INDIRA GANDHI COLLEGE OF ARTS AND SCIENCE

INDIRA NAGAR, PUDUCHERRY

(AFFILIATED TO PONDICHERRY UNIVERSITY)

DEPARTMENT OF COMPUTER SCIENCE

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF THE DEGREE

B.C.A

(BACHELOR OF COMPUTER APPLICATIONS)

BILLING SYSTEM FOR GROCERY SHOP

Guided by

Dr. C. S. Rajarajeswari, M.C.A., M.Phil., Ph.D.,

Assistant Professor

Department of Computer Science

Submitted by

ANTONI RAJ. K	20CA0502
ARIVAZHAGAN. S	20CA0503
JANARTHANAN. K	20CA0515

2020-2023 Batch

INDIRA GANDHI COLLEGE OF ARTS AND SCIENCE

INDIRA NAGAR, PUDUCHERRY

(AFFILIATED TO PONDICHERRY UNIVERSITY)

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this project work entitled **"BILLING SYSTEM FOR GROCERY SHOP"** was done by

ANTONI RAJ. K	20CA0502
ARIVAZHAGAN. S	20CA0503
JANARTHANAN. K	20CA0515

of VI Semester, B.C.A, in partial fulfillment of the requirement for the B.C.A Degree during the period 2020-2023, is the original work done by the candidates.

Project Guide **Head of the Department**

Submitted for the external VIVA-VOCE examination held on …….

Internal Examiner **External Examiner**

ACKNOWLEDGMENT

We thank the almighty God for open-handed us the opportunity for doing the project successfully.

It's our duty to express our sincere thanks to **Dr. KISHORE K JOHN, Ph.D., Principal,** Indira Gandhi College of Arts and Science, Puducherry for extending support to complete our project.

We would like to sincerely thank **Dr. C. S. RAJARAJESWARI, M.C.A., M.Phil., Ph.D., Head of the Department,** Department of Computer Science for the encouragement given by her while doing the project.

We would like to thank our guide **Dr. C. S. RAJARAJESWARI, M.C.A., M.Phil., Ph.D.,** Assistant Professor in Computer Science for this encouragement and valuable suggestion throughout the study phase, implementation phase, and preparation of the project.

We would like to express our sincere thanks to all our **STAFF MEMBERS** of the Department of Computer Science, Indira Gandhi College of Arts and Science. We also like to thank **Mrs. Ajapa Gayathri**, Department of Computer Science for her technical support and valuable ideas.

We take immense pleasure in conveying sincere thanks to the owner of **JAI GROCERY SHOP** for providing us the opportunity to do project work in this esteemed organization and employees for their critical comments and useful suggestions to complete this project.

Last but not least we would like to thank our beloved parents for all the understanding and the support extended to us, without which this achievement would not be implemented. We feel very proud of our cooperation and team spirit throughout this project.

CONTENTS

ABSTRACT

CHAPTER-1

ABSTRACT

This project is designed and developed for grocery store. The project is named as "BILLING SYSTEM FOR GROCERY SHOP".

This project is developed with VB.NET as a front end and SQL as the back end tool.

A billing system is a software application that helps the business to manage their financial transactions and billing process.

The system streamline the billing process by generating invoices, recording payment and tracking of outstanding balances. This abstract describes the key features and benefits of a billing system, including its ability to increase accuracy and efficiency reduce manual errors, improve cash flow, provide valuable insights into business finances. The abstract also covers some common billing system functionalities such as automated billing, payment processing and reporting as well as the importance of data and compliance in the billing process. Overall, a billing system is an essential tool for any business that wants to streamline its functional operations and improve its bottom line.The grocery billing system is an automated system that aims to simplify the billing process of grocery shop.

This system includes a user-friendly interface that enables to shop staff to enter customer information, scan products and generates bills quickly. The system also maintains a database of products and their prices, which can be easily updated. Additionally the system generates various reports such as sales reports and inventory reports, helping shop owner to make informed business decision.

This project deals with the five modules. Each module is explained below.

1. Product
2. Sales
3. Supplier
4. Home delivery
5. Return Stock
6. Report

STOCK:

The overall stock details of the shop is stored in a database each one with the unique item code number. Each item is separate by their product equality on hand is specified.

SALES:

Billing feature of the details about the total ornaments purchased can be included in it. This module include the purchase of product of the stock level whenever purchased.

SUPPLIER:

Order placing facility is also available in the project in case of same bulk order to the supplier details be collected and the type of products and advance amount will be saved in the database and the order will be saved.

HOME DELIVERY:

Regular customer details (for e.g. customer name, address, phone number) will be stored in database for the purpose of home delivery of products to the customer in this project.

RETURN STOCK:

Return of the damaged products from the customer will be stored in the database. This module include the return of damaged products of the stock level whenever returned.

REPORT:

Several reports are generated in this section. For each model one report is generated. Based on several requirement finally reports are generated.

PROBLEM DEFINITION AND FEASIBILITY ANALYSIS

CHAPTER-2

PROBLEM DEFINITION AND FEASIBILITY ANALYSIS

2.1 INTRODUCTION

The problem definition and feasibility analysis is an important phase in the development of the project. Problem identification influences the whole project and must be processed in an orderly fashion.

For the creation of a new system the developers must have to study the drawback of the existing system then only then the developer can overcome the drawbacks of the existing system.

2.2 PROBLEM DEFINITION

The major problems identified in the existing system are

- Data maintenance is a manual process
- Time consuming process
- Searching for particular item by manual process
- Data verification is a tedious work
- Stock availability can be checked

So it has been decided to develop a computerized system for grocery shop in order to eliminate these problems. The product system contains various modules for stock and saving, searching for the availability of stock under various criteria and generation of various reports based on requirements.

2.3 EXISTING SYSTEM

In the shop the details of the stock, saving accounts and staff are recorded in the register. All the information are stored and processed. If any details are needed then the management will look into the written records and search for the particular details and get the details.

In the existing system

- The processing time is more and stationary need is also more.
- Since it is manual, accuracy cannot be guaranteed
- There is possibility for duplication of entries which may lead to error
- Maintenance of records for future reference is also very difficult process.
- The records in volumes and files need to be protected for long time and thus occupy more space.

2.4 PROPOSED SYSTEM

To overcome the proposed system is designed after thorough understanding of the existing system and drawbacks of manual operation.The proposed system contains five modules namely stock, savings, billing, ordering of the people.

2.4.1 FEATURES OF THE PROPOSED SYSTEM

The proposed system consists of the following general features.

- Very interactive

- Simple and easy to understand
- Symbolic representation of concepts
- Advanced search option
- Generation of required reports based on user requirements

2.5 FEASIBILITY STUDY

The feasibility study is basically the test of the proposed system in the light of its workability, meeting user requirements, effective use of resource and the cost system. The objective of feasibility is not to solves the problems but to acquire an idea of its scope. In the first step, we have prepared the full details about users demonstrate needs, resource requirements, problem can refined.

The system is to be considered as feasible only if the proposed system is useful and it is determine at the preliminary stage. Thus purposed system is to gather, analyze and documents the data needed to make an informal intelligent decision regarding a system practically.

The feasibility study concern with the consideration made to verify whether the system is fit to be developing in all terms. The main goal of feasibility study is the cost and benefits are estimated with the greater accuracy.

The primary objectives of feasibility study are to weigh up two types of feasibility

- Operational feasibility
- Technical feasibility

2.5.1 Operation Feasibility

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The system consider for development has undergone for the operation feasibility study, which ensure that the system is implemented within the end users the students. Thus the projects is operation feasible.

.

2.5.2 Technical Feasibility

A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements.

Technical feasibility analysis make a comparison between the levels of technology available and the technology is determined by the factor such as the software tools, platform, etc. Since the resources required for the development of this project is already available with the operating system, this project technically feasible.

2.5.3 Economic Feasibility

Economic analysis could also be referred to as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

Economic feasibility involves analyzing the costs and benefits of a project, including the costs of materials, labor, and equipment, as well as the projected revenue from sales or other sources of income. Economic feasibility is an important consideration when determining whether a project or venture should be undertaken, and it is often used in conjunction with other types of feasibility analysis, such as technical feasibility and operational feasibility.

SOFTWARE REQUIREMENTS SPECIFICATIONS

CHAPTER-3

SOFTWARE REQUIREMENTS SPECIFICATIONS

Software requirements specifications are primarily concerned with user and external view of the software products with technical specification of requirements for the software product. The goal of software requirements is to completely and consistently specify the technical requirements of the software product. These general description of software requirements specification require and summarize the processing environment for development operation and maintenance.

This project overviews the common supportive platform for the users as well as the developer. This project doesn't require special feature rather than normal architecture. The minimum system configuration needed for developing and using the proposed system has been listed below.

3.1 HARDWARE REQUIREMENTS

Processor: 11th Gen Intel Core i5-11320H

RAM: 16 GB

SDD: 512GB

Monitor: Lenovo IdeaPad

Keyboard: Backlit Keyboard

3.2 SOFTWARE REQUIREMENTS

Front End Tool : Visual Basic .NET

Back End Tool : SQL Server

SYSTEM DESIGN

CHAPTER-4

SYSTEM DESIGN

4.1 INTRODUCTION

Software design phase deals with moving from the problem domain of the system into conceiving, planning out and specifying the externally observable characteristics of software products. So in the design phase, the criteria involved in the design of the various module of "Billing System For Grocery Shop" are analyzed. The interconnection among function, data structures and packing of the software product. Design phase consists of three phase.

- External design
- Architectural design
- Detailed design

4.1.2 EXTERNAL DESIGN

External design involves conceiving, planning out and specifying the externally observable characteristics of the system. In practice, it is not possible to perform requirements definition without doing some preliminary design. So for "product automation system" we have design our front end in many ways. The user interface design for this system would be neat, pleasant and aesthetic. So we decided to use visual Studio 2022 as the front end design tool because coding style is not very complicated, graphical user interface and navigation design can be implemented in an elegant manner.

4.1.3 ARCHITECTURAL DESIGN

Architectural design concerns about describing the overall feature of the software, defining the requirements and establishing the high level view of the system. During architectural design, the software components in the project are identified and decomposed into processing module. The architectural design of the project is established through data flow diagram and system chart.

4.2 MODULE DESIGN

This proposed system, “BILLING SYSTEM FOR GROCERY SHOP” is designed to satisfy the end user of the organization. The modules need to be constructed in a well defined manner. The different modules identified are follows:

- Product entry
- Sales entry
- Supplier entry
- Home Delivery entry
- Return product entry
- Report

This phase concentrate on the algorithmic description of each module for implementation. The data requirement of each module is identified and the existing phases are correlated in the module. design. The data requirement of each module is identified and the algorithm processing activities is conceived out.

Product

The product module maintains the details of the products in the agency. The product details are product id, product name, product type, quantity and price. When a new product enters into the showroom the product id is allocated to the product by the agency. This will be the unique identification number for the particular product. In the show room there may be different titles from different companies and each has different color and different types.

Sales

When the product gets sold by the shop, then we can go for sales modules this sales modules maintains the details of the sales done by the shop and also will calculate the amount for the particular sales if any discount occurs.In the shop there may have different customer for different product. All the customer's personal details are collected here because of contacting those customer in future. Customer module contains customer number, customer name, customer address, and phone number. With this, which brand purifier by the customer and what purifier is that type will also be stored here.

Supplier

This module helps the shop to get the required products details for the business. This module maintains the details supplier details are supplier name, supplier id, supplier address and his stocks. It also generates reports for send required products.

Home Delivery

This module is where you go to add update and delete information about your customer. You can enter and edit their physical customer name, address, phone number and mail id. It's useful module to visit whenever you need to drill into a customer's history and check other information about them for home delivery for damaged products.

Return Product

This module is responsible for the damaged product of the customer who brought the product from the shop. It stores and maintains the information in the database for the damaged product.

