**CHAPTER-1**

**INTRODUCTION**

Globalization, large corporations and other factors in the business environment have contributed to a large shift in this early paradigm and have forced the business owners of today to implement a more overarching. Strategy go stay alive one of these changes that business owners have come to realize is the importance of maintaining a strong database presence.

In the contemporary world, efficient record keeping cannot be overemphasize. Imagine the scenario when the manual processes and manual modes of instruction get replaced with electronic systems. One of such replacement can be done in the activities of organization by using an advanced mechanism like putting all the data regarding the factory in one place where all people can see it at anytime. So, the client who is the owner of Ramamurthy cement spun pipes and bricks factory Mr.Srinivas Gunti approached us to make a desktop application for his factory.

**1.1 ABOUT PROJECT:**

The modules in this project are interdependent i.e., the admin will update all the necessary information like adding employee, attendance, product details, daily productions, maintaining customer details etc. In stock module stock name, in-stock, stock usage is updated. Details about transportation are maintained in transportation module whereas the accountant can view all the details that are updated from the modules owner, stock and transportation. This project provides more ease for managing the data than manually maintaining in the documents.

**1.2 EXSITING SYSTEM WITH DRAWBACKS:**

At present, client is maintaining all the details manually. Ramamurthy factory have details which include sales, expenditure, employee details and their product details etc that need to be manually checked by the owner at the end of the day.

**Drawbacks (Problems):**

In case of existing system, the management has to face a lot of problems. Few of them as follows:

* Storing information in registers leads to redundancy of data.
* Requires more man power.
* Time taken for canvassing is more.
* All the activities conducted by the organization may not be focused.
* Reports are generated manually.
* Security is not present.

**1.3 PROPOSED SYSTEM WITH FEATURES:**

Ramamurthy cement spun pipes and bricks factory has been automated in which details about sales, expenditures, employee details, product details and transportation details have to be entered and can be viewed easily. The problem of manually writing a receipt has been overcome by generating an automated receipt.

**Features:**

The database has following features:

* The administrator can easily retrieve and update information whenever required.
* Data integrity and confidentiality are provided.
* Details of employees, products, sales, expenditures and transportation are maintained in database.
* Profit and loss can be viewed in accounts module.
* Accurate information: The accurate information can be retrieved from the database.
* Data Security: Security is provided for the system. Authorized user can only access the system

**CHAPTER-2**

**ANALYSIS**

The goal of system analysis is to determine where the problem is in an attempt to fix the system .This step involves breaking down the system in different pieces to analyze the situation, analyzing project goal, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

**2.1 HARDWARE AND SOFTWARE REQUIREMENTS**

**2.1.1 Hardware Requirements (Preferable):**

|  |  |
| --- | --- |
| PROCESSOR | INTEL CORE I5 |
| RAM CAPACITY | 2GB |
| HARD DISK | 80GB |
| PROCESSOR SPEED | 2.40 GHz |

The selection of hardware is very important in the existence and proper working of any software. In the selection of hardware, size and the capacity requirements are also important.

Table: 2.1.1 Hardware Requirements

**Software Requirements (Preferable):**

One of the most difficult tasks is that, the selection of the software, once system requirement is known that is determining whether a particular software package fits the requirements.

|  |  |
| --- | --- |
| FRONT END | JAVA |
| IDE’s | NET BEANS 7.1.2 |
| OPERATING SYSTEM | WINDOWS 7 |
| BACK END | ORACLE |
| UML PLATFORM | STAR UML |

Table: 2.1.2 Software Requirements

**2.2 FUNCTIONAL REQUIREMENTS AND NON-FUNCTIONAL REQUIREMENTS**

**Functional Requirements:**

The following are the functional requirements of our project. They are:

* A database has to be created to maintain the details of sales, expenditure, employee, products and transportation.
* The information should be updated whenever required.
* It should be very efficient in retrieval and updating of information.
* It shouldn't accept null values whenever validated.

**Non Functional Requirements:**

* **Maintainability:** Maintainability is used to make future maintenance easier, meet new requirements.
* **Robustness:** Robustness is the quality of being able to withstand stress, pressures or changes in procedure or circumstance.
* **Reliability:** Reliability is an ability of a person or system to perform and maintain its functions in circumstances.
* **Size:** The size of a particular application play a major role, if the size is less then efficiency will be high.
* **Speed:** If the speed is high then it is good. Since the no of lines in our code is less, hence the speed is high.

**2.3 MODULE DESCRIPTION:**

The project has been divided into these modules:

* Owner
* Accounts
* Stock
* Transportation

**OWNER:**

The Owner gets logged in from the login page. Here owner can add, view and update the details of employee, daily attendance, expenditure names, employee expenditures, other expenditures, product details, daily production and customer details. The owner has the right to access or edit the details of any employee or the product whenever he required. Username and password is provided for the owner to improve the security. To update or cross check or to view any of the required details the owner should login with provided username and password.

**STOCK:**

The stock module consists of details such as stock name, in-stock and stock usage. These details are updated by stock supervisor. In stock module, details like how much of stock is present in the factory and how much of stock is used can be known. With this the owner can easily guess that which type of material should be purchased. Authentication is provided for the stock module in order to provide secrecy among other users.

**TRANSPORTATION:**

The transportation module consists of details such as transportation delivery details and transportation expenditure. The delivery details consist of type of product, how many items are to be transported to the customer and the destination details are maintained. Authentication is provided for the transportation module in order to provide secrecy among other users.

**ACCOUNTS:**

The Accountant can view profit and loss of the factory and the other aspects like sales and expenditures of employee, stock, transportation and other details. The accountant can also view the required data by providing from and to date accordingly to get the specified details in an easy manner with this the time gets reduced. To perform all these operations the accountant should login into the application by providing their specified username and password, the authentication is provided in order to get rid of secrecy, data loss and other security problems.

CHAPTER- 3

**DESIGN**

**3.1 Block Diagram**

The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation. The block diagram is shown in the figure 3.1.

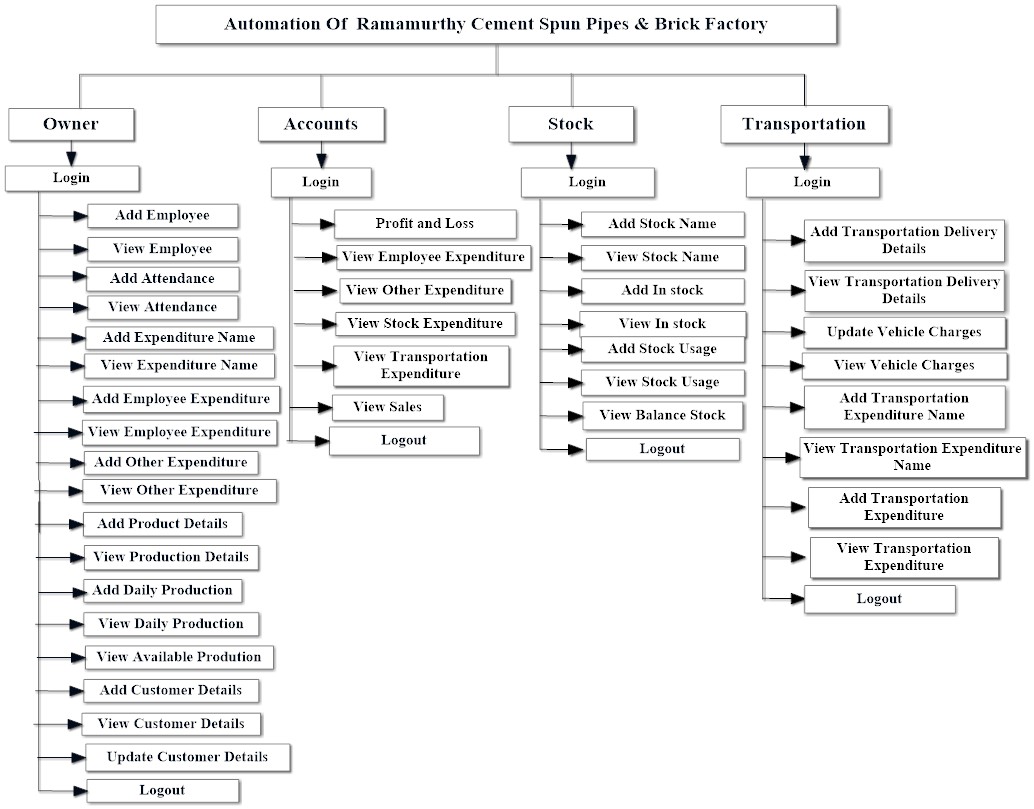


Figure 3.1 Block Diagram

In Automation of Desktop application for Ramamurthy cement spun pipes and bricks factory, owner can add, view and update the details of employee, attendance, employee expenditure, other expenditures etc. This shows the entire block description of the project.

3.2 **DATA FLOW DIAGRAMS**:

Data flow diagram (DFD) is a graphical representation of “flow” of data through an information system, modeling its process concepts. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFD’s can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It doesn’t show information about timing of processes, or information about whether processes will operate in sequence or parallel.

A DFD is also called as “bubble chart”.

**DFD Symbols:**

In the DFD, there are four symbols

* A square define a source or destination of system data.
* An arrow indicates dataflow. It is the pipeline through which the information flows.
* A circle or a bubble represents transforms dataflow into outgoing dataflow.
* An open rectangle is a store, data at reset or at temporary repository of data.

**Dataflow:** Data move in a specific direction from an origin to a destination.

**Process:** People, procedures or devices that use or produce (Transform) data. The physical component is not identified.

**Sources**: External sources or destination of data, which may be programs, organizations or other entity.

**Data store**: Here data is stored or referenced by a process in the system.s #

In our project, we had built the data flow diagrams at the very beginning of business process modeling in order to model the functions that our project has to carry out and the interaction between those functions together with focusing on data exchanges between processes.

**3.2.1 CONTEXT LEVEL DFD:**

A Context level Data flow diagram created using select structured systems analysis and design method (SSADM). This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, unless they are “owned” by external systems, e.g. are accessed by but not maintained by this system, however, these are often shown as external entities. The Context level DFD is shown in below figure.3.2.1.

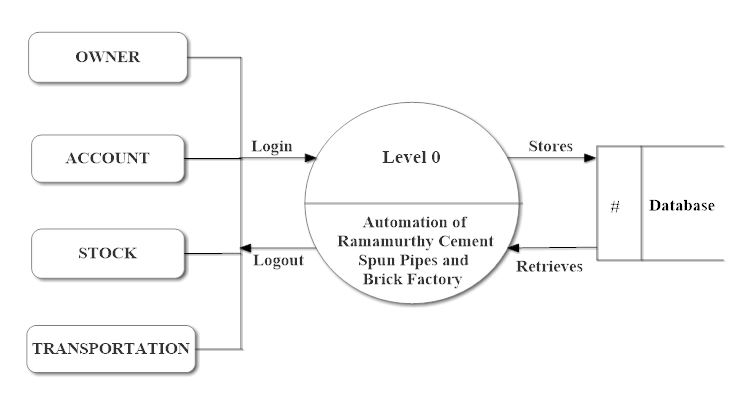


Figure 3.2.1 Context Level DFD

The Context Level Data Flow Diagram shows the data flow from the application to database and to the system.

**3.2.2 TOP LEVEL DFD:**

A data flow diagram is that which can be used to indicate the clear progress of a business venture. In the process of coming up with a data flow diagram, the level one provides an overview of the major functional areas of the undertaking. After presenting the values for most important fields of discussion, it gives room for level two to be drawn.

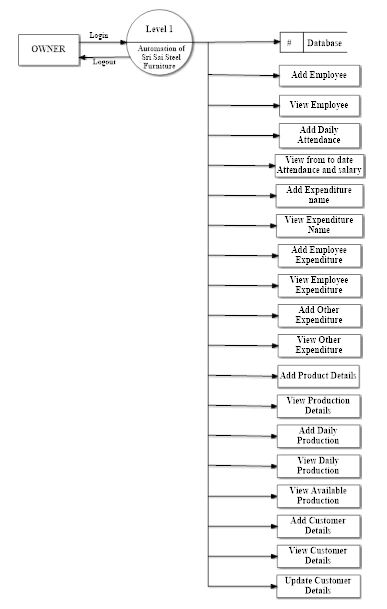


Figure3.2.2.1 Top Level DFD Owner Module

After the successful login, owner can perform all the operations which are shown in the figure 3.2.2.1.

