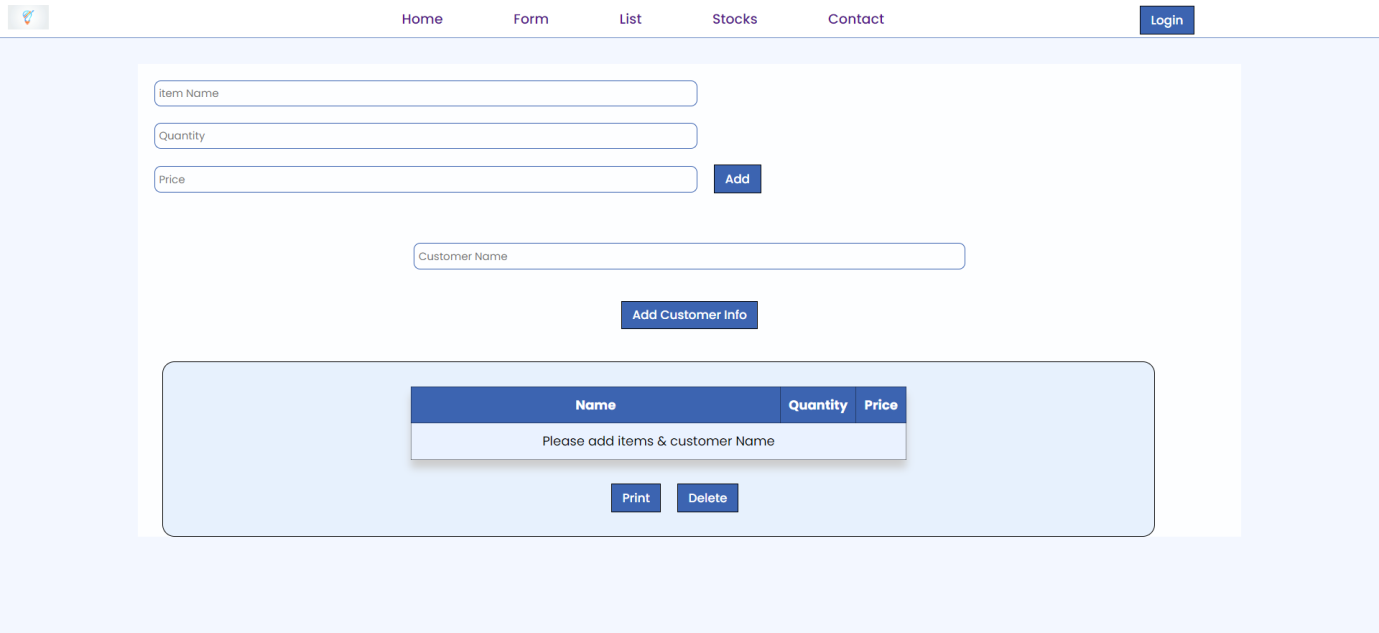
  )