Reports

This module generates different reports are product reports, sales reports, supplier reports, customer details reports for the end user in order to increase the efficiency of billing system.

4.2.1 Reports

This phase concentrates on identifying the various reports that are require by the end user. The major reports identified are.

- Supplier details reports
- Products details reports
- Customer details reports
- Stock details report
- Bill details reports

4.2.1.1 Stock report

Stock reports to the type of them and the total quantity which is on hand is generated.

4.2.1.2 Ordering report

Order report is generated based on the delivery dates at which the customer is ordered.

4.2.1.3 Billing report

ill report based on the quantity amount purchased according to the particular dates and the total sunt for which the quantity is purchased.

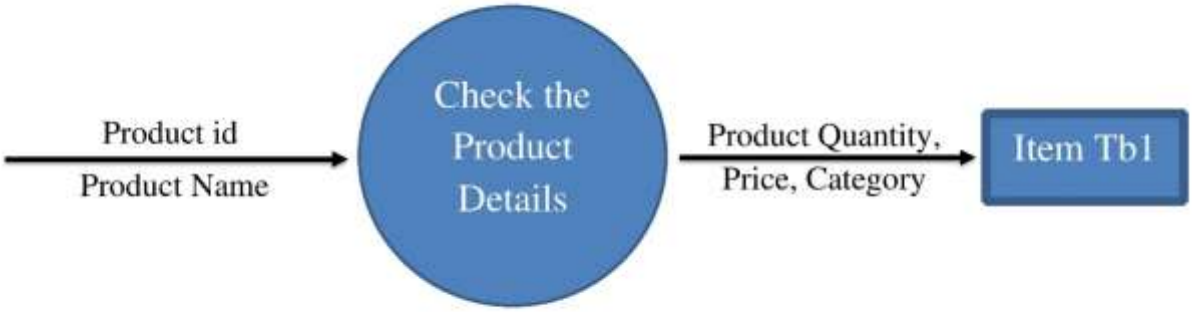
4.2.1.4 Savings report

Saving account member hot based on the quantity of product owned and the total amount will be listed.

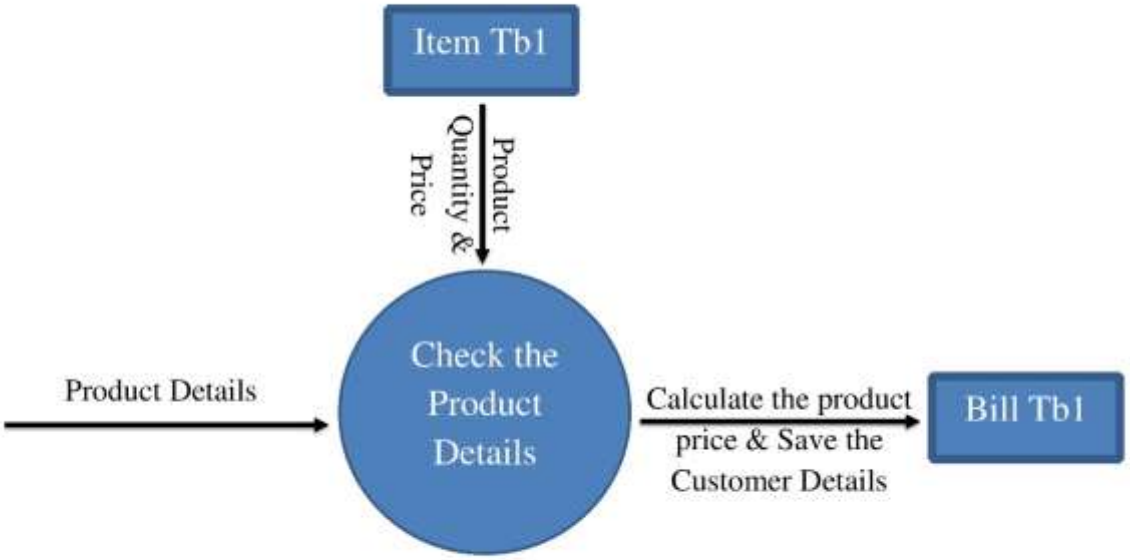
4.3 DATA FLOW DIAGRAM FOR BILLING SYSTEM FOR GROCERY SHOP

4.3.1 BSGS

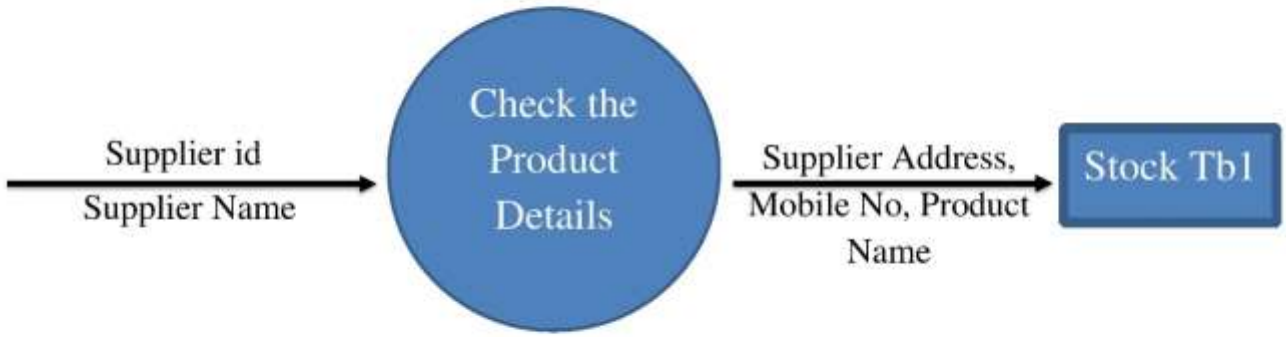
Product Details:



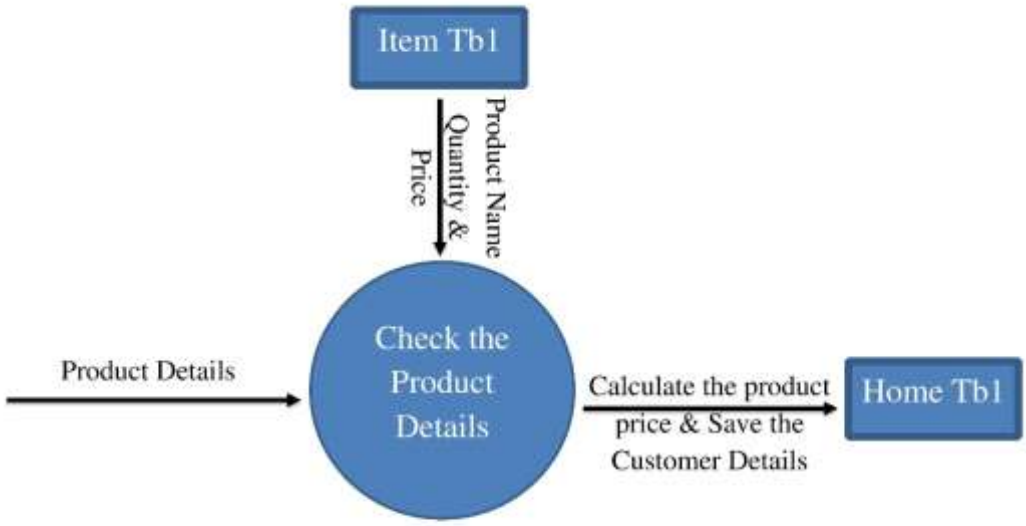
Bill details:



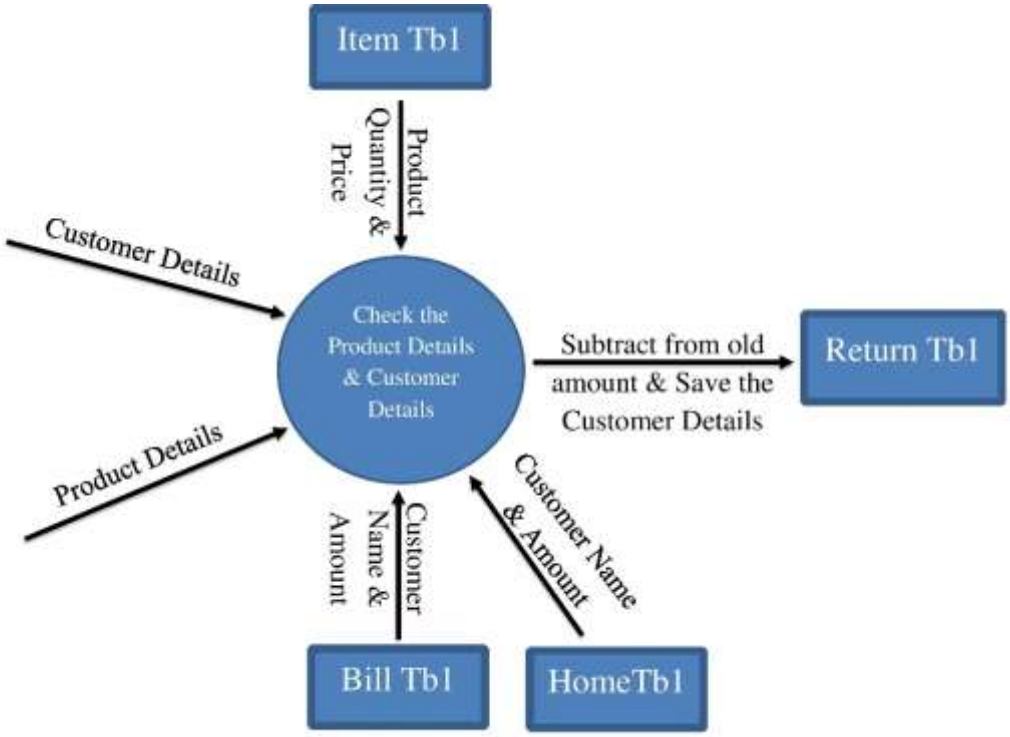
Supplier Details:



Home Delivery:



Return Product:



4.4 DATA DICTIONARY

A data dictionary contains a list of all list files in the database, the number of records in each file, and the names and type of each field. Most database management system deeps the data dictionary hidden from users to prevent them from accidentally destroying its contents.

Data dictionary do not contain any actual data from the database, only book keeping information for managing it. Without a data dictionary, however a database management system cannot access data from the database.

A well-developed data dictionary should be able to provide the following information:

- How many characters are there in a data item?
- By what names it is referred in the system.
- Where it is used in the system.

It is very important to update the data dictionary as changes occur. It is developed during the analysis, however its contents are used during system design as well.

Data dictionary and organized in of all data element that is pertinent to the system, with precise, rigorous definitions so that both user and system analyst will have a common understanding of input, output components of stores and even intermediate calculations.

Data Dictionary contains the following information:

Name: The primary name of the data or control item, the data store of an external entity

Alias: other names used for the first entry

Where-used/How-used: A listing of the processes that use that and control item and how it is used (e.g., input to the process, output from the process, as a store, as an external entity).