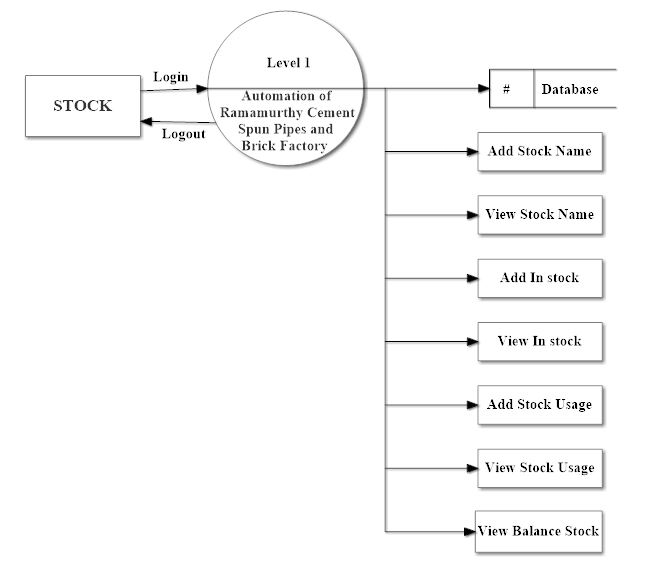


Figure3.2.2.2 Top Level DFD Stock Module

In stock module, the user should login by providing their valid username and password.

User can access or perform all the operations which are shown in the figure3.2.2.2.

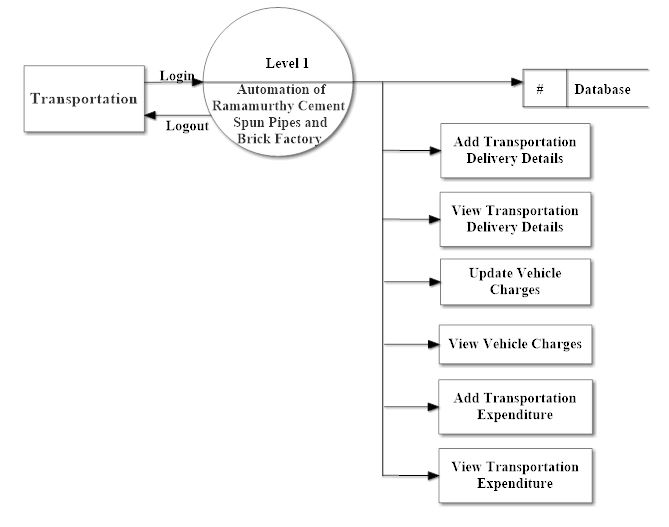


Figure3.2.2.3 Top Level DFD Transportation Module

User should provide his valid username and password to authenticate into this application. After the successful login user can perform the operations which are displayed in the figure3.2.2.3

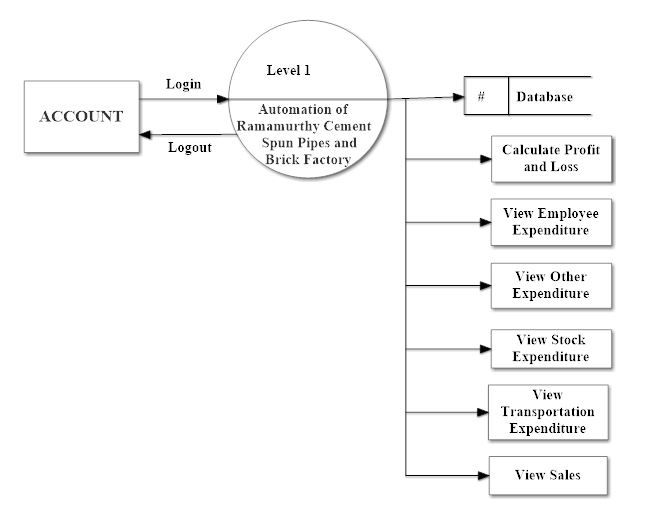


Figure3.2.2.4 Top Level DFD Accounts Module

After the successful login, user can perform operations related to database. The operations which can be performed by the user can be viewed in the figure3.2.2.4.

The Top Level DFD describes the outline of the project. Here the modules owner, stock, transportation and accounts are connected to the database.

**3.2.3 DETAILED LEVEL DFD:**

A detailed level data flow diagram shows the overall context of the system and its operating environment and shows the whole system in detailed form, i.e. by seeing the DFD we can say that what happens after each and every step. The purpose of this level is to show the major and high-level processes of the system and their interrelation. A process model will have one, and only one, decomposed in lower-level DFD.A decomposed in lower-level DFD diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the decomposed in lower-level DFD.

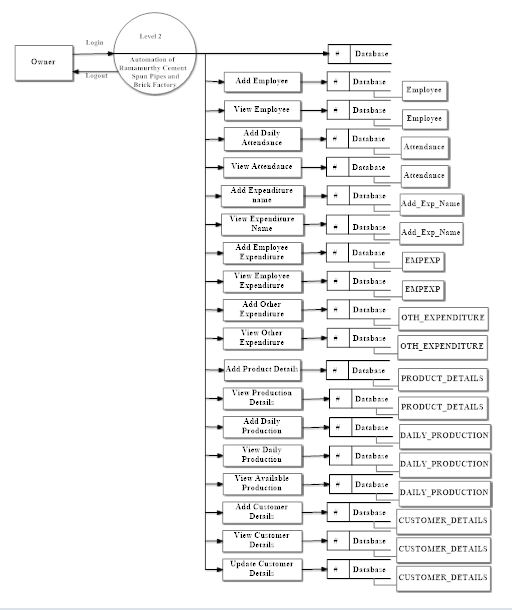


Figure3.2.3.1 Detailed Level DFD for Owner

Owner should login by providing the valid username and password. All the operations which are displayed in the above figure3.2.3.1 such as adding, updating, viewing can be performed and all these are linked to the database.

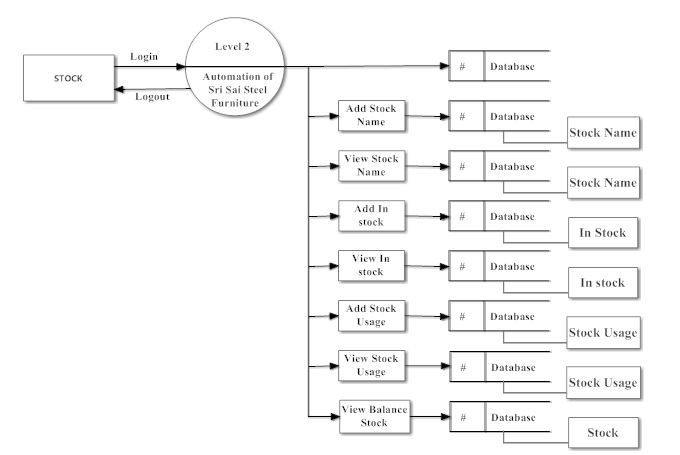


Figure3.2.3.2 Detailed Level DFD for Stock

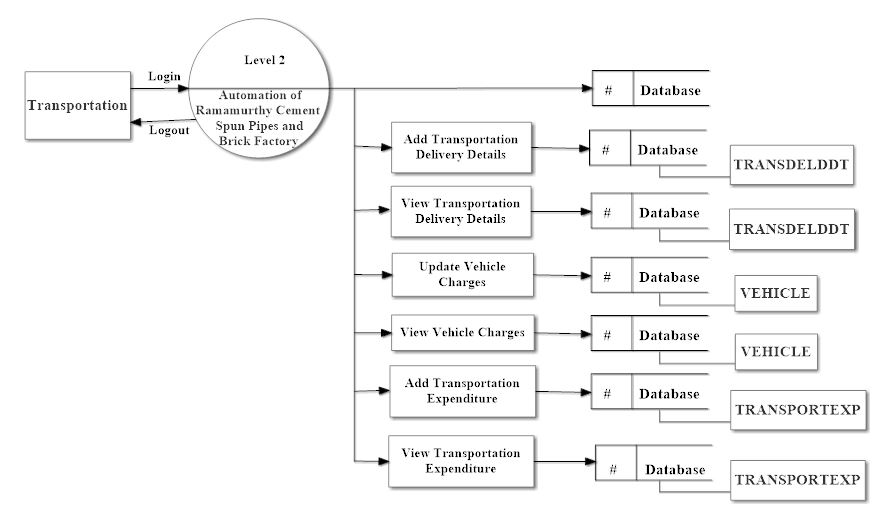
User should login into the database to perform all the operations such as adding stock and view as shown in figure 3.2.3.2. 

Figure3.2.3.3 Detailed Level DFD for Transportation

User should login into the database by providing the valid username and password, after the successful login user can add, view, and update the charges or expenditure for the transportation as shown in figure 3.2.3.3.

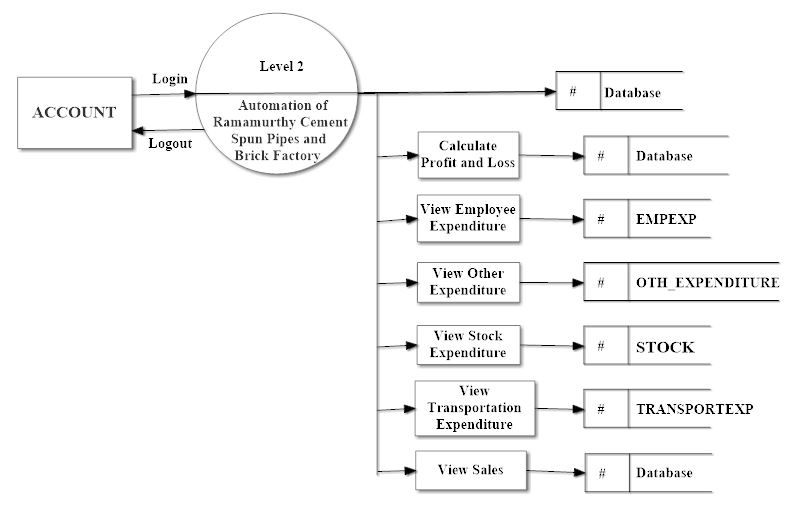


Figure 3.2.3.4 Detailed Level DFD for Account

In the account module user should login into database by specifying the valid username and password. Here user can perform operations such as calculating profit or loss, viewing of employee, stock, and other expenditures as shown in figure 3.2.3.4 .

**3.3 ER DIAGRAM**

In software engineering, an entity-relationship model(ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way the main components of ER models are entities (things) and the relationships that can exist among them[3].

Elements in ER Diagram

There are three basic elements in an ER Diagram

* Entity
* Attribute
* Relationship

There are more elements which are based on the main elements. They are weak entities, multivalued attribute, derived attribute, weak relationship and recursive relationship. Cardinality and ordinality are two other notations used in ER diagrams[4].

**Entity:** An entity can be a person, place, event or object that is relevant to a given system. They are represented by a rectangle and named using nouns.

**Weak Entity:** A weak entity is an entity that depends on the existence of another entity. It can be defined as an entity that cannot be identified by its own attributes.

**Attribute:**An attribute is a property, or characteristic of an entity, relationship, or another attribute.

**Multivalued Attribute:** If an attribute can have more than one value it is called an multivalued attribute.

**Derived Attribute:** An attribute based on another attribute.

**C:\Users\Gouthami\Desktop\Untitled.jpg**

**Relationship:** A relationship describes how entites interact.

**Cardinality:** Cardinality specifies how many instances of an entity relate to one instance of another entity. Cardinality specifies the maximum number of relationships[5].

* one,and only one(mandatory)
* many (zero or more-optional)
* 1...\* - one or more (mandatory)
* 0...1 - zero or one (optional)
* (1,n) - one or more (mandatory)
* (0,n) - zero or more (optional)
* (1,1) - one and only one (mandatory)
* **Ordinality:** Ordinality describes the relationship as either mandatory or optional. Ordinality specifies the absolute minimum number of relationships.
* **Database Keys:** Keys are used to establish and identify relation between tables. They also ensure that each record within atable can be uniquely identified by combination of one or more fields within a table.
* **Super Key:** Super Key is defined as a set of attributes within a table that uniquely identifies each record with in a table. Super Key is a superset of Candidate key.
* **Candidate Key:** Candidate keys are defined as the set of fields from which primary key can be selected. It is an attribute or set of attributes that can act as a primary key for a table to uniquel identify each record in that table.
* **Primary Key:** Primary key is a candidate key that is most appropriate to become main key of the table. It is a key that uniquely identify each record in a table.
* **Composite Key:** Key that consist of two or more attributes that uniquely identify an entity occurance is called composite key. But any attribute that makes up the composite key is not a simple key in its own.
* **Foreign Key:** A foreign key is a field(or collection of fields) in one table that uniquely identifies a row of another table.

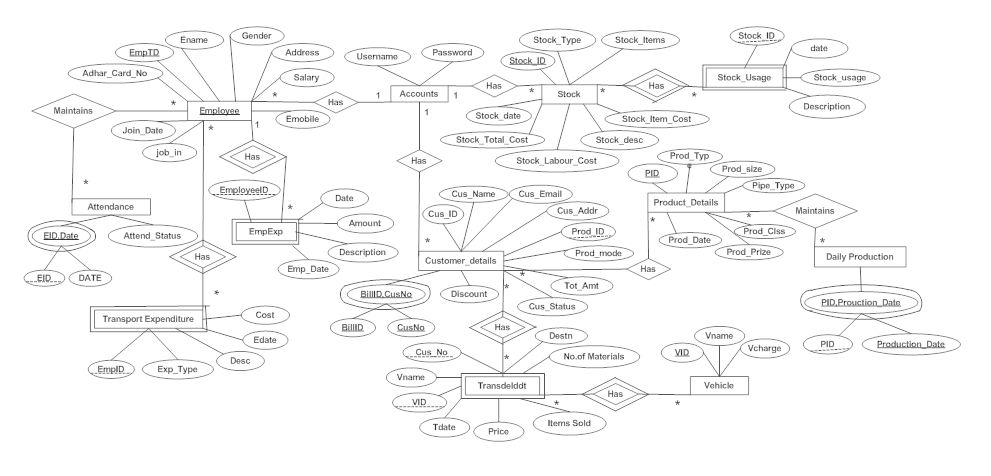
****

Figure 3.3 Entity Relationship Diagram.