}

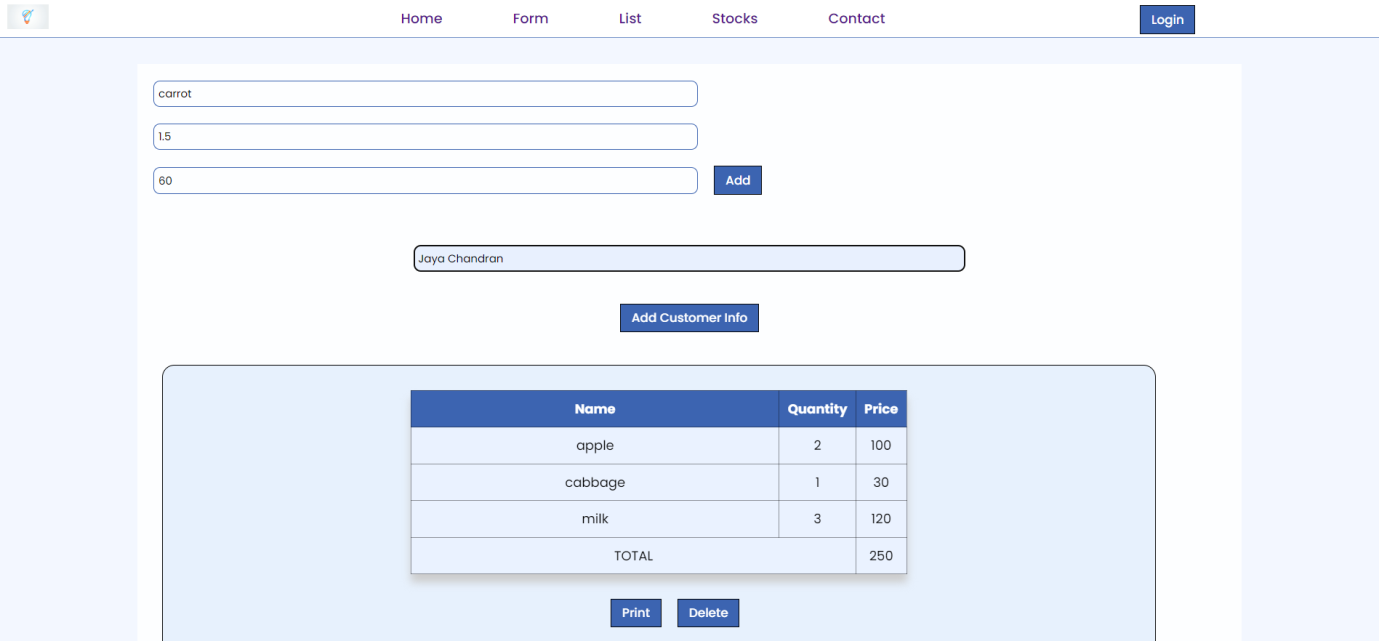
export default ListForm

**8.3 SAMPLE INPUT**

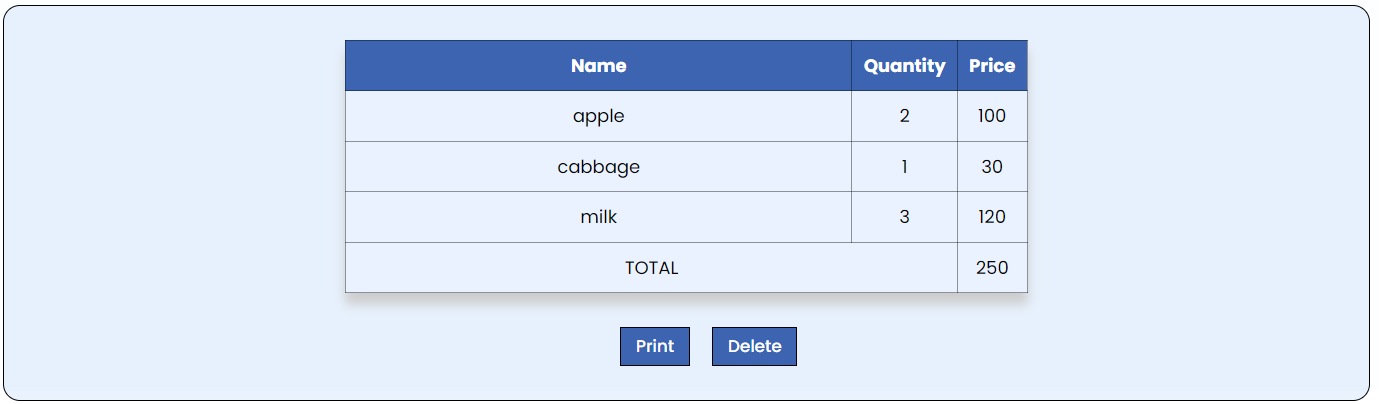