Content Description: A motion for representing content

Supplementary Information: Other information about data types, present values (if restrictions or limitations, and so forth known)

4.5 TABLE DESIGN

Database Name: departmentalstore

Table Name: ItemTb1

Table Description: Maintains the product details

Primary Key: Itid

SI.NO	FIELD NAME	DATA TYPE	DESCIRPTION
1	Itid	Int	It contains product id
2	ItName	Varchar(50)	It contains product name
3	ItQty	Int	It contains product quantity
4	ItPrice	Int	It contains product price
5	ItCat	Varchar(50)	It contains product category

Table Name: BillTb1

Table Description: Maintains the bill details

Primary Key: Bid

SI.NO	FIELD NAME	DATA TYPE	DESCIRPTION
1	Bid	Int	It contains bill number
2	ClientName	Varchar(50)	It contains client name
3	Amount	Int	It contains bill amount
4	BDate	Date	It contains the date of bill

Table Name: StockTb1

Table Description: Maintains the supplier details

Primary Key: StId

SI.NO	FIELD NAME	DATA TYPE	DESCIRPTION
1	StId	Int	It contains supply id
2	ComName	Varchar(50)	It contains supplier name
3	Mob	Int	It contains supplier phone number
4	Address	Varchar(50)	It contains supplier address
5	ProName	Varchar(50)	It contains product name

Table Name: ReturnTb1

Table Description: Maintains the return proudct details

Primary Key: Reid

SI.NO	FIELD NAME	DATA TYPE	DESCIRPTION
1	Reid	Int	It contains return productl id
2	ReCn	Varchar(50)	It contains client name who returns damaged product
3	ReAm	Int	It contains product bill amount
4	ReDate	Date	It contains the date of damaged product purchased

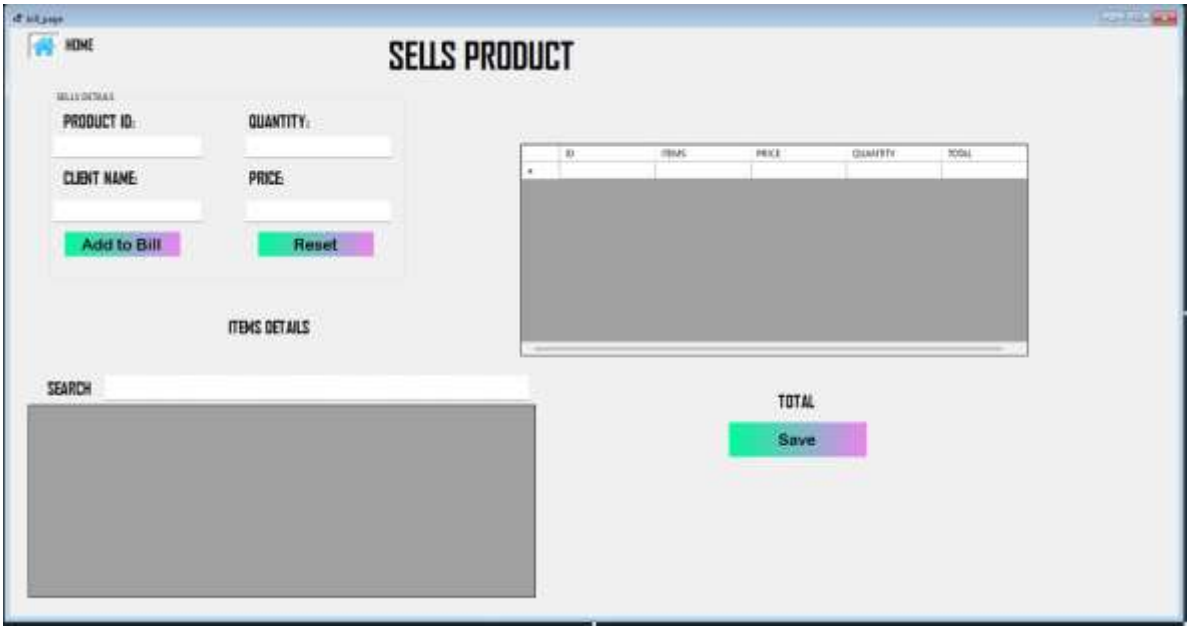
Table Name: HomeTb1

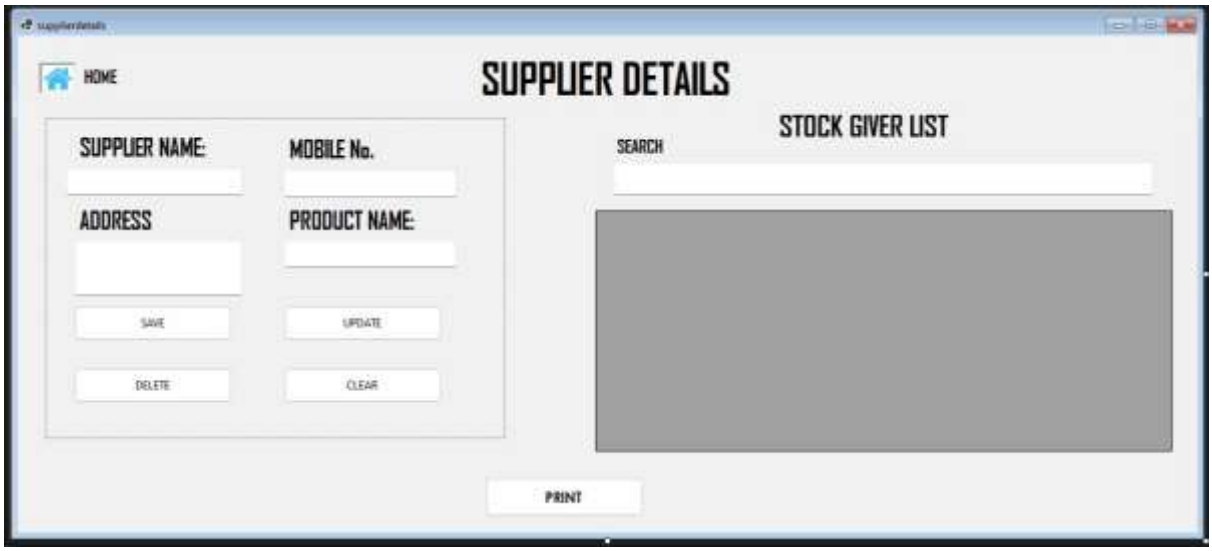
Table Description: Maintains the home delivery details

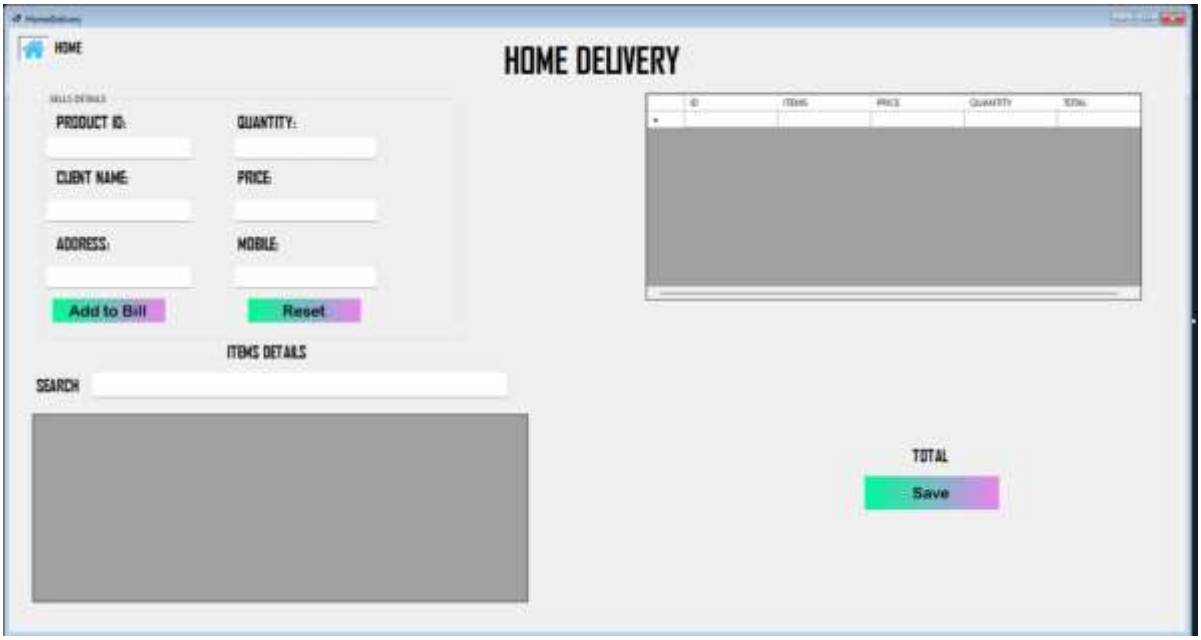
Primary Key: Hid

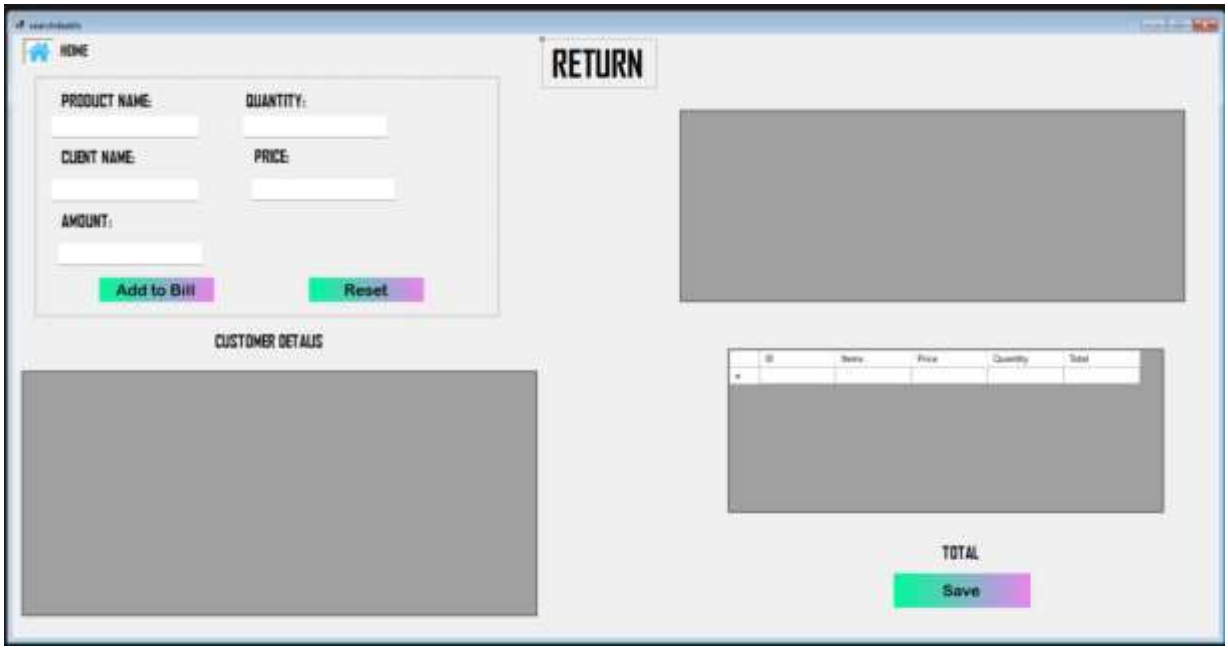
SI.NO	FIELD NAME	DATA TYPE	DESCIRPTION
1	Hid	Int	It contains home delivery id
2	homename	Varchar(50)	It contains client name who orders product
3	Homeamount	int	It contains product bill amount
4	hAddress	Varchar(50)	It contains the address of the home delivery
5	hmobile	Int	It contains the client phone number
6	hdate	Date	It contains the date of home delivery