The entity relationship diagram shows all the details and the operations which can be performed by the user. User or owner should login into the database to access or edit the details of stock or account or transportation or any other details. Every module and its operations are shown in the figure3.3.

**3.4 UML Diagrams**

The Unified Modeling Language (UML)[6] is a Standard language for specifying, visualizing, constructing and documenting the software system and its components. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows [7].

* **User Model View**

1. This view represents the system from the users perspective.
2. The analysis representation describes a usage scenario from the end-users perspective.

* **Structural Model View**

1. In this model the data and functionality are arrived from inside the system.
2. This model view models the static structures.

* **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

**3.4.1 Use Case Diagram**

Use case diagrams are one of the five diagrams in the UML for modeling the dynamic aspects of the systems (activity diagrams, sequence diagram, state chart diagram, collaboration diagram are the four other kinds of diagrams in the UML for modeling the dynamic aspects of systems).use case diagram are central to modeling the behavior of the system, a sub-system, or a class. Each one shows a set of use cases and actors and relations.

**3.4.1.1 Use Case Diagram For Owner Module:**

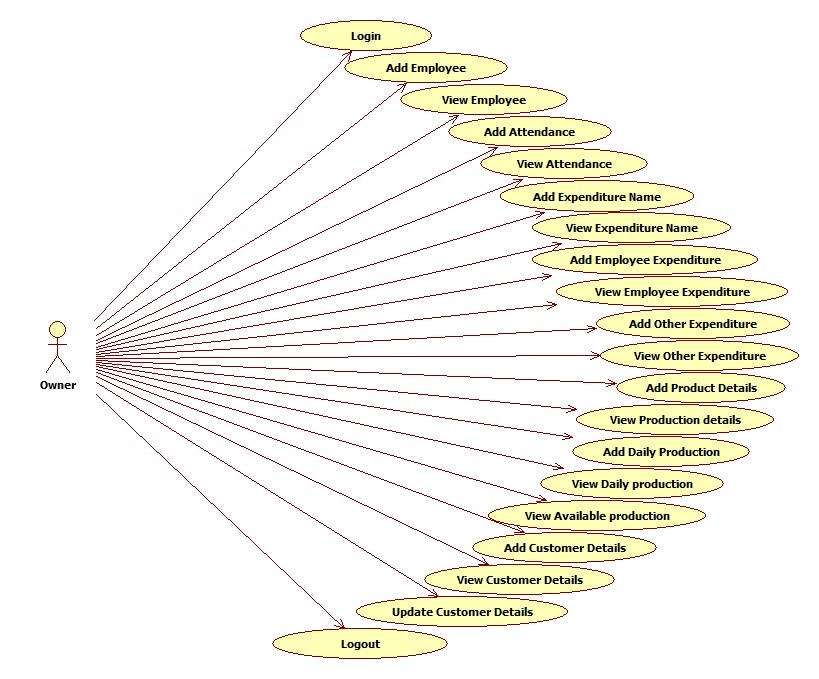


Figure 3.4.1.1 Use case Diagram For owner module

To perform any of the operations such as adding, viewing or updating of the employee, customer, expenditure, daily production etc owner should login into the application as shown in figure 3.4.1.1.

**3.4.1.2 Use case Diagram for Stock Module:**

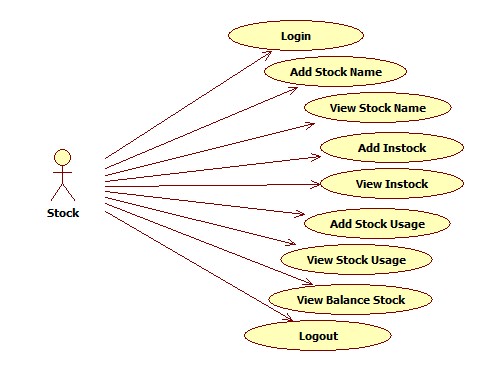


Figure 3.4.1.2 Use case Diagram for Stock Module

The above figure is use case diagram for stock module, user should login into the application to add or view the stock details as shown in figure 3.4.1.2.

**3.4.1.3 Use case Diagram For Transportation Module:**

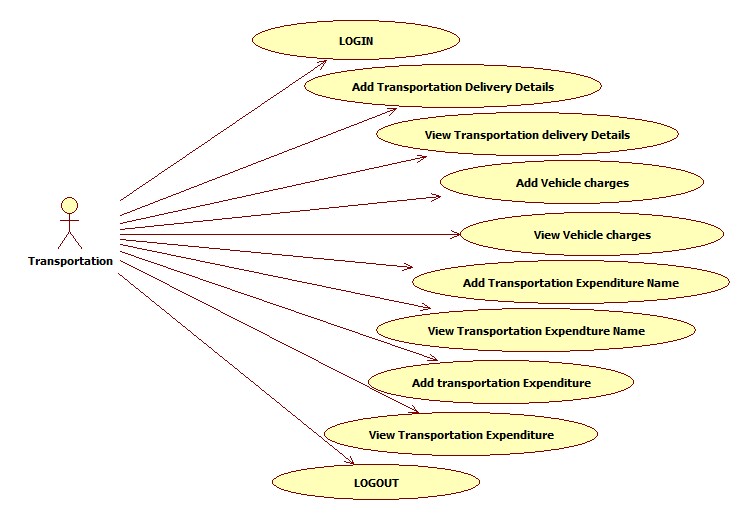


Figure 3.4.1.3 Use case Diagram For Transportation Module

After the successful login into transportation page, user can perform operations like adding or viewing of the charges, delivery details etc as shown in figure 3.4.1.3.

**Figure 3.4.1.4 Use case Diagram For Accounts Module:**

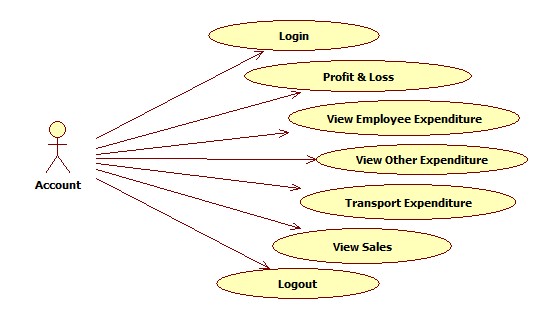


Figure 3.4.1.4 Use case Diagram For Accounts Module

After the successful login into accounts page, user can perform operations like profit and loss and view employee expenditure, other expenditure, transportation expenditure and view sales as shown in figure 3.4.1.4.

**3.4.2 Class Diagram**

A Class Diagram shows a set of classes, interfaces and collaborations and their relationships. These diagrams are most common diagram in modeling object oriented systems.

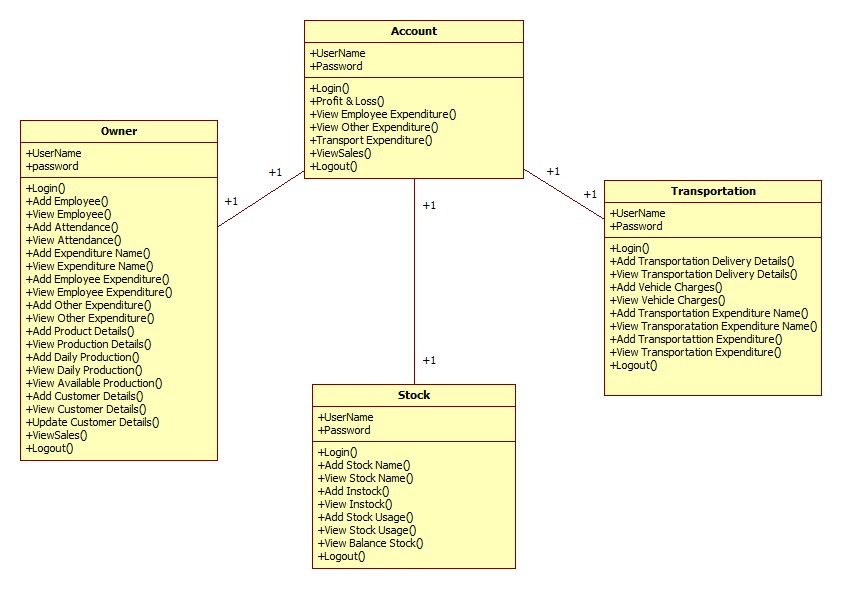


Figure 3.4.2 Class diagram

The above figure 3.4.2 is class diagram of Ramamurthy cement spun pipes and brick factory. The class diagram represents set of classes, interfaces, collaborations, and their relationships.

**3.4.3 Sequence Diagram**

Sequence diagram is an interaction diagram which is focuses on the time ordering of messages. It shows a set of objects and messages exchanged between these objects. This diagram illustrates the dynamic view of a system.

**3.4.3.1 Sequence diagram for Owner Module:**

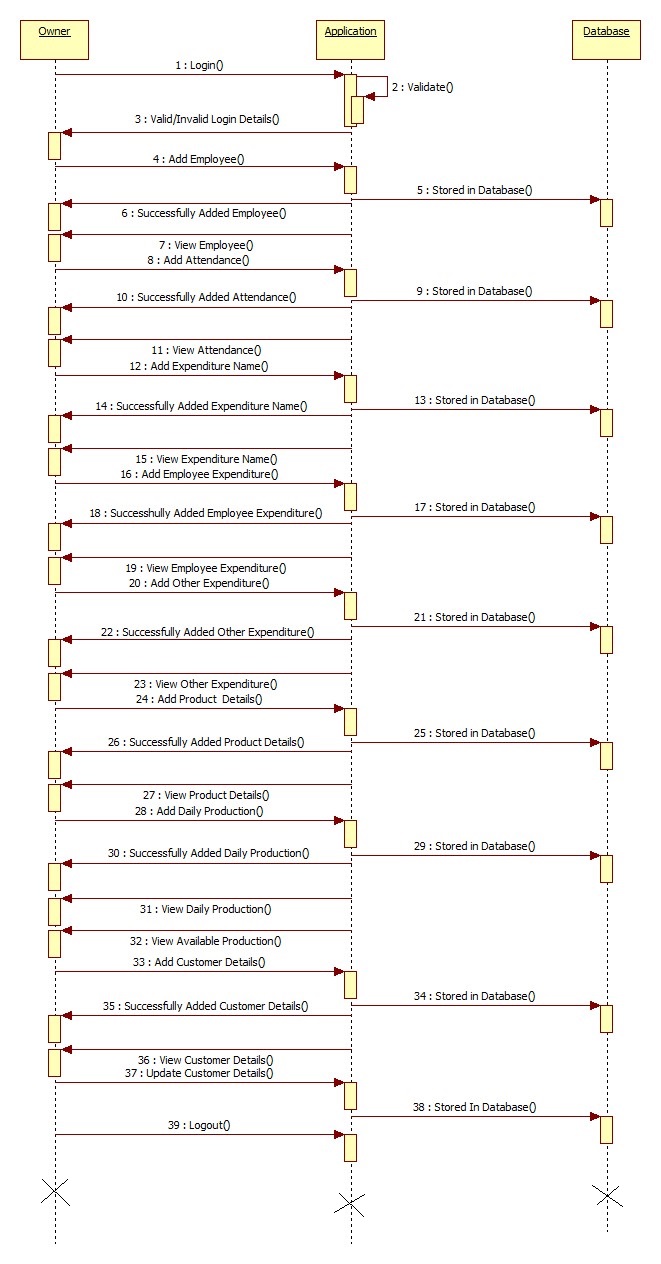


Figure 3.4.3.1 Sequence diagram for Owner Module

Above figure 3.4.3.1 is the sequence diagram for owner module, here all the operations of the owner module is shown in the sequential order.