**INITIAL WINDOW :-**

****

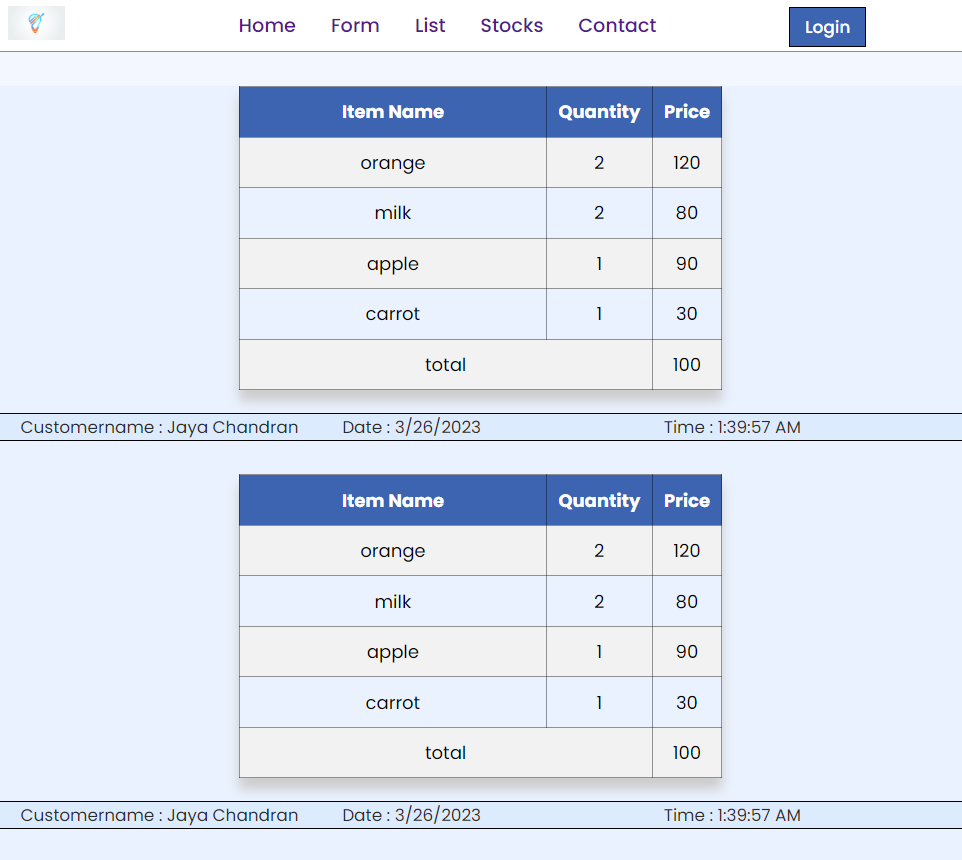
**SAMPLE INPUT: -**

****

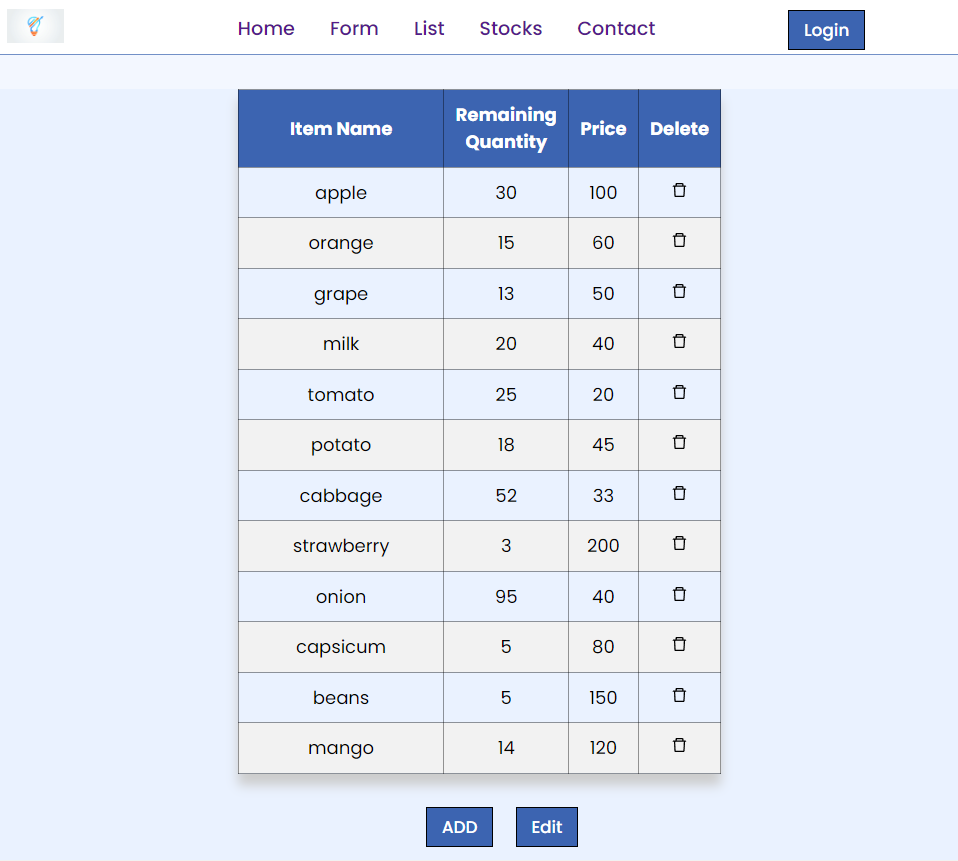