4.6 FORM DESIGN


HOME
SELLS PRODUCT
PRODUCT ID:
QUANTITY:
CLIENT NAME:
PRICE:
Add to Bill
Reset
ITEMS DETAILS
SEARCH
TOTAL
Save

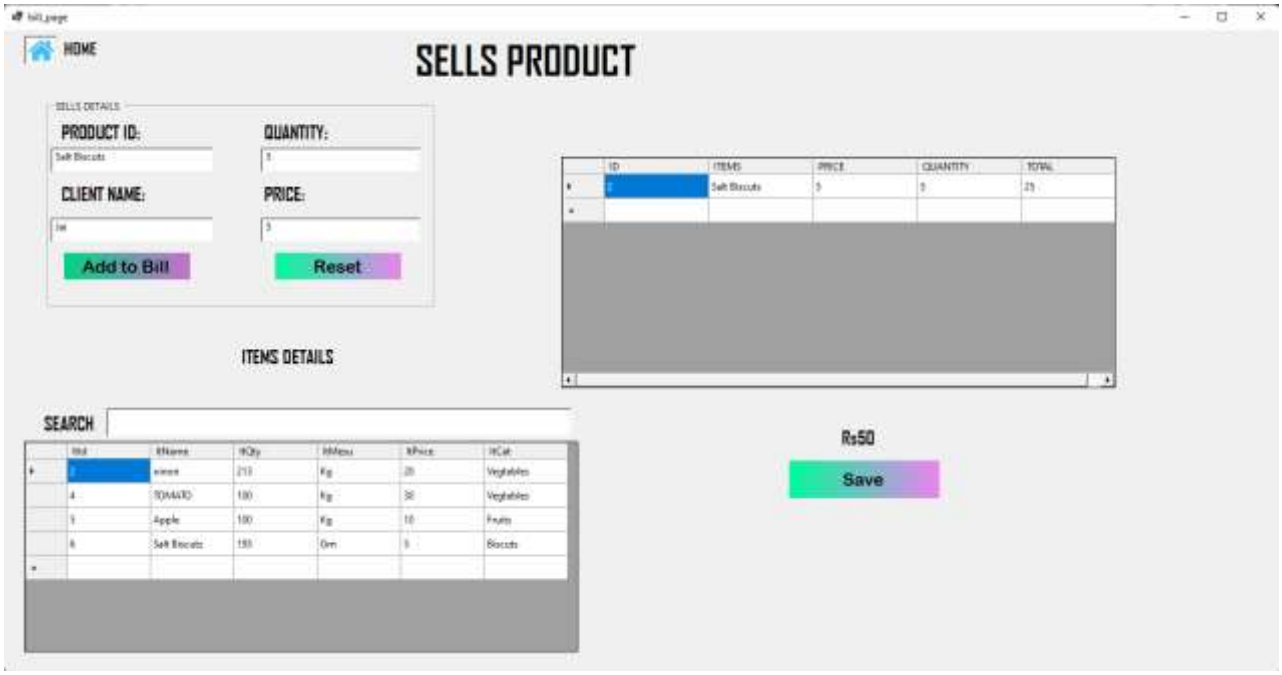

HOME
SUPPLIER DETAILS
SUPPLIER NAME:
MOBILE No.
ADDRESS
PRODUCT NAME:
SAVE
UPDATE
DELETE
CLEAR
STOCK GIVER LIST
SEARCH
PRINT

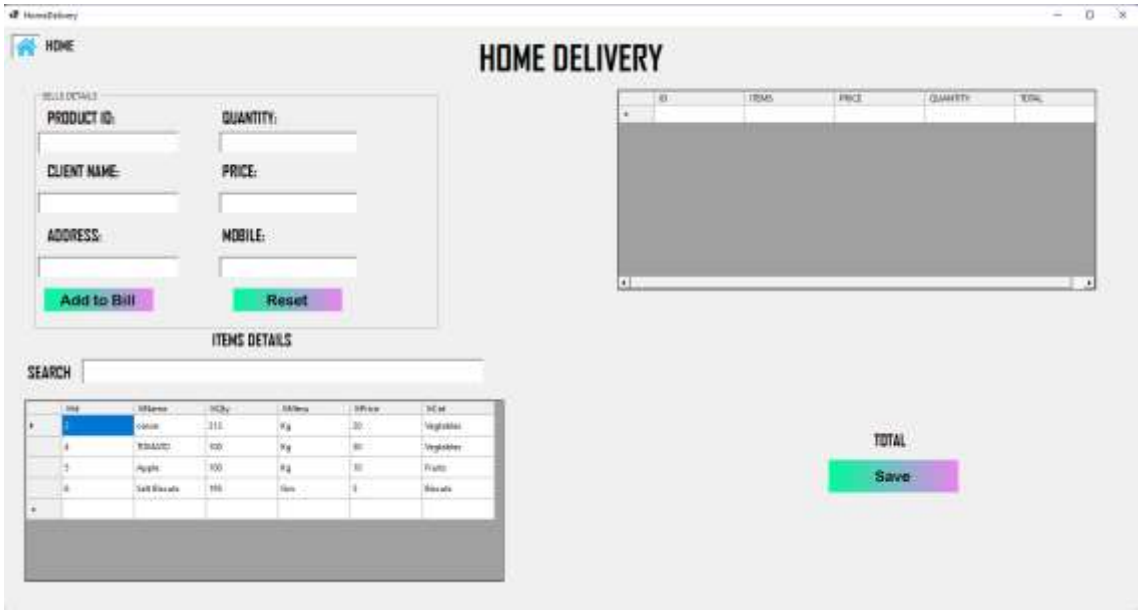

HOME
HOME DELIVERY
PRODUCT ID:
QUANTITY:
CLIENT NAME:
PRICE:
ADDRESS:
MOBILE:
Add to Bill
Reset
ITEMS DETAILS
SEARCH
TOTAL
Save

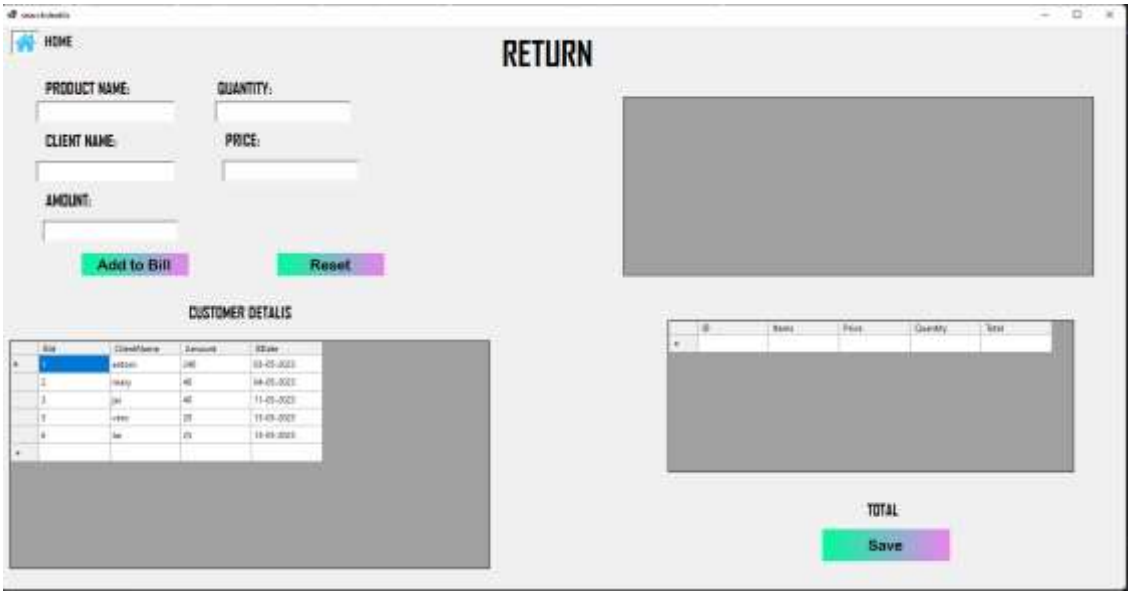

HOME
RETURN
PRODUCT NAME:
QUANTITY:
CLIENT NAME:
PRICE:
AMOUNT:
Add to Bill
Reset
CUSTOMER DETAILS
TOTAL
Save

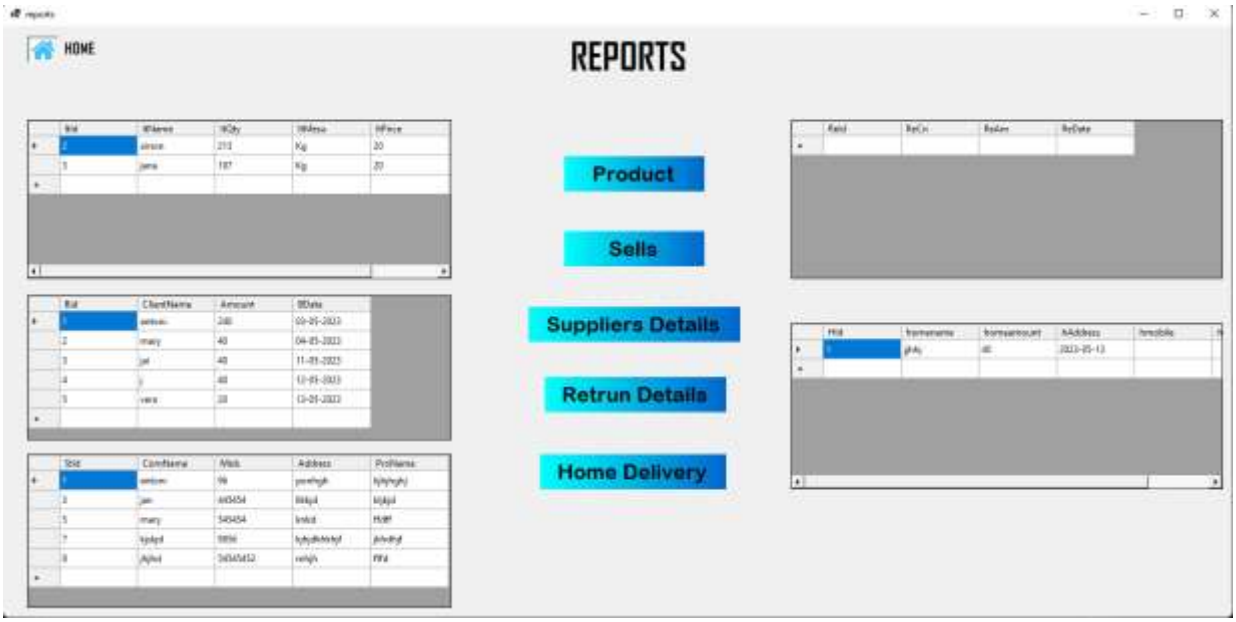

HOME
REPORTS
Product
Sells
Suppliers Details
Retrun Details
Home Delivery

4.7 OUTPUT DESIGN




HOME
HOME DELIVERY
PRODUCT ID:
QUANTITY:
CLIENT NAME:
PRICE:
ADDRESS:
MOBILE:
Add to Bill
Reset
ITEMS DETAILS
SEARCH
TOTAL
Save


HOME
RETURN
PRODUCT NAME:
QUANTITY:
CLIENT NAME:
PRICE:
AMOUNT:
Add to Bill
Reset
CUSTOMER DETAILS
TOTAL
Save


HOME
REPORTS
Product
Sells
Suppliers Details
Retrun Details
Home Delivery

CODING TESTING

AND

IMPLEMENTATIONS

CHAPTER-5

CODING TESTING AND IMPLEMENTATIONS

5.1 CODE DESCRIPTION

The coding stands are being proposed and user to improve the coding style and attach the information about the various and functions along with the various names. This is not only improves the styles but also gives a definitions way to be understood by others, other than the one who code it.

The programming style that is the coding style is depends upon the programmers. By program we mean its structured meaning and not its stylish presentation. A documents has no value if it does not convert its meaning. Documents are validated against a particular structural rule set and processing then documents we will get to know their exact structural against a specific rule set.

The code writing is critical on and it should be so clear and understanding to every programmer. A programmer should be able to understand and must be able to do correction on the existing code; if necessary certain stands are needed to achieve the above objective.

The standard followed by us are listed below

- Program should be simple, clear and easy to understand
- Naming conventions.
- Value conventions.

5.2 Testing

All the modules of the system are combined and are put to the operational use. This means the new and old system are run in parallel for some time, errors are identified and the corresponding are to be corrected to get the required output.

The set of working programs and initialized tables are also provided for easy start of the user. In addition, system documentation is also provided. All users have been trained to use the system.