**3.4.3.2 Sequence diagram for stock module:**

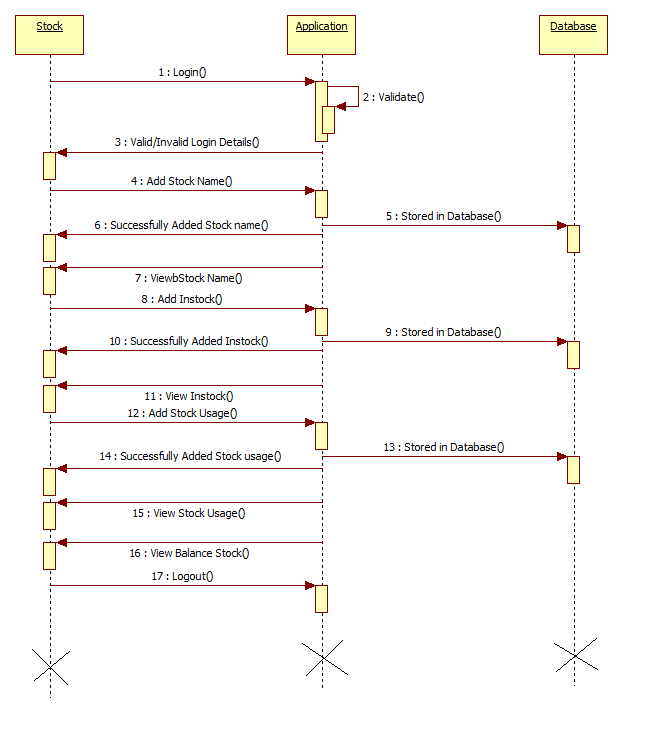


Figure 3.4.3.2 Sequence diagram for Stock Module

The figure 3.4.3.2 shows the sequence diagram for Stock Module. In this the user can perform sequence of actions.

**3.4.3.3 Sequence diagram for Transportation Module:**

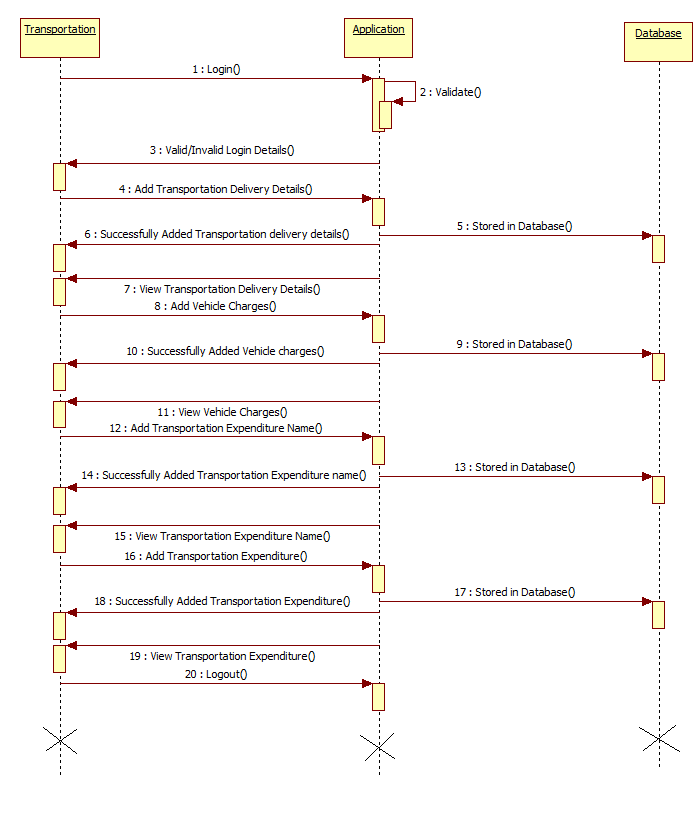


Figure 3.4.3.3 Sequence diagram for Transportation Module

The figure 3.4.3.3 shows the sequence diagram for Transportation Module. In this the user can perform sequence of actions.

**3.4.3.4 Sequence diagram for Accounts Module:**

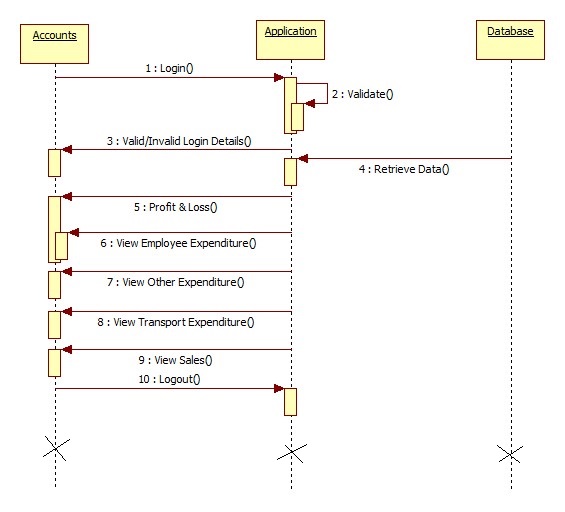


Figure 3.4.3.4 Sequence diagram for Accounts Module

The figure 3.4.3.4 shows the sequence diagram for Accounts Module. In this the User can perform sequence of actions.

**3.4.4 Collaboration Diagram**

Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Collaboration diagram and sequence diagram are isomorphic.

**3.4.4.1 Collaboration diagram for owner module:**

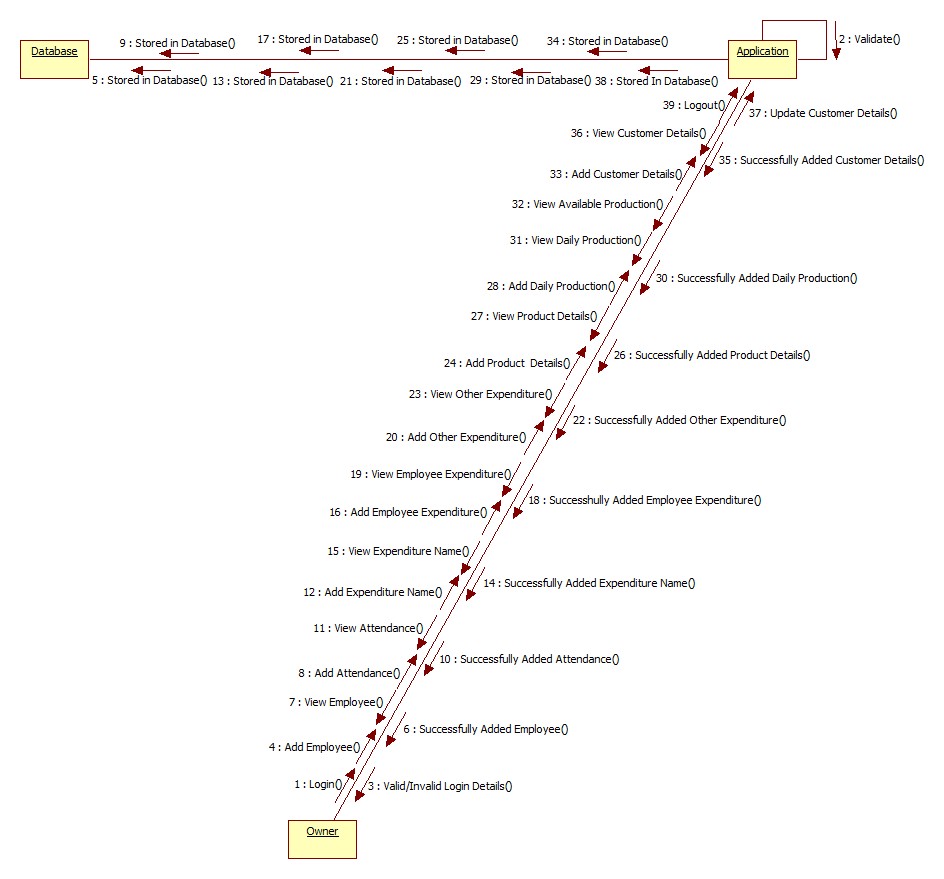
****

Figure 3.4.4.1 Collaboration Diagram for Owner Module

The figure 3.4.4.1 shows the Collaboration diagram for Owner Module. In this the Owner can perform Collaboration of actions.

**3.4.4.2 Collaboration diagram for stock module:**

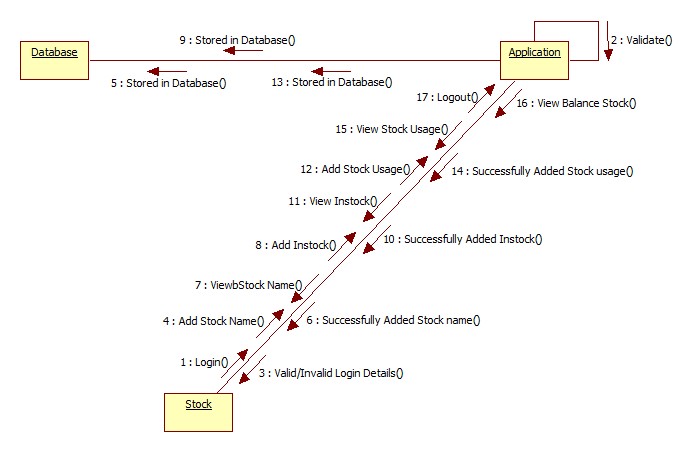


Figure 3.4.4.2 Collaboration Diagram for Stock Module

The figure 3.4.4.2 shows the Collaboration diagram for Stock Module. In this the user can perform Collaboration of actions.

**3.4.4.3 Collaboration diagram for Transportation Module:**

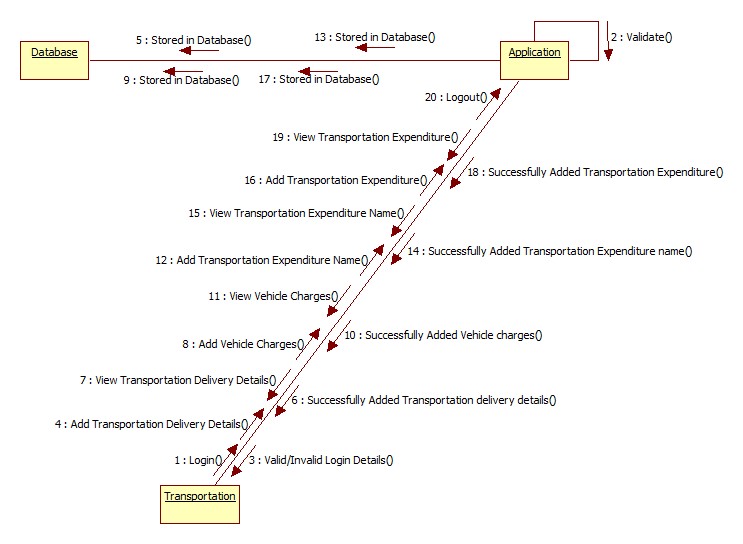


Figure 3.4.4.3 Collaboration Diagram for Transportation module

The figure 3.4.4.3 shows the Collaboration diagram for Transportation Module. In this the user can perform Collaboration of actions.

**3.4.4.4 Collaboration diagram for Accounts Module:**

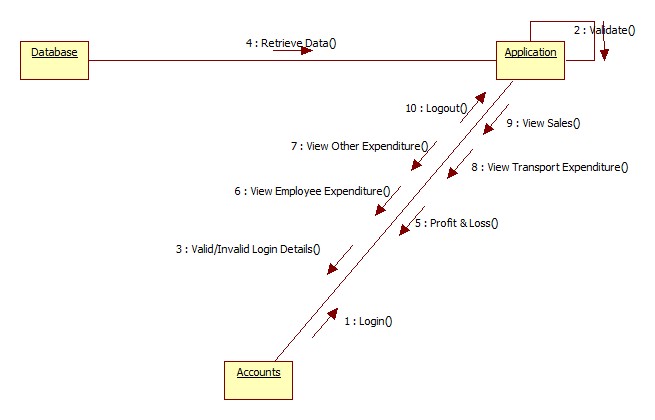


Figure 3.4.4.4 Collaboration Diagram for Accounts module

The figure 3.4.4.4 shows the Collaboration diagram for Accounts Module. In this the user can perform Collaboration of actions.