**INVOICE :-**

****

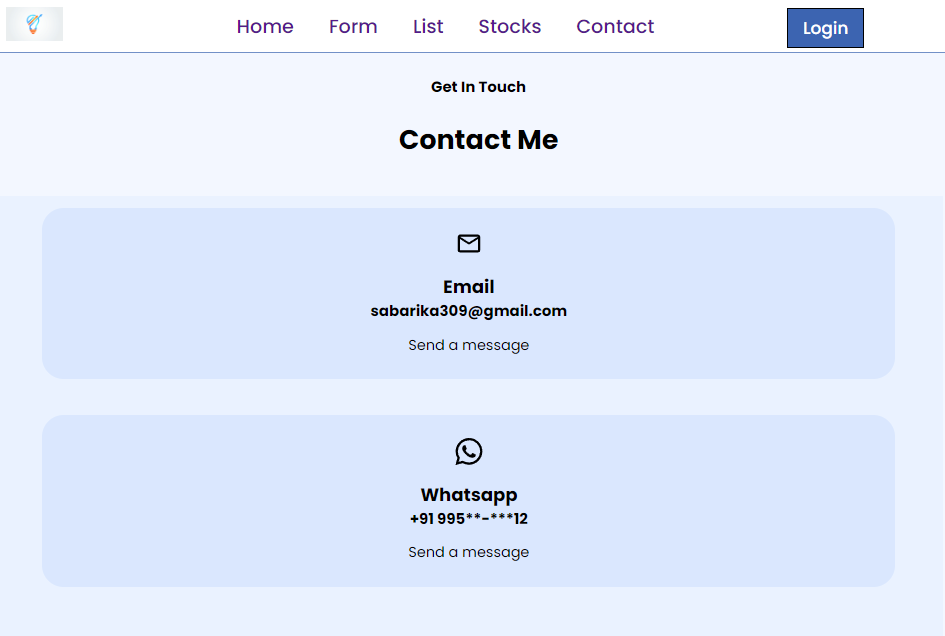
**INVOICE HISTORY :-**

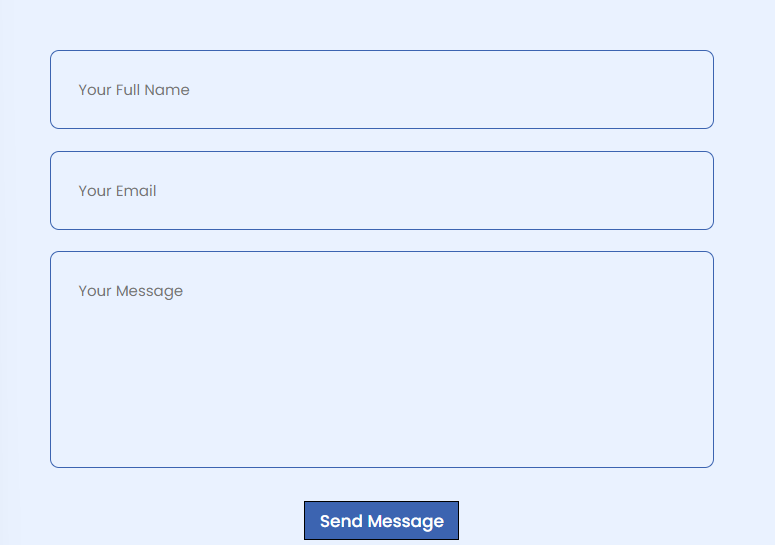