5.2.1 Unit Testing

Unit testing comprises the set of tests performed by an individual programmer prior to integration of the unit into a large system. The situation illustrated as follows:

Coding & debugging unit testing integration testing

A program unit is usually small enough that the programmer who developed it can test it in great deal, and certainly in greater detail than will be possible when the unity is integrated into an evolving software product.

There are four categories of tests that a programmer will typically perform on a program unit.

- Functional tests
- Performance tests
- Stress tests
- Structure tests

5.2.1.1 Black Box Testing

Functional Test

Functional test involve exercising the code with nominal input values for which the expected results are known, as well as boundary value (minimum values, maximum values, and values on and just outside the functional boundaries).

Performance Test

Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, response time, and device utilization by the program unit.

Stress Test

Stress tests are those tests designed to intentionally break the unit. A great deal can be learned about the strength and limitation of a program by examining the manner in which a program unit breaks

5.2.1.2 White Box Testing

Structure Test

Structure tests are concerned with exercising the internal logical of a program and traversing Particular execution paths.

Test Cases

If a module is data coupled with another module, the unit testing is performed with the dummy values. For all the modules, confirmation to consistent user interface is mainly checked. Validations are done at each and every stage. Checks are made in each and every unit by giving invalid data the system is found to function successfully in all units. This test focused on each module individually, ensuring that if functions properly as a as a unit. Hence, named is unit testing.

5.2.2 Integration Testing

The integration strategy dictates the order in which modules mist be available, and thus exerts a strong influence on the order in which modules are written, debugged, and unit tested. Integration testing address the issues associated with the dual problems of verification and program construction. After the software has been integration a set of high order tests are conducted. The main objective is this testing process is toke unit tested modules and build a program structure while that has been dictated by the design.

Bottom-Up Testing

Bottom-up Integration Testing is a strategy in which the lower level modules are tested first. These tested modules are then further used to facilitate the testing of higher level modules. The process continues until all modules at top level are tested. Once the lower level modules are tested and integrated, then the next level of modules are formed.\

Top-Down Testing

Top Down Integration Testing is a method in which integration testing takes place from top to bottom following the control flow of software system. The higher level modules are tested first and then lower level modules are tested and integrated in order to check the software functionality. Stubs are used for testing if some modules are not ready.

5.2.3 Acceptance Testing

Acceptance testing involves planning and execution of functional tests; performance tests, and stress tests to verify that the implemented system satisfies its requirements.

5.3 Implementation

Program system implementation has been decided to use Visual Basic.Net 2005 as the front-end and MS Access as the back-end tool for implementation of the proposed System. The facilities provided in the development software improve the quality as well as the user friendliness of the system . we are intended to develop. In this stage the design plan is converted module by module on the program code.

The project.mdb is created on MS access which includes the table such as rates, savings, stock, order placing, staff details, billing. The type widths of the fields in these tables are set appropriately.

CONCLUSION

CHAPTER-6

CONCLUSION

As specified in the definition, all the Features that are required for "BILLING SYSTEM FOR GROCERY SHOP " are finished successfully. All the complication concerned with this project is successfully solved. There is a scope of future development in this project.

The new system promises to be accurate, adequate and produce timely information and when needed by the management. the system was found to be stable under all condition.

The system as flexibility for future improvement, modification and expansion. In future, if the management needs some more reports, then can be built and integrated with this system. This software is fully ser friendly one.

This project can be implemented in the future to meet the current technologies trends and needs

FUTURE ENCHANCEMENTS

CHAPTER-7

FUTURE ENCHANCEMENTS

There is always improvement in any software package, however good and efficient it may be. But the important requires that the system should be flexible enough for the further modification. Considering this important factor, the system is designed in such a way that further enhancement without affecting the system presently developed.

- Even though the screen printing form is computerized, still the unavoidable manual work is needed to verify the certificates and document.
- Now it was developed for screen printing forms only, in future it was enhanced into all other printing forms, digital printing etc.

APPENDIX-A

BIBLIOGRAPHY

We need to refer these following books while developing this project. In spite of version mismatch these books gave us lot of useful information related this project.

BOOK REFERENCE

- "Getting Started with Visual Studio 2022"
  Dirk Strauss
- "Microsoft visual basic .NET programming for the absolute beginner"
  Jonathan S. Harbour
- "Microsoft access quick reference"
  Chan Winter

APPENDIX-B

\REVIEW OF LITERATURE

This document contains general and specific requirements of the individual modules in the system. This will explain the performance, testing, and constrains of the system.

VISUAL BASIC

Visual Basic(VB.NET)is the third-generation event-driven programming language and integrated development environment (IDE) from Microsoft for its COM programming model. VB is also considered a relatively easy to learn and use programming language, because of its graphical development features and BASIC heritage.

Visual basic of derived from BASIC and enable the rapid application development (RAD) of graphical application, access database using object, remote or ActiveX data object, and creation of ActiveX control and objects, scripting such as VBA and VBS script are syntactically similar to visual basic, but perform

A programmer can put together an application using the components provide with visual basic salt Programs written in visual basic can also use the windows APL, but doing so requires external function declaration.

Language features

The visual basic NET was designed to be easily learned and used by beginner programmers. The language not only allows programmers to create simple GUI application. Programming in VB.NET is a combination of visually arranging components or control on a from specifying attributes and action of those components and writing additional lines of code for more functionality. Since default attributes and actions are defined for the components, a simple program can be created without the programmer having to write many lines of code. Performance problem were experienced by earlier versions but with faster computers and native code compilation this has become less of an issue.

Although programs can be compiled into native code executables from version 5 onward, they still require the presence of runtime libraries of approximately 1 MB in size. This runtime is included by default in windows 2000 and later but for earlier versions of windows like 95/98/NT it must be distributed together with the executable.

Forms are created using drag-and-drop techniques. A tool is used to place control (e.g., textboxes, buttons) On the form (window). Control have attributes and event handlers associated with them. Default values are provided when the control is created but may be changed by the programmer. Many attribute value can be modified during run time based on user actions or changes in the environment providing application. Form example, code can be inserted into the resize event handler to reposition a control so that it remains centred on the expand to fill from the from etc. By inserting code in the event handler for key press

in a text box, the program can automatically translate the case of the text being entered, or even prevent certain characters from being instead.

visual basic can created executables (EXE file), ActiveX control or DLL files, but is primarily used to develop windows application and to interface database system. Dialog boxes with less functionality can be used to provide pop-up capability. Control provide the basic functionality of the application while programmers can insert additional logic within the appropriate eventhandlers. For example, a drop-down combination box will automatically display its list and allow the user to select any element.

And event handler is called when and item is selected, which can them execute additional code created by the programmer to perform some action based on which element was selected such as populating a related list.

Alternatively a visual basic component can have no user interface and instead provide ActiveX object to other programs via component object model (COM). This allows for server-side processing or and add-in module.

The language is garbage collected using reference counting, has a large library of utility objects and as basic object oriented support. Since the more common components are included in the many other programming language, visual basic is generally not case sensitive, although it will transform keywords into a standard case configuration and force the case of variables names to conform to the case of the entry within the symbol table. String comparisons are case sensitive by default but can be made case insensitive if so desired.

The visual basic compiler is shared with other visual studio language (c, c++), but restriction in the IDE do not allow the creation of some targets (window model DLLs) and threading model.

.NET FRAMEWORK

The Microsoft .NET framework is a software framework that can be installed on computers running Microsoft window operating system. It includes a large library of coded solution to common programming problems and virtual machine that manage the execution of programs written specifically for the framework. The .NET framework is a Microsoft offering and is intended to be used by most new application created fro the windows platform. The framework's base class library provides a large range of features including user interface, data access, database connectivity, cryptography, application development, numeric algorithms, and network communication. The class library is used by programmers, who combine it with their own code to produce application.

Common Language Infrastructure

The purpose of the common language infrastructure, or CLI, is to provide a language-neutral platform for application development and execution, including function for exception handling, garbage collection, security, and interoperability. By implementing by the core aspects of the NET framework within

the scope of the CLR, this functionality will not be tied to a single language but will be available across the many language supported by the framework Microsoft's implementation of the CLI is called the common or CLR.

The CLI includes the Common Type System (CTS) and Common Language Specification (CLS). No matter which programming language they are written in, CLI applications are compiled into Intermediate Language (IL), which is further compiled into the target machine language by the Common Language Runtime (CLR) software.

Common Type System

In Microsoft's .NET Framework, the Common Type System (CTS) is a standard that specifies how type definitions and specific values of types are represented in computer memory. It is intended to allow programs written in different programming languages to easily share information. As used in programming languages, a type can be described as a definition of a set of values (for example, "all integers between 0 and 10"), and the allowable operations on those values (for example, addition and subtraction).

Common Language Specification

A Common Language Specification (CLS) is a document that says how computer programs can be turned into Common Intermediate Language (CIL) code. When several languages use the same bytecode, different parts of a program can be written in different languages. Microsoft uses a Common Language Specification for their .NET Framework. To fully interact with other objects regardless of the language they were used in, objects must expose to callers only those features that are common to all the languages they must exchange information with.

It was always a dream of Microsoft to unite all different languages under one umbrella and CLS is one step towards that. Microsoft has defined CLS which are nothing but guidelines for languages to follow so that it can communicate with other .NET languages in a seamless manner.

Common Language Runtime

The Common Language Runtime (CLR) is a component of the Microsoft .NET Framework that manages the execution of .NET applications. It is responsible for loading and executing the code written in various .NET programming languages, including C#, VB.NET, F#, and others.

The CLR provides many services to .NET applications, including memory management, type safety, security, and exception handling. It also provides Just-In-Time (JIT) compilation, which compiles the CIL code into machine code on the fly as the program runs, optimizing performance.

Additionally, the CLR provides a framework for developing and deploying .NET applications, including a set of libraries, called the .NET Framework Class Library, which provides access to a wide range of functionality, such as input/output operations, networking, database connectivity, and user interface design.

JIT(Just In Time Compiler):

In computing, just-in-time (JIT) compilation (also dynamic translation or run-time compilations) is a way of executing computer code that involves compilation during execution of a program (at run time) rather than before execution. This may consist of source code translation but is more commonly bytecode translation to machine code, which is then executed directly. A system implementing a JIT compiler typically continuously analyses the code being executed and identifies parts of the code where the speedup gained from compilation or recompilation would outweigh the overhead of compiling that code.

It is responsible for converting the CIL (Common Intermediate Language) into machine code or native code using the Common Language Runtime environment.

VB.NET

The VB.NET stands for Visual Basic. Network Enabled Technologies. It is a simple, high-level, object-oriented programming language developed by Microsoft in 2002. It is a successor of Visual Basic 6.0, that is implemented on the Microsoft .NET framework. Furthermore, it supports the OOPs concept, such as abstraction, encapsulation, inheritance, and polymorphism. Therefore, everything in the VB.NET language is an object, including all primitive data types (Integer, String, char, long, short, Boolean, etc.), user-defined data types, events, and all objects that inherit from its base class. It is not a case sensitive language, whereas, C++, Java, and C# are case sensitive language.

Applications built using the VB.NET language are very reliable and scalable, relying on the .NET Framework to access all libraries that help to execute a VB.NET program. With this language, you can develop a fully object-oriented application that is similar to an application created through another language such as C++, Java, or C#. In addition, applications or programs of VB.NET are not only running on the window operating system but can also run on Linux or Mac OS.

The VB.NET language is designed in such a way that any new beginner or novice and the advanced programmer can quickly develop a simple, secure, robust, high performance of web, windows, console, and mobile application running on .NET Framework.

Version of Visual Basic .NET

Microsoft launched VB.NET in 2002 as the successor to its original Visual Basic language, the last version of which was Visual Basic 6.0. Succeeding the classic Visual Basic version 6.0, the first version of Visual Basic .NET debuted in 2002. As of 2020, ten versions of Visual Basic .NET are released.