**3.4.5 Activity Diagram**

An Activity diagram shows the flow from activity to activity within a system it emphasizes the flow of control among objects**.**

**3.4.5.1 Activity diagram for Owner Module:**

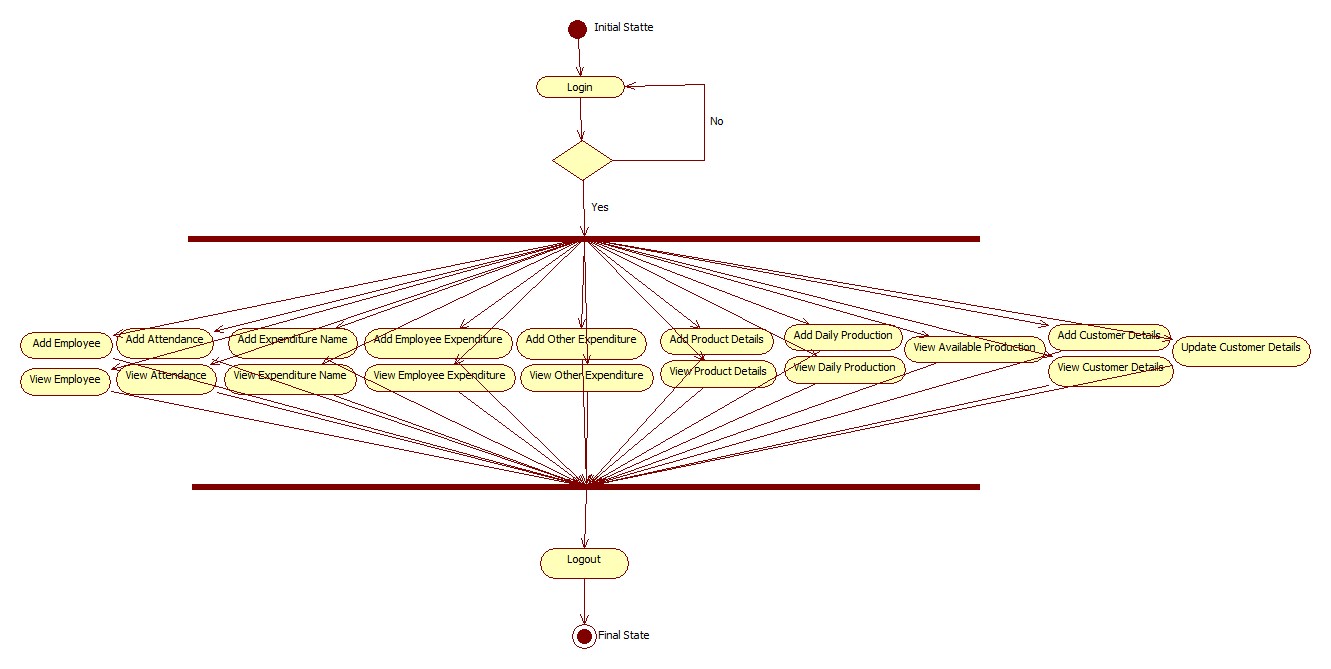


Figure 3.4.5.1 Activity diagram for Owner Module

The figure3.4.5.1 shows the Activity diagram for Admin. In this the Admin can perform activity of actions.

**3.4.5.2 Activity diagram for Stock Module:**

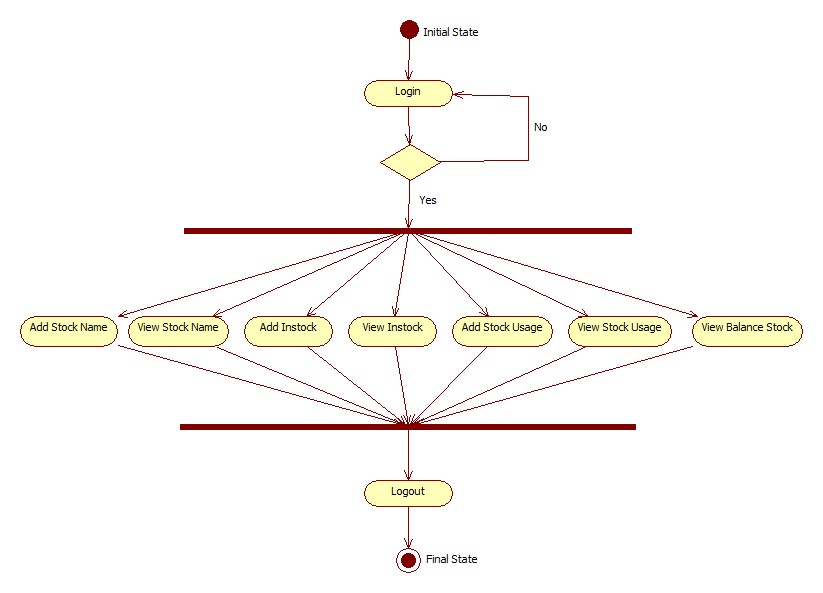


Figure 3.4.5.2 Activity Diagram For Stock Module

The figure3.4.5.2 shows the Activity diagram for Admin. In this the Admin can perform activity of actions.

**3.4.5.3 Activity diagram for Transportation Module:**

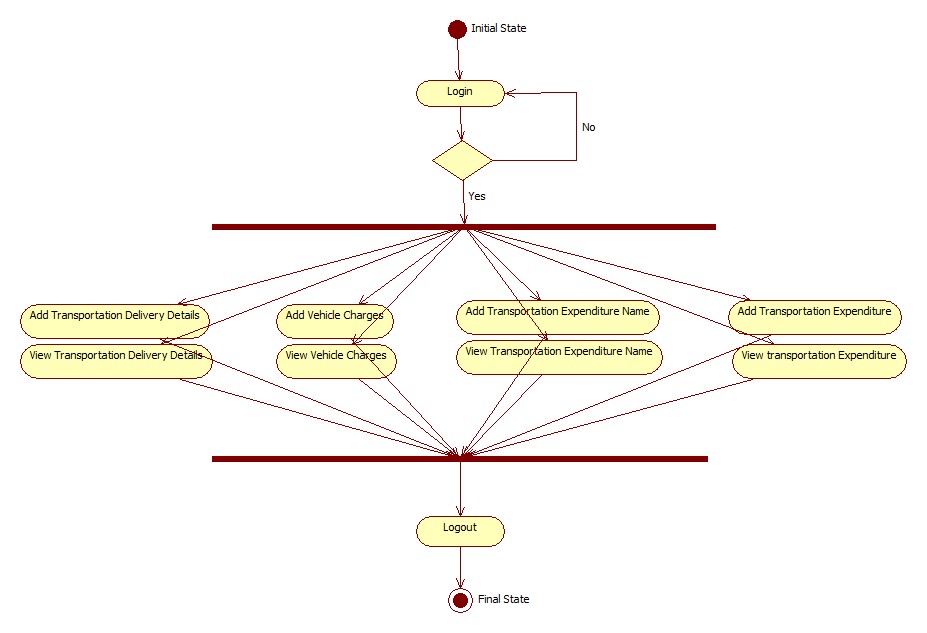


Figure 3.4.5.3 Activity Diagram for Transportation Module

The figure3.4.5.3 shows the Activity diagram for Transportation Module. In this the user can perform activity of actions.

**3.4.5.4 Activity diagram for Accounts Module:**

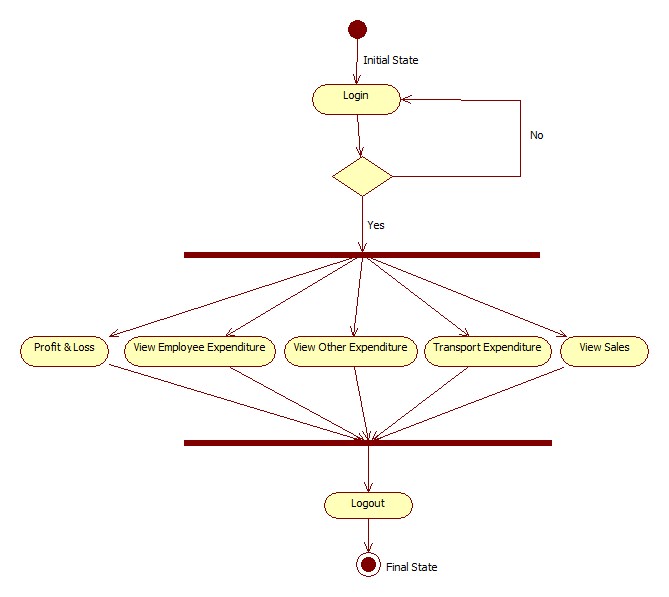


Figure 3.4.5.4 Activity Diagram for Accounts Module

The figure 3.4.5.4 shows the Activity diagram for Accounts Module. In this the Admin can perform activity of actions.

**3.5 DATA DICTIONARY:**

The design of tables which stores details of employee, attendance, daily production, stock, transportation details etc.

**3.5.1 Login Details:**

The design of table 3.5.1 which stores details about login menu with list of attributes Username, Password, Saq1, Saa1, Type.

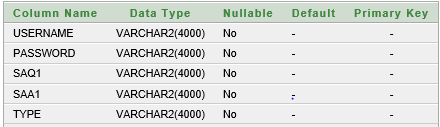
****

Table 3.5.1 Database For Login Details

**3.5.2 Employee Details:**

The design of table 3.5.2 which stores details about employee menu with list of attributes Empid, EName, Emobile, Jobin, Address, Gender, Salary, Aadhaar number, Joindate.

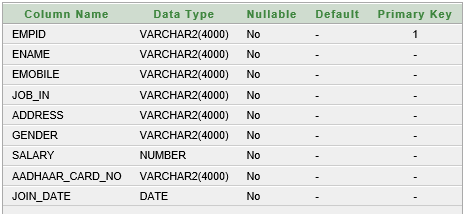


Table 3.5.2 Database For Employee Details

**3.5.3 Employee Attendance:**

The design of table 3.5.3 which stores details about Employee Attendance menu with list of attributes Eid, Edate, Attendstatus.

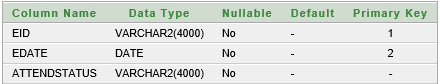


Table 3.5.3 Database For Employee Attendance

**3.5.4 Product Details:**

The design of table 3.5.4 which stores details about product menu with list of attributes Pid, Product type, Pipe type, Product size, Product class, Product price, Product date.

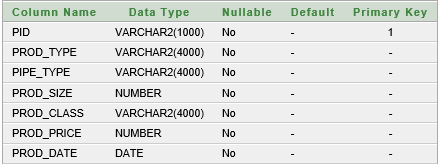


Table 3.5.4 Database For Product Details

**3.5.5 Daily Production:**

The design of table 3.5.5 which stores details about Daily Production menu with list of attributes Pid, Production date.

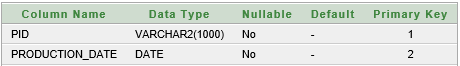


Table 3.5.5 Database For Daily Production

**3.5.6 Stock Name:**

The design of table 3.5.6 which stores details about Stock name with list of attributes Stock\_id and Stock\_Name.

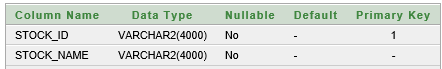


Table 3.5.6 Database For Stock Name

**3.5.7 Stock:**

The design of table 3.5.7 which stores details about Stock menu with list of attributes Stockid, Stock type, Stock items, Stock item cost, Stock description, Stock labor cost, Stock total cost, Stock date.

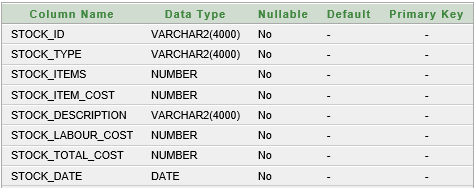


Table 3.5.7 Database For Stock

**3.5.8 Stock Usage:**

The design of table 3.5.8 which stores details about Stock usage menu with list of attributes Stockid, Stock Name, Stock quantity usage, Stock usage date, Stock description.

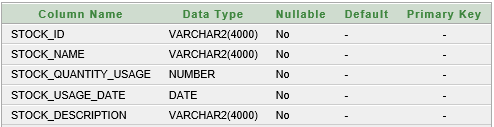


Table 3.5.8 Database For Stock Usage

**3.5.9 Transportation Details:**

The design of table 3.5.9 which stores details about transportation menu with list of attributes Vname, Vnum, Cmbno, Cname, Destn, Nm, Itsl, Pri, Tdate.

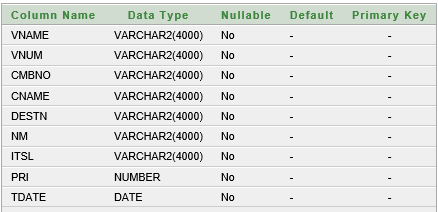


Table 3.5.9 Database For Transportation Details

**3.5.10 Transportation Expenditure:**

The design of table 3.5.10 which stores details about Transportation Expenditure menu with list of attributes Expid, Expname.

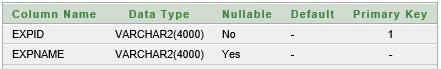


Table 3.5.10 Database For Transportation Expenditure

**3.5.11 Vehicle Charges:**

The design of table 3.5.11 which stores details about vehicle charges menu with list of attributes Vid, Vname, Vcharge.

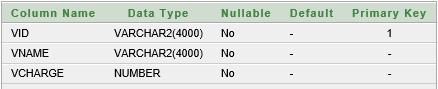


Table 3.5.11 Database For Vehicle Charges

**3.5.12 Employee Expenditure:**

The design of table 3.5.12 which stores details about Employee Expenditure menu with list of attributes Employee id, Name, Amount, Description, Exp date.

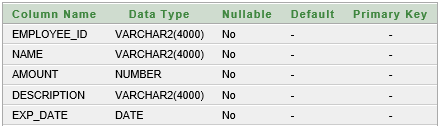


Table 3.5.12 Database For Employee Expenditure

**3.5.13 Adding Expenditure Names:**

The design of table 3.5.13 which stores details about Adding Expenditure Names menu with list of attributes Exp id, Exp name.

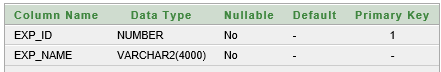


Table 3.5.13 Database For Adding Expenditure Names

**3.5.14 Other Expenditure:**

The design of table 3.5.14 which stores details about Expenditure menu with list of attributes Exp id, Exp type, Exp amount, Exp date, Exp description.