****

**STOCK REPORT :-**

****

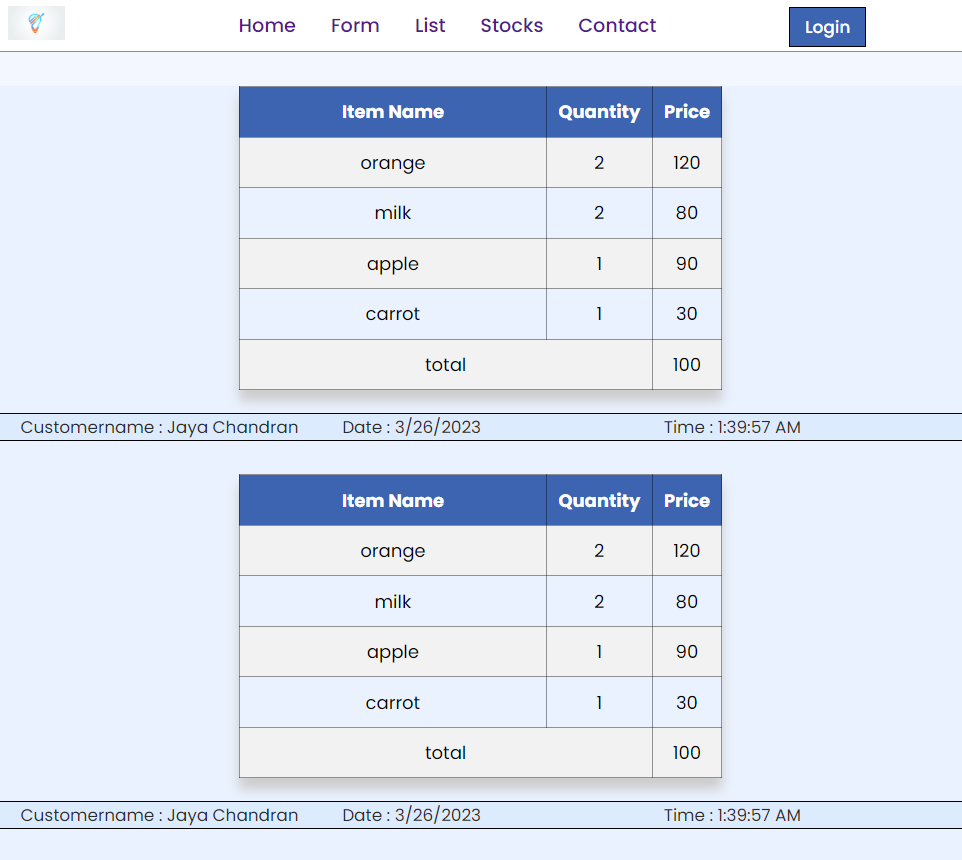
**CONTACT SECTION :-**

****

****

**8.4SAMPLE OUTPUT**

**INVOICE HISTORY :-**

****