- **Visual Basic .NET 2002 (VB 7.0)**

  The first version, Visual Basic .NET, relies on .NET Framework 1.0. The most important feature is managed code, which contrasts with the classic Visual Basic.

- **Visual Basic .NET 2003 (VB 7.1)**

  Visual Basic .NET 2003 was released with .NET Framework 1.1. New features included support for the .NET Compact Framework and a better VB upgrade wizard. Improvements

were also made to the performance and reliability of .NET IDE (particularly the background compiler) and runtime.

- **Visual Basic .NET 2005 (VB 8.0)**

  After Visual Basic .NET 2003, Microsoft dropped ".NET" from the name of the product, calling the next version Visual Basic 2005.

  For this release, Microsoft added many features intended to reinforce Visual Basic .NET's focus as a rapid application development platform and further differentiate it from C#.

- **Visual Basic .NET 2008 (VB 9.0)**

  Visual Basic 9.0 was released along with .NET Framework 3.5 on November 19, 2007.

  For this release, Microsoft added many features, including:

  A true conditional operator, "If(condition as boolean, truepart, falsepart)", to replace the "IIf" function.
  Anonymous types
  Support for LINQ
  Lambda expressions
  XML Literals
  Type Inference
  Extension methods

- **Visual Basic .NET 2010 (VB 10.0)**

  In April 2010, Microsoft released Visual Basic 2010. Microsoft had planned to use Dynamic Language Runtime (DLR) for that release but shifted to a co-evolution strategy between Visual Basic and sister language C# to bring both languages into closer parity with one another.

- **Visual Basic 2012 (VB 11.0)**

  Visual Basic 2012 was released alongside .NET Framework 4.5. Major features introduced in this version include:
  Asynchronous programming with "async" and "await" statements
  Iterators
  Call hierarchy
  Caller information
  "Global" keyword in "namespace" statements

- **Visual Basic 2013 (VB 12.0)**

  Visual Basic 2013 was released alongside .NET Framework 4.5.1 with Visual Studio 2013. Can also build .NET Framework 4.5.2 applications by installing Developer Pack.

- **Visual Basic 2015 (VB 14.0)**

  Visual Basic 2015 (code named VB "14.0") was released with Visual Studio 2015. Language features include a new "?." operator to perform inline null checks, and a new string interpolation feature is included to format strings inline.

- **Visual Basic 2017 (VB 15.x)**

  Visual Basic 2017 (code named VB "15.0") was released with Visual Studio 2017. Extends support for new Visual Basic 15 language features with revision 2017, 15.3, 15.5, 15.8. Introduces new refactorings that allow organizing source code with one action.

- **Visual Basic 2019 (VB 16.0)**

  Visual Basic 2019 (code named VB "16.0") was released with Visual Studio 2019. It is the first version of Visual Basic focused on .NET Core.

TOOLS

Visual Studio

Microsoft visual studio includes native support for data programming with Microsoft SQL server. It can be used to write and debug code to be executed by SQL CLR. It also includes a data designer that can be used to graphically create view are edit database scheme. Queries can be created either visually or using code. SSMS 2008 onwards provides intelligence for SQL queries as well.

MS Access

Microsoft Access also know as Microsoft office access 2007, is a database management system from Microsoft that combines the relational Microsoft jet database engine with the graphical user interface and software-development tools. It is a member of the Microsoft office suite of application include in the professional and higher edition or sold separately.

Microsoft Access stores data in its own format based on the Access jet data base engine. It can also import or link directly to data stored in other application in databases.

Software developers and data architects can use Microsoft Access to develop application software and "power user" can use it to build software application. Like other office application access is

supported by visual basic for application an object-oriented programming language that can reference a variety of objects including DAO(data access object), ActiveX data object and many other ActiveX component. Visual object used in from and report expose their methods and properties in the VBA programming environment and VBA code modules may declare and call window operating-system function.

ADVANTAGE OF VB.NET OVER VB

Visual basic won't support large scale application but you can build them around visual basic. Because the need of enterprise or large scale application were at odds with visual basic mostly because of the lack of important object-oriented programming features that would make large-scale development project manageable. This often led to code bloat and old design that were difficult to maintain.

Visual basic has been known as rapid application development (RAD) platform. One fairly common approach to developing application was to implement them quickly in visual basic isolate the critical performance areas and replace the components. This type of development offered three distinct advantages.

- You could create a fully functional application relatively quickly.
- You had a reference implementation for the admittedly harder to build C++ components.
- Your test people could build tests based on the visual basic code right away providing a great functional check against any replacement components.

The limitation of COM

COM has to be the most successful component architecture in the entire history of computing. Visual basic is by far the best platform for creating COM components quickly and easily. But COM is not without its own problem:

- You cannot inherit from Com components.COM offers no inherent ability to extend COM component. This is a significant architectural limitation although it obviously has not presented too great a limitation.
- COM dependent on a registry. This causes interesting deployment issues. Most COM cannot deploy without some kind of installation package.
- Versioning under COM is it own sort of hell. Over time a new term was coined to describe this situation: DLL hell. Lack of version checking new DLL over write DLL and DLLs with some name being installed in a common directory have all contributed to this problem.

Visual basic: the next generation

Visual basic .NET is a major advancement as a language and development platform. The combination of visual basic .NET and the .NET framework provides a wealth of features unmatched by classic visual basic.

Moving Beyond COM:

Visual basic .NET and the .Net common language runtime (CLR) address the limitations of COM in four key areas:

Implementation inheritance:

Visual basic .NET not only provides implementation inheritance, but it goes a step further. This architecture of the .NET framework allows components to be extended though inheritance mechanism.

Reduce registry dependencies:

.NET components don't require the use of the system registry. The recommended way to manage your visual basic .NET application's settings is though an application configuration file, not the registry.

Side-by-side development supporting to version of the same application on the same machine used to require a lot of work. You had to ensure that newer version of the same COM components wouldn’t overwrite each other.

APPENDIX-C

SAMPLE CODING

LOADING PAGE CODE:

```
Public Class loding
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        MyProgress.Increment(1)
        Dim percentage As String
        percentage = Convert.ToString(MyProgress.Value)
        PerLb.Text = percentage + "%"
        If MyProgress.Value = 100 Then
            Me.Hide()
            Dim obj = New home
            obj.Show()
            Timer1.Enabled = False
        End If
    End Sub

    Private Sub loding_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Timer1.Start()
    End Sub
End Class
```

HOMEPAGE CODE:

```
Public Class home
    Private Sub Guna2GradientButton1_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton1.Click
        Dim obj = New pd
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub Guna2GradientButton2_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton2.Click
        Dim obj = New bill_page
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub Guna2GradientButton3_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton3.Click
        Dim obj = New supplierdetails
        obj.Show()
        Me.Hide()
    End Sub
```

```
    Private Sub Guna2GradientButton4_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton4.Click
        Dim obj = New HomeDelivery
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub Guna2GradientButton5_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton5.Click


            Dim obj = New searchdeatils
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub Guna2GradientButton6_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton6.Click
        Dim obj = New reports
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub home_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub
End Class
```

PRODUCT INVENTORY PAGE CODE:

```
Imports System.Data.SqlClient
Public Class pd
        Dim Con = New
        SqlConnection("DataSource=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Lenovo\On
        eDrive\Documents\departmentalstore.mdf;Integrated Security=True;Connect Timeout=30")

    Private Sub clear()
        ItNameTb.Text = ""
        QtyTb.Text = ""
        MesuCb.SelectedIndex = 0
        PriceTb.Text = ""
        CatCb.SelectedIndex = 0
    End Sub

    Private Sub DisplayItem()
        Con.Open()
        Dim query = "select * from ItemTb1"
```

```
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        adapter.Fill(ds)
        ItemDGV.DataSource = ds.Tables(0)
        Con.Close()
    End Sub
    Dim key = 0



    Private Sub Button1_Click_1(sender As Object, e As EventArgs) Handles Button1.Click
        If ItNameTb.Text = "" Or QtyTb.Text = "" Or MesuCb.SelectedIndex = -1 Or PriceTb.Text = "" Or
CatCb.SelectedIndex = -1 Then
            MsgBox("Missing Information")
        Else
            Try
                Con.Open()
                Dim query = "insert into ItemTb1 values('" & ItNameTb.Text & "'," & QtyTb.Text         & ",'" &
MesuCb.SelectedItem.ToString() & "'," & PriceTb.Text & ",'" & CatCb.SelectedItem.ToString() & "') "
                Dim cmd As SqlCommand
                cmd = New SqlCommand(query, Con)
                cmd.ExecuteNonQuery()
                MsgBox("Item Saved Successfully")
                Con.Close()
                DisplayItem()
                clear()
            Catch ex As Exception

            End Try
        End If
    End Sub

    Private Sub Button2_Click_1(sender As Object, e As EventArgs) Handles Button2.Click
        If ItNameTb.Text = "" Or QtyTb.Text = "" Or MesuCb.SelectedIndex = -1 Or CatCb.SelectedIndex = -
1 Or PriceTb.Text = "" Then
            MsgBox("Missing Information")
        Else
            Try
                Con.Open()
  Dim query = "Update ItemTB1 set ItName='" & ItNameTb.Text & "',ItQty=" & QtyTb.Text &     ",ItPrice=
" & PriceTb.Text & ",ItCat= '" & CatCb.SelectedItem.ToString() & "' where Itid=" & key & ""
                Dim cmd As SqlCommand
                cmd = New SqlCommand(query, Con)
                cmd.ExecuteNonQuery()
                MsgBox("Item Updated Successfully")
                Con.Close()
                DisplayItem()
```

```
            clear()
        Catch ex As Exception

        End Try
    End If
End Sub

Private Sub Button3_Click_1(sender As Object, e As EventArgs) Handles Button3.Click
    If key = 0 Then
        MsgBox("Select Item to Delete")
    Else
        Try
            Con.Open()
            Dim query = "delete from ItemTb1 where ItId=" & key & ""
            Dim cmd As SqlCommand
            cmd = New SqlCommand(query, Con)
            cmd.ExecuteNonQuery()
            MsgBox("Item Deleted Successfully")
            Con.Close()
            DisplayItem()
            clear()
        Catch ex As Exception

        End Try
    End If
End Sub

Private Sub pd_Load_1(sender As Object, e As EventArgs) Handles MyBase.Load
    DisplayItem()
End Sub

Private Sub ItemDGV_CellMouseClick_1(sender As Object, e As DataGridViewCellMouseEventArgs)
Handles ItemDGV.CellMouseClick
    Dim row As DataGridViewRow = ItemDGV.Rows(e.RowIndex)
    ItNameTb.Text = row.Cells(1).Value.ToString
    QtyTb.Text = row.Cells(2).Value.ToString
    MesuCb.SelectedItem = row.Cells(3).Value.ToString
    PriceTb.Text = row.Cells(4).Value.ToString
    CatCb.SelectedItem = row.Cells(5).Value.ToString
    If ItNameTb.Text = "" Then
        key = 0
    Else
        key = Convert.ToInt32(row.Cells(0).Value.ToString)
    End If
End Sub

Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
    clear()
End Sub

Private Sub Label9_Click(sender As Object, e As EventArgs) Handles Label9.Click
    Dim Obj = New home
```

```
        Obj.Show()
        Me.Hide()
    End Sub

    Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
        Dim Obj = New home
        Obj.Show()
        Me.Hide()
    End Sub


    Private Sub search()
        Con.Open()
        Dim query = "Select * from ItemTb1 where ItId like'%" & SearchTb.Text & "%' or ItName like '%" &
SearchTb.Text & "%' or ItCat like'%" & SearchTb.Text & "%'"
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        Dim dt As DataTable
        dt = New DataTable
        adapter.Fill(dt)
        If dt.Rows.Count > 0 Then
            ItemDGV.DataSource = dt
        Else
            MsgBox("No Record Found!")
        End If
        Con.Close()
    End Sub

    Private Sub SearchTb_KeyUp(sender As Object, e As KeyEventArgs) Handles SearchTb.KeyUp
        search()
    End Sub
End Class
```