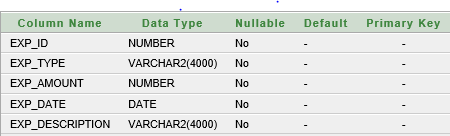


Table 3.5.14 Database For Other Expenditure

**3.5.15 Customer Details:**

The design of table 3.5.15 which stores details about Customer menu with list of attributes Bill id, Cus id, Cus name, Cus num, Cus email, Cus addr, Prod type, Pipe type, Prod size, Prod class, Prod item, Prod mode, Tot amt, Cus status, Cus date, Discount.

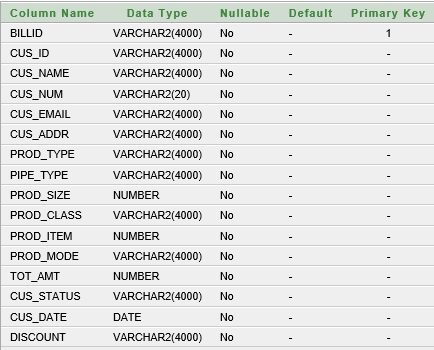


Table 3.5.15 Database For Customer Details

**CHAPTER-4**

**IMPLEMENTATION**

The project is implemented using JAVA technology. The description of this framework is discussed below.

#### 4.1 INTRODUCTION TO JAVA

Initially the language was called as “oak” but it was renamed as “Java[1]” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

* Java is a programmer’s language.
* Java is cohesive and consistent.
* Except for those constraints imposed by the Internet environment, Java gives Mthe programmer, full control.
* Finally, Java is to Internet programming where C was to system programming.

**4.2 JAVA TECHNOLOGY**

Java technology is both a programming language and a platform.

**4.2.1 The Java Programming Language**

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* Simple
* Architecture neutral
* Object oriented
* Portable
* Distributed
* High performance
* Interpreted
* Multithreaded
* Robust
* Dynamic
* Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM[2]. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



**4.2.2 The Java Platform**

* A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.
* The Java platform has two components:
* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)
* You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.
* The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

**4.2.3 What Can Java Technology Do?**

The most common types of programs written in the Java programming language are applets and applications. If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

**The essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

Applets: The set of conventions used by applets.

**Networking:** URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.

**Internationalization:** Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

**Security:** Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.

**Software components:** Known as JavaBeans TM, can plug into existing component architectures.

**Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).

**Java Database Connectivity (JDBCTM):** Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



**4.2.4 How Will Java Technology Change My Life?**

We can’t promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly:** Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write better code:** The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop programs more quickly:** Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java:** You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write once, run anywhere:** Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **Distribute software more easily:** You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

**4.2.5 Importance of java to the internet:**

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Java can be used to create two types of programs.

**Applications and Applets:**

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java’s ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

**4.3 JAVA VIRTUAL MACHINE (JVM):**

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that’s has been generated by the compiler will not corrupt the machine that it’s loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

Overall Description

# Java Source

## Java byte code

# JavaVM

Java

.Class

Figure 4.3. Picture showing the development process of JAVA Program

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The. Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

**4.4 JAVA ARCHITECTURE:**

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

**4.4.1 COMPILATION OF CODE:**

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

Compiling and interpreting Java Source Code During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be a Intel Pentium Windows 95 or Sun SARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

**4.5 INTRODUCTION TO SWINGS**

Swing is a set of classes that provides more powerful and flexible functionalities when compared to AWT components. And each component in swing has more capabilities. For example a button can have an image and a text string with it.

AWT components are heavy weight, but Swing components are light weight because they are written entirely in Java. They are platform independent. They have the same look and feel on different operating systems or for each platform.

**4.5.1 COMPONENTS:**

The classes used for design of Win GUI and Lin Gui of this project are from javax.swing package. This package includes the components as follows:

* **JPanel** is Swing's version of the AWT class Panel and uses the same default layout, Flow Layout. JPanel is descended directly from JComponent.
* **JFrame** is Swing's version of Frame and is descended directly from that class. The components added to the frame are referred to as its contents; these are managed by the content Pane. To add a component to a JFrame, we must use its content Pane instead.
* **JInternalFrame** is confined to a visible area of a container it is placed in. It can be confined, maximized and layered.
* **JWindow** is Swing's version of Window and is descended directly from that class. Like Window, it uses Border Layout by default.
* **JDialog** is Swing's version of Dialog and is descended directly from that class. Like Dialog, it uses Border Layout by default. Like JFrame and JWindow, JDialog contains a root Pane hierarchy including a content Pane, and it allows layered and glass panes. All dialogs are modal, which means the current thread is blocked until user interaction with it has been completed. JDialog class is intended as the basis for creating custom dialogs; however, some of the most common dialogs are provided through static methods in the class JOptionPane.
* **JLabel**, descended from JComponent, is used to create text labels.
* The abstract class Abstract Button extends class JComponent and provides a foundation for a family of button classes, including JButton.
* **JTextField** allows editing of a single line of text. New features include the ability to justify the text left, right, or center, and to set the text's font.
* **JPasswordField** (a direct subclass of JTextField) you can suppress the display of input. Each character entered can be replaced by an echo character. This allows confidential input for passwords, for example. By default, the echo character is the asterisk.
* **JTextArea** allows editing of multiple lines of text. JTextArea can be used in conjunction with class JScrollPane to achieve scrolling. The underlying JScrollPane can be forced to always or never have either the vertical or horizontal scrollbar.
* **JButton** is a component the user clicks to trigger a specific action.
* **JRadioButton** is similar to JCheckbox, except for the default icon for each class. A set of radio buttons can be associated as a group in which only one button at a time can be selected.
* **JCheckbox** is not a member of a checkbox group. A checkbox can be selected and deselected, and it also displays its current state.
* **JComboBox** is like a drop down box. You can click a drop-down arrow and select an option from a list. For example, when the component has focus, pressing a key that corresponds to the first character in some entry's name selects that entry. A vertical scrollbar is used for longer lists.
* **JList** provides a scrollable set of items from which one or more may be selected. JList can be populated from an Array or Vector. JList does not support scrolling directly; instead, the list must be associated with a scroll pane. The view port used by the scroll pane can also have a user-defined border. JList actions are handled using ListSelectionListener.
* **JTabbedPane** contains a tab that can have a tool tip and a mnemonic, and it can display both text and an image.
* **JToolbar** contains a number of components whose type is usually some kind of button which can also include separators to group related components within the toolbar.
* **Flow Layout** when used arranges swing components from left to right until there's no more space available. Then it begins a new row below it and moves from left to right again. Each component in a Flow Layout gets as much space as it needs and no more.
* **Border Layout** places swing components in the North, South, East, West and center of a container. You can add horizontal and vertical gaps between the areas.
* **Grid Layout** is a layout manager that lays out a container's components in a rectangular grid. The container is divided into equal-sized rectangles, and one component is placed in each rectangle.
* **Grid Bag Layout** is a layout manager that lays out a container's components in a grid of cells with each component occupying one or more cells, called its display area. The display area aligns components vertically and horizontally, without requiring that the components be of the same size.
* **JMenubar** can contain several JMenu's. Each of the JMenu's can contain a series of JMenuItem’s that you can select. Swing provides support for pull-down and popup menus.
* **Scrollable JPopupMenu** is a scrollable popup menu that can be used whenever we have so many items in a popup menu that exceeds the screen visible height.

**4.6 INTRODUCTION TO NET BEANS**

Net Beans provides you a modern slim PHP IDE. Very fast (even in larger projects) and good code completion, file/code/property explorer and excellent HTML/CSS/JavaScript support. Subversion and debug support works almost out of the box as we will see below. Current version of Net Beans is 6.8. Version 6.9 with a lot of small goodies, ex. ability to detect changes outside Net Beans so far had been a bit of a mystery, ability to refactor inline css codes to external such etc., is around the corner (eta early June 2010, currently on beta status and beta is quite stable). As a last note about version 6.9, since the installation is made to version specific folders, if you want to give 6.9beta a run you can do so without interfering with your 6.8 setup.

**4.6.1 Installation**

**4.6.1.1 Installation on Windows**

Prerequisite: to be able to install and run Net Beans you would need Java 6 or 5. If you don't have that installed, download it from http://java.sun.com/javase/downloads/ widget/jdk6.jsp.

1. Download Net Beans (http://netbeans.org/downloads/index.html) and make sure to download the PHP version. The installation is straight forward and amounts to clicking through the installation wizard which defaults to correct and workable values.

**4.6.1.2 Installation on Linux: Ubuntu 10.04**

1. Open the terminal and install Net beans (Java) version with the following command: sudo apt-get update &&sudo apt-get install net beans

2. Start Net Beans (menu Applications > Programming > Net Beans IDE)

3. Install the PHP plug-in for Net beans: Tools > Plug-in > Available Plug-in > select PHP and install it.

**4.7 ODBC**

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data source.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**4.8 JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

**4.8.1 JDBC Goals**

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java. The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

**4.8.1.1 SQL Level API**

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

**4.8.1.2 SQL Conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

1. JDBC must be implemental on top of common database interfaces  
The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

2. Provide a Java interface that is consistent with the rest of the Java system.Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

3. Keep it simple

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

4. Use strong, static typing wherever possible

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

5. Keep the common cases simple

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking and for dynamically updating the cache table we go for MS Access database. Java have two things: a programming language and a platform.

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

**JavaProgram**

**Compilers**

**Interpreter**

**My Program**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

**4.9 FRONT END DESIGN**

The home page is used to facilitate navigation to other pages on the site by providing links. This page can be viewed by admin. This is home page of Ramamurthy cement spun pipes and brick factory as shown figure 4.2.1.



Figure: 4.2.1 Screenshot of Home page

After successful login the admin can view home page. In which it contains menu, the menu items are of add, view and customer as shown in figure 4.2.2

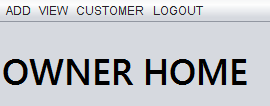
****

Figure: 4.2.2 Screenshot of Administrator’s homepage

The following screenshot is employee form. This form consists of the text fields where every text field is mandatory to be filled according to their particulars as shown in figure 4.2.3

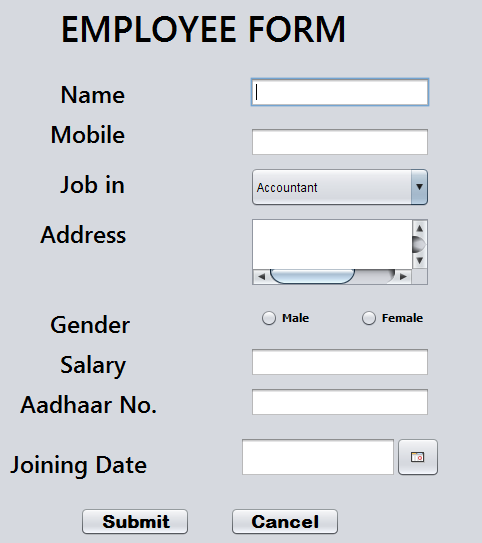


Figure 4.2.3 Screenshot of Employee Form

In the following form, the daily attendance of the employee is updated. In this form when the owner selects employee id, the employee name and employee mobile number will be auto generated. The form is shown in figure 4.2.4.

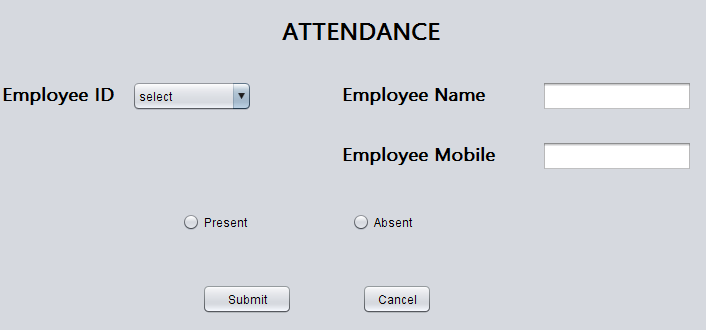


Figure 4.2.4: Screenshot of Employee Attendance

The below screenshot is about the adding of employee expenditure. In this form employee id, employee name, amount is the particulars. After selecting the employee id the name is auto generated. The amount field and description is updated by the owner as shown in figure 4.2.5.

****

Figure 4.2.5: Screenshot of Employee Expenditure

The below screenshot will show other expenditure form particulars and fields. These details are provided by the owner accordingly and are updated by clicking on the submit button. The screenshot is shown in figure 4.2.6.