SALES CODE:

```
Imports System.Data.SqlClient

Public Class bill_page
    Dim Con = New SqlConnection("Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Lenovo\OneDrive\Documents\departme
ntalstore.mdf;Integrated Security=True;Connect Timeout=30")
    Private Sub AddBill()
        Try
            Con.Open()
            Dim query = "Insert into BillTb1 values('" & ClNameTb.Text & "','" & GrdTotal & "','" &
DateTime.Now.ToLongDateString & "')"
            Dim cmd As SqlCommand
            cmd = New SqlCommand(query, Con)
```

```
            cmd.ExecuteNonQuery()
            MsgBox("Bill Saved Successfully")
            Con.Close()
            TotalLbl.Text = "Total"
            BillDGV.Rows.Clear()
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub
    Private Sub DisplayItem()
        Con.Open()
        Dim query = "select * from ItemTB1"
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        adapter.Fill(ds)
        ItemDGV.DataSource = ds.Tables(0)
        Con.Close()
    End Sub
    Dim key = 0, Stock = 0
    Private Sub UpdateItem()
        Dim newQty = Stock - Convert.ToInt32(QtyTb.Text)
        Try
            Con.Open()
            Dim query = "Update ItemTB1 set ItQty=" & newQty & " where Itid=" & key & ""
            Dim cmd As SqlCommand
            cmd = New SqlCommand(query, Con)
            cmd.ExecuteNonQuery()
            Con.Close()
            DisplayItem()
        Catch ex As Exception

        End Try
    End Sub
    Private Sub Reset()
        ItNameTb.Text = ""
        PriceTb.Text = ""
        QtyTb.Text = ""
        'TotalLbl.Text = "Total"
        key = 0
        Stock = 0
    End Sub
    Dim i = 0, GrdTotal = 0
    Private Sub bill_page_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        DisplayItem()
    End Sub

    Private Sub ItemDGV_CellMouseClick(sender As Object, e As DataGridViewCellMouseEventArgs)
Handles ItemDGV.CellMouseClick
```

```
        Dim row As DataGridViewRow = ItemDGV.Rows(e.RowIndex)
        ItNameTb.Text = row.Cells(1).Value.ToString
        PriceTb.Text = row.Cells(4).Value.ToString
        If ItNameTb.Text = "" Then
            key = 0
        Else
            key = Convert.ToInt32(row.Cells(0).Value.ToString)
            Stock = Convert.ToInt32(row.Cells(2).Value.ToString)
        End If
    End Sub

    Private Sub Label9_Click(sender As Object, e As EventArgs) Handles Label9.Click
        Dim obj = New home
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
        Dim obj = New home
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub Guna2GradientButton1_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton1.Click
        If ClNameTb.Text = "" Then
            MsgBox("Enter Client Name")
        Else
            AddBill()
        End If
    End Sub

    Private Sub Guna2GradientButton3_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton3.Click
        Reset()
    End Sub

    Private Sub Guna2GradientButton2_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton2.Click
        If ItNameTb.Text = "" Or PriceTb.Text = "" Then
            MsgBox("Select the Item")
        ElseIf QtyTb.Text = "" Then
            MsgBox("Enter the Qunatity")
        ElseIf QtyTb.Text > Stock Then
            MsgBox("No Enough Stock")
        Else
            Dim rnum As Integer = BillDGV.Rows.Add()
            i = i + 1
            Dim total = Convert.ToInt32(QtyTb.Text) * Convert.ToInt32(PriceTb.Text)
            BillDGV.Rows.Item(rnum).Cells("Column1").Value = i
            BillDGV.Rows.Item(rnum).Cells("Column2").Value = ItNameTb.Text
            BillDGV.Rows.Item(rnum).Cells("Column3").Value = PriceTb.Text
```

```
            BillDGV.Rows.Item(rnum).Cells("Column4").Value = QtyTb.Text
            BillDGV.Rows.Item(rnum).Cells("Column5").Value = total
            GrdTotal = GrdTotal + total
            Dim tot As String
            tot = "Rs" + Convert.ToString(GrdTotal)
            TotalLbl.Text = tot
            UpdateItem()
            DisplayItem()
            'Reset()
        End If
    End Sub

    Private Sub GroupBox1_Enter(sender As Object, e As EventArgs) Handles GroupBox1.Enter

    End Sub
    Private Sub search()
        Con.Open()
        Dim query = "Select * from ItemTb1 where ItId like'%" & SearchTb.Text & "%' or ItName like '%" &
SearchTb.Text & "%' or ItCat like '%" & SearchTb.Text & "%' "
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        Dim dt As DataTable
        dt = New DataTable
        adapter.Fill(dt)
        If dt.Rows.Count > 0 Then
            ItemDGV.DataSource = dt
        Else
            MsgBox("No Record Found!")
        End If
        Con.Close()
    End Sub

    Private Sub SearchTb_KeyUp(sender As Object, e As KeyEventArgs) Handles SearchTb.KeyUp
        search()
    End Sub
End Class
```

SUPPLIER PAGE CODE:

```
Imports System.Data.SqlClient
Imports System.Text.RegularExpressions

Public Class supplierdetails
    Dim Con = New SqlConnection("Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Lenovo\OneDrive\Documents\departme
ntalstore.mdf;Integrated Security=True;Connect Timeout=30")
    Private Sub clear()
        CoNameTb.Text = ""
```

```
        MobTb.Text = ""
        AddTb.Text = ""
        ProNameTb.Text = ""
        SearchTb.Text = ""
    End Sub
    Private Sub DisplayItem()
        Con.Open()
        Dim query = "Select * from StockTb1"
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        adapter.Fill(ds)
        StockDGV.DataSource = ds.Tables(0)
        Con.Close()
    End Sub
    Dim key = 0

    Private Sub StockDGV_CellMouseClick(sender As Object, e As DataGridViewCellMouseEventArgs)
Handles StockDGV.CellMouseClick
        Dim row As DataGridViewRow = StockDGV.Rows(e.RowIndex)
        CoNameTb.Text = row.Cells(1).Value.ToString
        MobTb.Text = row.Cells(2).Value.ToString
        AddTb.Text = row.Cells(3).Value.ToString
        ProNameTb.Text = row.Cells(4).Value.ToString
        If CoNameTb.Text = "" Then
            key = 0
        Else
            key = Convert.ToInt32(row.Cells(0).Value.ToString)
        End If
    End Sub

    Private Sub login_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        DisplayItem()
    End Sub

    Private Sub Button2_Click_1(sender As Object, e As EventArgs) Handles Button2.Click
        If CoNameTb.Text = "" Or MobTb.Text = "" Or AddTb.Text = "" Or ProNameTb.Text = "" Then
            MsgBox("Missing Information")
        Else
            Try
                Con.Open()
                Dim query = "Update StockTb1 set ComName='" & CoNameTb.Text & "',Mob=" & MobTb.Text
& ",Address='" & AddTb.Text & "',ProName='" & ProNameTb.Text & "' where StId=" & key & ""
                Dim cmd As SqlCommand
                cmd = New SqlCommand(query, Con)
                cmd.ExecuteNonQuery()
                MsgBox("Item Updated Successfully")
                Con.Close()
                DisplayItem()
```

```
            clear()
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End If
End Sub

Private Sub Button4_Click_1(sender As Object, e As EventArgs) Handles Button4.Click
    clear()
End Sub

Private Sub Button3_Click_1(sender As Object, e As EventArgs) Handles Button3.Click
    If key = 0 Then
        MsgBox("Select Details to Delete")
    Else
        Try
            Con.Open()
            Dim query = "delete from StockTb1 where StId=" & key & ""
            Dim cmd As SqlCommand
            cmd = New SqlCommand(query, Con)
            cmd.ExecuteNonQuery()
            MsgBox("Details Deleted Successfully")
            Con.Close()
            DisplayItem()
            clear()
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End If
End Sub

Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    If CoNameTb.Text = "" Or MobTb.Text = "" Or AddTb.Text = "" Or ProNameTb.Text = "" Then
        MsgBox("Missing Information")
    Else
        Try
            Con.Open()
            Dim query = "insert into StockTb1 values('" & CoNameTb.Text & "'," & MobTb.Text & ",'" &
AddTb.Text & "','" & ProNameTb.Text & "')"
            Dim cmd As SqlCommand
            cmd = New SqlCommand(query, Con)
            cmd.ExecuteNonQuery()
            MsgBox("Details Saved Successfully")
            Con.Close()
            DisplayItem()
            clear()
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End If
End Sub
Private Sub search()
```

```
        Con.Open()
        Dim query = "Select * from StockTb1 where ComName like '%" & SearchTb.Text & "%'"
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        Dim dt As DataTable
        dt = New DataTable
        adapter.Fill(dt)
        If dt.Rows.Count > 0 Then
            StockDGV.DataSource = dt
        Else
            MsgBox("No Record Found!")
        End If
        Con.Close()
    End Sub


    Private Sub SearchTb_KeyUp(sender As Object, e As KeyEventArgs) Handles SearchTb.KeyUp
        search()
    End Sub

    Private Sub Label4_Click(sender As Object, e As EventArgs) Handles Label4.Click
        Dim Obj = New home
        Obj.Show()
        Me.Hide()
    End Sub

    Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
        Dim Obj = New home
        Obj.Show()
        Me.Hide()
    End Sub
End Class
```