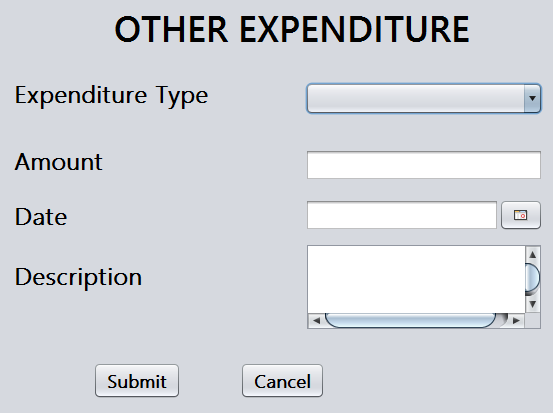


Figure 4.2.6: Screenshot of Other Expenditure

In the following screenshot Add Product Details form is displayed. In this form the product type is provided with the drop down menu with this the owner can easily select the type of product and the size of the product should also be updated. Class of the product is also provided with drop down menu and the price is fixed by the owner. Figure 4.2.7 is the screenshot of the Add Product Details.

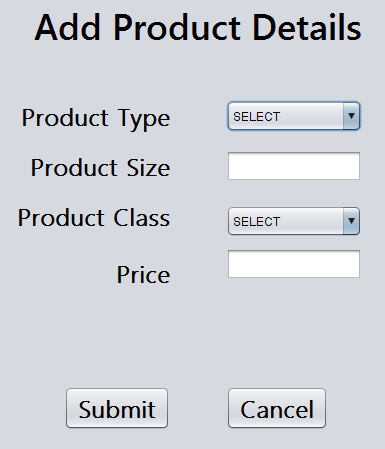


Figure 4.2.7: Screenshot of Add Product Details

This form consists the details of the daily production, the form is shown as an image so that we cannot edit it here. Product type, pipe type, product class, and product size are provided with the drop down menu to perform faster operations. The number of items is to be updated by the owner. Figure 4.2.8 shows the daily production form.

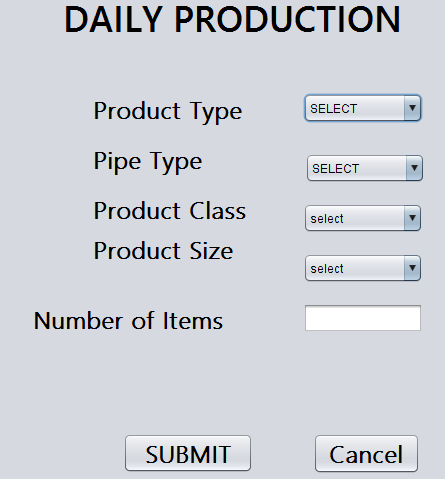


Figure 4.2.8: Screenshot of Daily Production

The following screenshot is about the Customer Details form. The form consists of the particulars like contact number, customer name, Email id, and other details of the product which has been consumed by the customer. Type of payment, amount and all other fields are provided. In this form all these fields are mandatory and it is updated by the owner. Figure 4.2.9 is the screenshot of Customer Details.

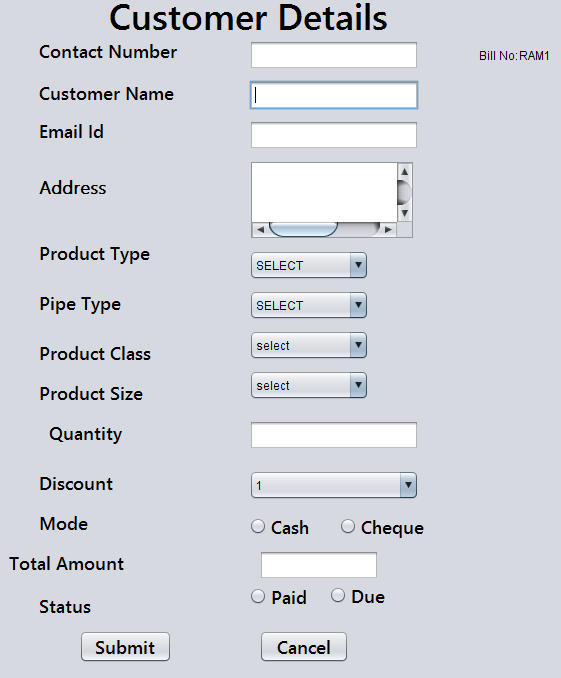


Figure 4.2.9: Screenshot of Adding Customer Details

The following screenshot is about the Add In Stock form. In this form the In stock details are updated by the owner. Figure 4.2.10 is about add in stock form which is shown below.

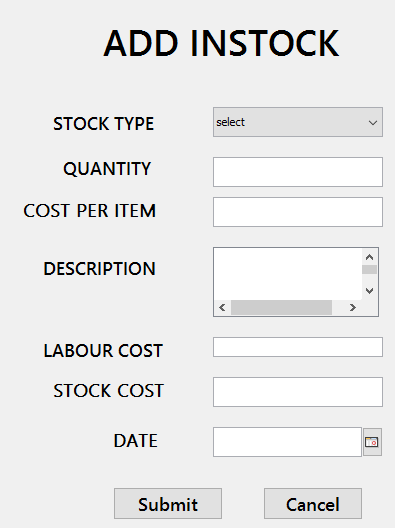


Figure 4.2.10: Screenshot of In-stock

The following screenshot is about adding stock usage. In this the usage of stock is updated, particulars such as stock name, quantity used on which day and for what purpose they are used. Figure 4.2.11 is the screenshot of Add Stock Usage.

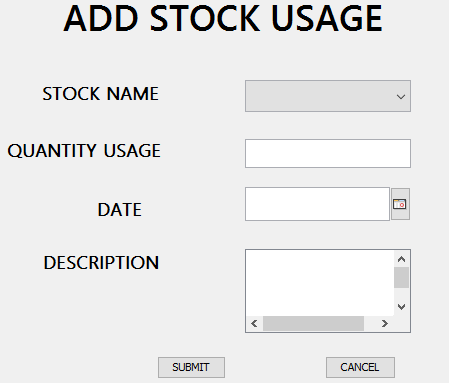


Figure 4.2.11: Screenshot of Add Stock Usage.

The below figure is the screenshot of Adding Transportation Expenditure Name form. In this form, expenditure name field should be updated by the owner and it is mandatory. The screenshot is shown in Figure: 4.2.12.

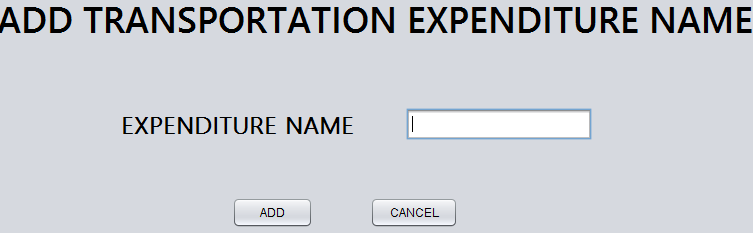


Figure 4.2.12: Screenshot of Adding Transportation Expenditure Name

The below screenshot will provide the details of Transportation Expenditure. This form consists of details like cost, date, description and the type of expenditure. Figure 4.2.13 is the screenshot of Transportation Expenditure form.

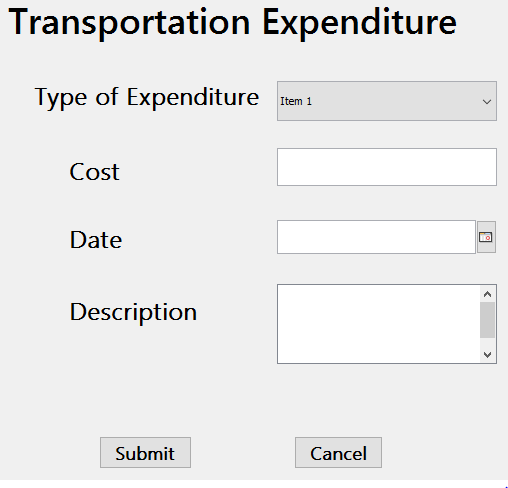


Figure 4.2.13: Screenshot of Transportation Expenditure

The following figure is screenshot of Transportation Delivery Details form. This form has the fields according to their particulars such as vehicle number, vehicle name and other details those all are shown in the following screenshot. These all fields are mandatory and are to be filled by the owner. Figure 4.2.14 is the screenshot of Transportation Delivery Details form.

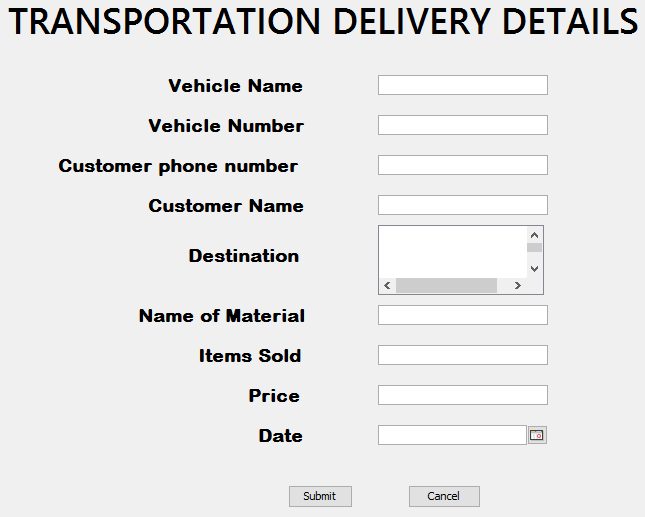


Figure 4.2.14: Screenshot of Transportation Delivery Details

Following screenshot is about the vehicle details updating form. In this form we have the fields such as vehicle name and cost. Owner should select the type of vehicle in drop down menu and the cost field should be filled. Figure 4.2.15 is the screenshot of Update Vehicle Details form.

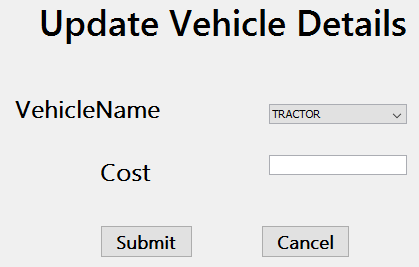


Figure 4.2.15: Screenshot of Update Vehicle Details

The below figure is screenshot of Profit and Loss form. With the help of this form the owner can easily view and print the details, by choosing from and to date. Profit and loss form is as shown figure 4.2.16

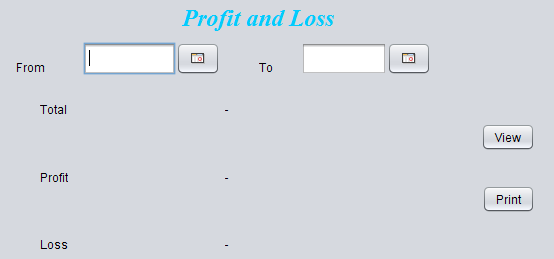


Figure 4.2.16: Screenshot of Profit and Loss

**CHAPTER-5**

**TESTING**

It is the process of testing [9] the functionality and it is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as at undiscovered error. A successful test is one that uncovers an as at undiscovered error. Software testing[8] is usually performed for one of two reasons:

* Defect Detection
* Reliability estimation

**5.1 BLACK BOX TESTING:**

The base of the black box testing [10] strategy lies in the selection of appropriate data as per functionality and testing it against the functional specifications in order to check for normal and abnormal behavior of the system. Now a days, it is becoming to route the testing work to a third party as the developer of the system knows too much of the internal logic and coding of the system, which makes it unfit to test application by the developer.

The following are different types of techniques [involved in black box testing. They are:

* Decision Table Testing
* All pairs testing
* State transition tables testing
* Equivalence Partitioning

**5.1.1 Decision table testing:**

In this the variables are tested for both true and false cases. In this all this type of such cases can be tested.

The Ramamurthy cement spun pipes and brick factory details can be visible only to the authorized user. For unauthorized users the details are hidden. By applying this test the similar type of test cases are identified and solved the issues.

**5.1.2 All pairs testing:**

In this all the input parameters which are to be filled are checked for all possible combinations.

During different types of registration all the fields that are available in the forms need to be filled by the users. If any one of the field is left unfilled an error message gets displayed. Throughout the code development phase, validation was not written for forms which led the user to leave some of the fields unfilled. By this test, test case was identified and solved.

**5.1.3 State transition tables:**

By using this current state of the application and to which states the application will move based on input parameters.

After successful login the user will be redirected to their respective pages in case of failure will be redirected to home page. In the similar manner the different test cases are generated and verified.

**5.2 WHITE BOX TESTING:**

White box testing [10] requires access to source code. Though white box testing [10] can be performed any time in the life cycle after the code is developed, it is a good practice to perform white box testing [10] during unit testing phase.

In designing of database the flow of specific inputs through the code, expected output and the functionality of conditional loops are tested.

**5.3 PERFORMANCE TESTING:**

The performance of the application can be tested through this test. Since the no of lines of code is less, performance is good.

**5.4 DATABASE TESTING:**

This test involves testing whether the values entered through the form gets stored in the database correctly or not. In this, test is made on the database to check values are saved or not.

**5.5 ACCEPTANCE TESTING:**

Testing to verify a product meets customer specified requirements.

The acceptance test suite is run against supplied input data. Then the results obtained are compared with the expected results of the client. A correct match was obtained.