HOME DELIVERY PAGE CODE:

```
Imports System.Data.SqlClient

Public Class HomeDelivery
    Dim Con = New SqlConnection("Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Lenovo\OneDrive\Documents\departme
ntalstore.mdf;Integrated Security=True;Connect Timeout=30")
    Private Sub AddBill()
        Try
            Con.Open()
            Dim query = "Insert into HomeTb1 values('" & ClNameTb.Text & "','" & AddressTb1.Text & "','" &
MobileTb.Text & "','" & GrdTotal & "','" & DateTime.Now.ToLongDateString & "')"
            Dim cmd As SqlCommand
            cmd = New SqlCommand(query, Con)
```

```
        cmd.ExecuteNonQuery()
        MsgBox("Bill Saved Successfully")
        Con.Close()
        TotalLbl.Text = "Total"
        BillDGV.Rows.Clear()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
Private Sub DisplayItem()
    Con.Open()
    Dim query = "select * from ItemTB1"
    Dim cmd = New SqlCommand(query, Con)
    Dim adapter As SqlDataAdapter
    adapter = New SqlDataAdapter(cmd)
    Dim builder As New SqlCommandBuilder(adapter)
    Dim ds As DataSet
    ds = New DataSet
    adapter.Fill(ds)
    ItemDGV.DataSource = ds.Tables(0)
    Con.Close()
End Sub
Dim key = 0, Stock = 0
Private Sub UpdateItem()
    Dim newQty = Stock - Convert.ToInt32(QtyTb.Text)
    Try
        Con.Open()
        Dim query = "Update ItemTB1 set ItQty=" & newQty & " where Itid=" & key & ""
        Dim cmd As SqlCommand
        cmd = New SqlCommand(query, Con)
        cmd.ExecuteNonQuery()
        Con.Close()
        DisplayItem()
    Catch ex As Exception

    End Try
End Sub
Private Sub Reset()
    ItNameTb.Text = ""
    PriceTb.Text = ""
    QtyTb.Text = ""
    'TotalLbl.Text = "Total"
    key = 0
    Stock = 0
End Sub
Dim i = 0, GrdTotal = 0

Private Sub HomeDelivery_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    DisplayItem()
End Sub
```

```
    Private Sub ItemDGV_CellMouseClick(sender As Object, e As DataGridViewCellMouseEventArgs)
Handles ItemDGV.CellMouseClick
        Dim row As DataGridViewRow = ItemDGV.Rows(e.RowIndex)
        ItNameTb.Text = row.Cells(1).Value.ToString
        PriceTb.Text = row.Cells(4).Value.ToString
        If ItNameTb.Text = "" Then
            key = 0
        Else
            key = Convert.ToInt32(row.Cells(0).Value.ToString)
            Stock = Convert.ToInt32(row.Cells(2).Value.ToString)
        End If
    End Sub

    Private Sub Label9_Click(sender As Object, e As EventArgs) Handles Label9.Click
        Dim obj = New home
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
        Dim obj = New home
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub Guna2GradientButton1_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton1.Click
        If ClNameTb.Text = "" Then
            MsgBox("Enter Client Name")
        Else
            AddBill()
        End If
    End Sub

    Private Sub Guna2GradientButton3_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton3.Click
        Reset()
    End Sub

    Private Sub Guna2GradientButton2_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton2.Click
        If ItNameTb.Text = "" Or PriceTb.Text = "" Then
            MsgBox("Select the Item")
        ElseIf QtyTb.Text = "" Then
            MsgBox("Enter the Qunatity")
        ElseIf QtyTb.Text > Stock Then
            MsgBox("No Enough Stock")
        Else
            Dim rnum As Integer = BillDGV.Rows.Add()
            i = i + 1
            Dim total = Convert.ToInt32(QtyTb.Text) * Convert.ToInt32(PriceTb.Text)
            BillDGV.Rows.Item(rnum).Cells("Column1").Value = i
```

```
            BillDGV.Rows.Item(rnum).Cells("Column2").Value = ItNameTb.Text
            BillDGV.Rows.Item(rnum).Cells("Column3").Value = PriceTb.Text
            BillDGV.Rows.Item(rnum).Cells("Column4").Value = QtyTb.Text
            BillDGV.Rows.Item(rnum).Cells("Column5").Value = total
            GrdTotal = GrdTotal + total
            Dim tot As String
            tot = "Rs" + Convert.ToString(GrdTotal)
            TotalLbl.Text = tot
            UpdateItem()
            DisplayItem()
            'Reset()
        End If
    End Sub
    Private Sub search()
        Con.Open()
        Dim query = "Select * from ItemTb1 where ItId like'%" & SearchTb.Text & "%' or ItName like '%" &
SearchTb.Text & "%' or ItCat like '%" & SearchTb.Text & "%' "
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        Dim dt As DataTable
        dt = New DataTable
        adapter.Fill(dt)
        If dt.Rows.Count > 0 Then
            ItemDGV.DataSource = dt
        Else
            MsgBox("No Record Found!")
        End If
        Con.Close()
    End Sub

End Class
```

RETURN PAGE CODE:

```
Imports System.Data.SqlClient

Public Class searchdeatils
    Dim Con = New SqlConnection("Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Lenovo\OneDrive\Documents\departme
ntalstore.mdf;Integrated Security=True;Connect Timeout=30")
    Private Sub DisplayItem()
        Con.Open()
        Dim query = "select * from BillTb1"
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
```

```
        ds = New DataSet
        adapter.Fill(ds)
        CusDGV.DataSource = ds.Tables(0)
        Con.Close()
    End Sub
    Private Sub DisplayItem1()
        Con.Open()
        Dim query = "select * from ItemTb1"
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        adapter.Fill(ds)
        ItemDGV.DataSource = ds.Tables(0)
        Con.Close()
    End Sub
    Private Sub UpdateItem()
        Dim newQty = Stock + Convert.ToInt32(QtyTb.Text)
        Try
            Con.Open()
            Dim query = "Update ItemTB1 set ItQty=" & newQty & " where Itid=" & key & ""
            Dim cmd As SqlCommand
            cmd = New SqlCommand(query, Con)
            cmd.ExecuteNonQuery()
            MsgBox("Item Updated Successfully")
            Con.Close()
            DisplayItem1()
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub
    Private Sub searchdeatils_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        DisplayItem()
    End Sub
    Dim key = 0, Stock = 0
    Private Sub CusDGV_CellMouseClick(sender As Object, e As DataGridViewCellMouseEventArgs)
Handles CusDGV.CellMouseClick
        Dim row As DataGridViewRow = CusDGV.Rows(e.RowIndex)
        ClNameTb1.Text = row.Cells(1).Value.ToString
        AmountTb.Text = row.Cells(2).Value.ToString
        If ClNameTb1.Text = "" Then
            key = 0
        Else
            key = Convert.ToInt32(row.Cells(0).Value.ToString)
            Stock = Convert.ToInt32(row.Cells(2).Value.ToString)
        End If
    End Sub
    Dim i = 0, gdtotal = 0
    Private Sub Guna2GradientButton2_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton2.Click
```

```
        If ClNameTb1.Text = "" Or AmountTb.Text = "" Then
            MsgBox("Select the Item")
        ElseIf QtyTb.Text = "" Then
            MsgBox("Enter the Qunatity")
        Else
            Dim rnum As Integer = NewDGV.Rows.Add()
            i = i + 1
            Dim tot1 = Convert.ToInt32(QtyTb.Text) * Convert.ToInt32(PriceTb.Text)
            NewDGV.Rows.Item(rnum).Cells("C1").Value = i
            NewDGV.Rows.Item(rnum).Cells("C2").Value = ItNameTb.Text
            NewDGV.Rows.Item(rnum).Cells("C3").Value = PriceTb.Text
            NewDGV.Rows.Item(rnum).Cells("C4").Value = QtyTb.Text
            NewDGV.Rows.Item(rnum).Cells("C5").Value = tot1
            gdtotal = gdtotal + tot1
            Dim tot2 As String
            tot2 = "Rs" + Convert.ToString(gdtotal)
            TotalLbl.Text = tot2
            UpdateItem()
            DisplayItem1()
            'Reset()
        End If
    End Sub
    Private Sub search()
        Con.Open()
        Dim query = "Select * from ItemTb1 where ItId like'%" & ItNameTb.Text & "%' or ItName like '%" &
ItNameTb.Text & "%' or ItCat like '%" & ItNameTb.Text & "%' "
        Dim cmd = New SqlCommand(query, Con)
        Dim adapter As SqlDataAdapter
        adapter = New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(adapter)
        Dim ds As DataSet
        ds = New DataSet
        Dim dt As DataTable
        dt = New DataTable
        adapter.Fill(dt)
        If dt.Rows.Count > 0 Then
            ItemDGV.DataSource = dt
        Else
            MsgBox("No Record Found!")
        End If
        Con.Close()
    End Sub
    Private Sub Guna2GradientButton1_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton1.Click
        Try
            Con.Open()
            Dim query = "insert into ReturnTb1 values('" & ClNameTb1.Text & "'," & gdtotal & ",'" &
DateTime.Today.ToLongDateString & "')"
            Dim cmd As SqlCommand
            cmd = New SqlCommand(query, Con)
            cmd.ExecuteNonQuery()
            MsgBox("Bill Saved Successfully")
```

```
            Con.Close()
            TotalLbl.Text = "Total"
            NewDGV.Rows.Clear()
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub ItNameTb_KeyUp(sender As Object, e As KeyEventArgs) Handles ItNameTb.KeyUp
        search()
    End Sub

    Private Sub ItemDGV_CellMouseClick(sender As Object, e As DataGridViewCellMouseEventArgs)
Handles ItemDGV.CellMouseClick
        Dim row As DataGridViewRow = ItemDGV.Rows(e.RowIndex)
        ItNameTb.Text = row.Cells(1).Value.ToString
        PriceTb.Text = row.Cells(4).Value.ToString
        If ItNameTb.Text = "" Then
            key = 0
        Else
            key = Convert.ToInt32(row.Cells(0).Value.ToString)
            Stock = Convert.ToInt32(row.Cells(2).Value.ToString)
        End If
    End Sub

    Private Sub Label9_Click(sender As Object, e As EventArgs) Handles Label9.Click
        Dim obj = New home
        obj.Show()
        Me.Hide()
    End Sub

    Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
        Dim obj = New home
        obj.Show()
        Me.Hide()
    End Sub
End Class
```

REPORTS PAGE CODE:

```
Imports System.Data.SqlClient

Public Class reports
    Dim Con = New SqlConnection("Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Lenovo\OneDrive\Documents\departme
ntalstore.mdf;Integrated Security=True;Connect Timeout=30")
    Private Sub Label9_Click(sender As Object, e As EventArgs) Handles Label9.Click
        Dim obj = New home
        obj.Show()
        Me.Hide()
    End Sub
```

```
Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
    Dim obj = New home
    obj.Show()
    Me.Hide()
End Sub
Private Sub Item()
    Con.Open()
    Dim query = "select * from ItemTB1"
    Dim cmd = New SqlCommand(query, Con)
    Dim adapter As SqlDataAdapter
    adapter = New SqlDataAdapter(cmd)
    Dim builder As New SqlCommandBuilder(adapter)
    Dim ds As DataSet
    ds = New DataSet
    adapter.Fill(ds)
    RItemDGV.DataSource = ds.Tables(0)
    Con.Close()
End Sub
Private Sub sell()
    Con.Open()
    Dim query = "select * from BillTb1 "
    Dim cmd = New SqlCommand(query, Con)
    Dim adapter As SqlDataAdapter
    adapter = New SqlDataAdapter(cmd)
    Dim builder As New SqlCommandBuilder(adapter)
    Dim ds As DataSet
    ds = New DataSet
    adapter.Fill(ds)
    RSellDGV.DataSource = ds.Tables(0)
    Con.Close()
End Sub
Private Sub Home()
    Con.Open()
    Dim query = "select * from HomeTb1"
    Dim cmd = New SqlCommand(query, Con)
    Dim adapter As SqlDataAdapter
    adapter = New SqlDataAdapter(cmd)
    Dim builder As New SqlCommandBuilder(adapter)
    Dim ds As DataSet
    ds = New DataSet
    adapter.Fill(ds)
    RHomeDGV.DataSource = ds.Tables(0)
    Con.Close()
End Sub
Private Sub return1()
    Con.Open()
    Dim query = "select * from ReturnTb1"
    Dim cmd = New SqlCommand(query, Con)
    Dim adapter As SqlDataAdapter
    adapter = New SqlDataAdapter(cmd)
    Dim builder As New SqlCommandBuilder(adapter)
    Dim ds As DataSet
```

```
        ds = New DataSet
        adapter.Fill(ds)
        RReturnDGV.DataSource = ds.Tables(0)
        Con.Close()
    End Sub

    Private Sub RItemDGV_CellContentClick(sender As Object, e As DataGridViewCellEventArgs) Handles
RItemDGV.CellContentClick
        Item()
    End Sub

    Private Sub RSellDGV_CellContentClick(sender As Object, e As DataGridViewCellEventArgs) Handles
RSellDGV.CellContentClick
        sell()
    End Sub

    Private Sub RReturnDGV_CellContentClick(sender As Object, e As DataGridViewCellEventArgs)
Handles RReturnDGV.CellContentClick
        return1()
    End Sub

    Private Sub RHomeDGV_CellContentClick(sender As Object, e As DataGridViewCellEventArgs)
Handles RHomeDGV.CellContentClick
        Home()
    End Sub

    Private Sub Guna2GradientButton1_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton1.Click
        PrintPreviewDialog1.ShowDialog()
        PrintDocument1.Print()
    End Sub

    Private Sub Guna2GradientButton2_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton2.Click
        PrintPreviewDialog2.ShowDialog()
        PrintDocument2.Print()
    End Sub

    Private Sub Guna2GradientButton3_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton3.Click
        PrintPreviewDialog3.ShowDialog()
        PrintDocument3.Print()
    End Sub

    Private Sub Guna2GradientButton4_Click(sender As Object, e As EventArgs) Handles
Guna2GradientButton4.Click
        PrintPreviewDialog4.ShowDialog()
        PrintDocument4.Print()
    End Sub
End Class
```