**Test Case1: Admin Login (successful)**

**Test Case:**  Admin logged (successful).

**Test Description:** Provide admin right by checking the details.

**Pre Condition:** Database connectivity.

**Action Performed:** Entered valid login details without leaving any field.

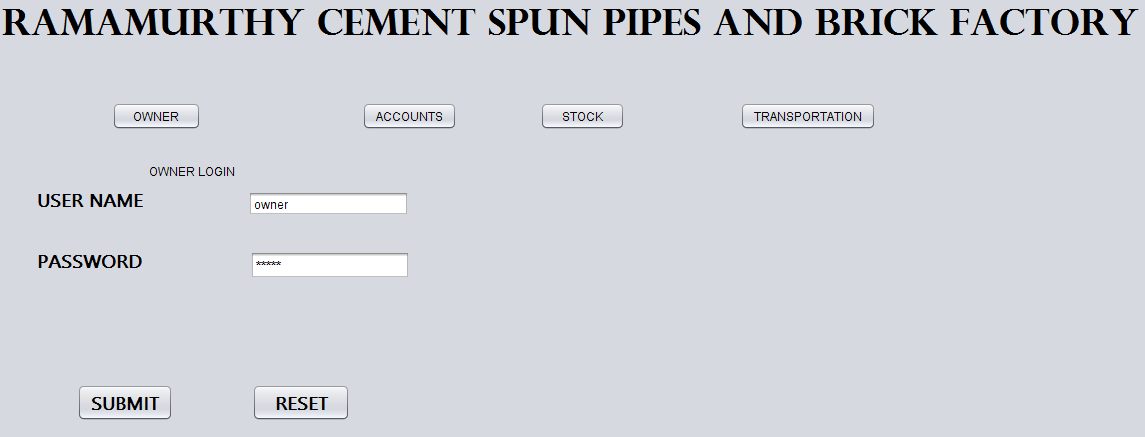


Figure 5.1: Screenshot of Owner Login Page

**Expected Results:** Successful login.

**Condition Verified:** Yes.

**Result:** Success

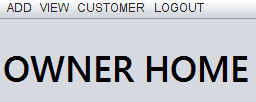
****

Figure 5.2: Screenshot of Owner Home Page

**Test Case2: Admin Login (Unsuccessful)**

**Test Case:**  Admin logged (Unsuccessful).

**Test Description:**  report error message.

**Pre Condition:** Database connectivity.

**Action Performed:** Entered invalid login details.

**Expected Result:** Display an error message alert box regarding the error.

**Result:** Success.

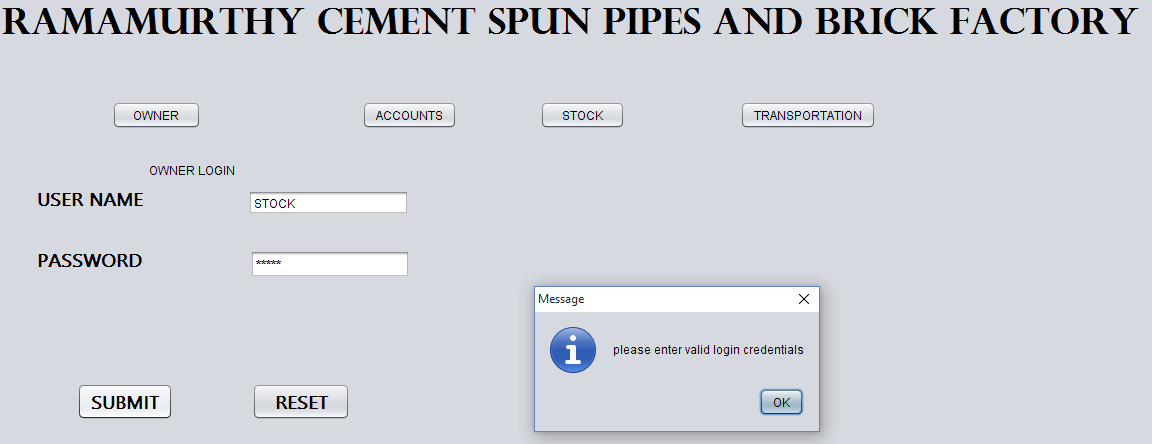


Figure 5.3: Screenshot of Owner Login Page

**CHAPTER-6**

**RESULTS**

After performing testing we get certain results. The results obtained from testing as shown below. Results of the system can be expressed and evaluated in terms of output screen. The output screen can be used to show the objectives set at the beginning are achieved at the end.

**6.1 View Employee Details:**

The following figure6.1 show details about employee.

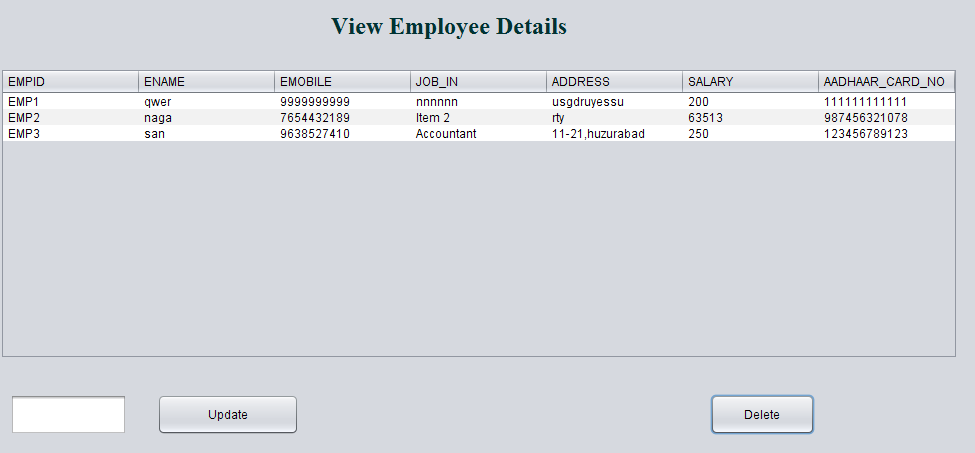


Figure 6.1: Screenshot of View Employee Details.

**6.2 View Attendance and Salary:**

The following figure6.2 show details about employee attendance and salary.

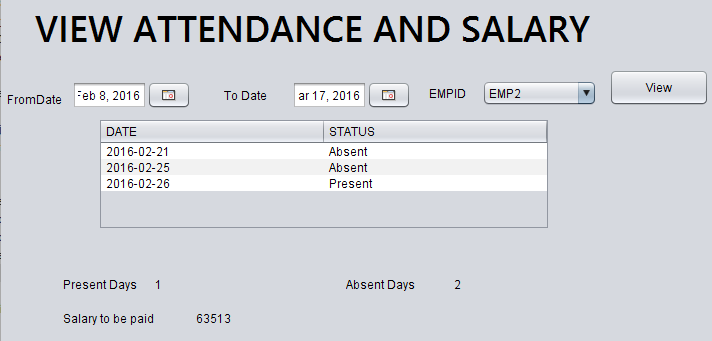


Figure 6.2: Screenshot ofView Attendance and Salary

**6.3 View Employee Expenditure:**

The following figure6.3 show details about employee expenditure .

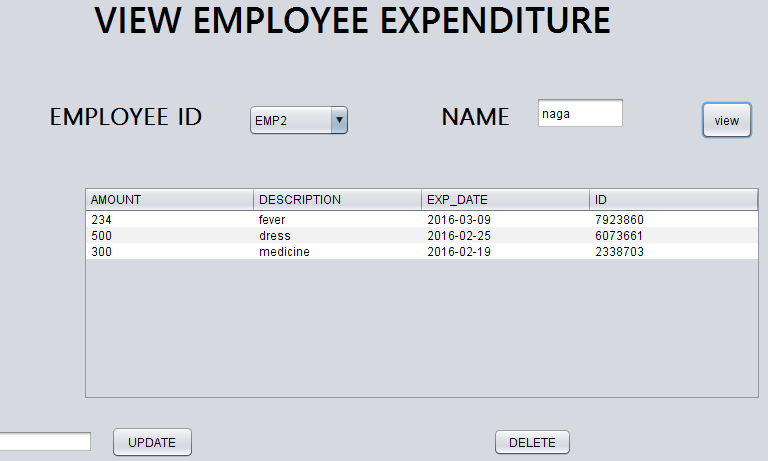


Figure 6.3: Screenshot ofView Employee Expenditure

**6.4 View Expenditure Name:**

The following figure6.4 show details about expenditure.

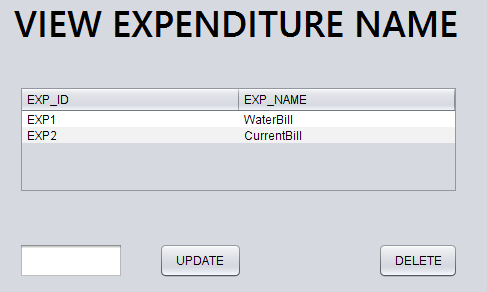


Figure 6.4: Screenshot ofView Expenditure Name

**6.5 View Other Expenditure Details:**

The following figure6.5 show details about other Expenditure .

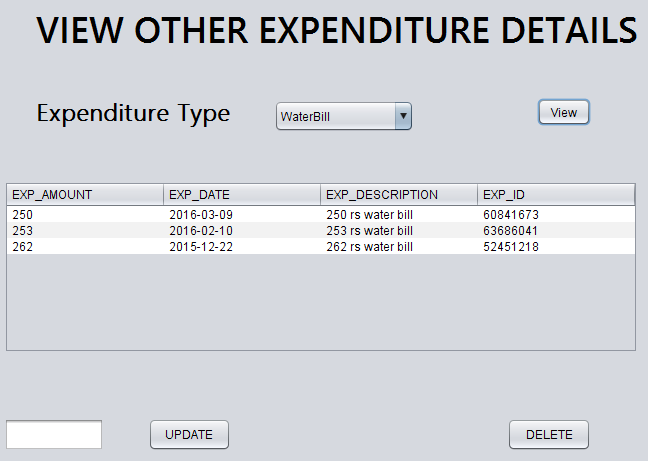


Figure 6.5: Screenshot ofView Other Expenditure Details

**6.6 View Product Details:**

The following figure6.6 show details about product.

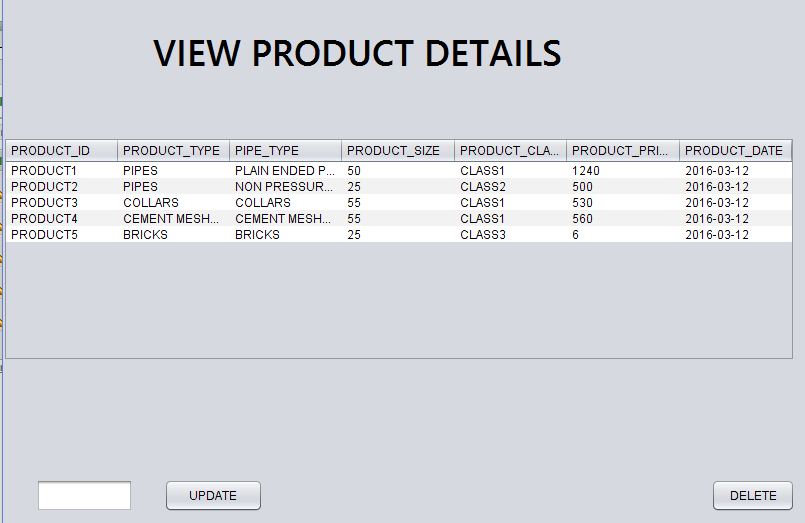


Figure 6.6: Screenshot ofView Product Details

**6.7 View Daily Production Details:**

The following figure6.7 show details about Daily Production .

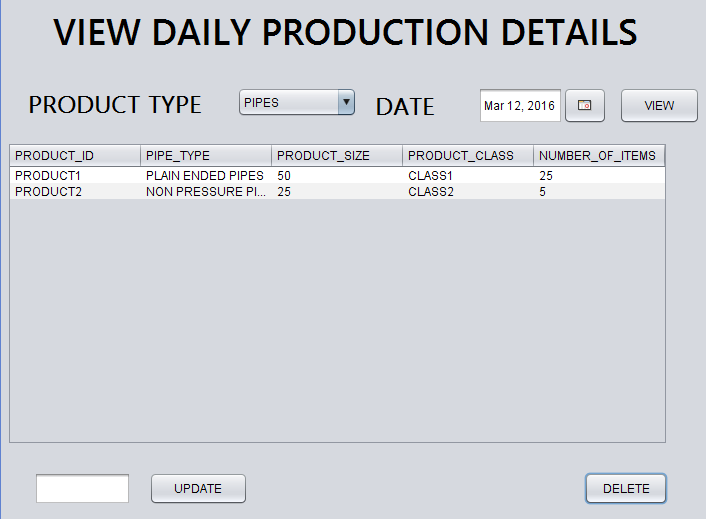


Figure 6.7: Screenshot of View Daily Production Details

**6.8 View Available Products:**

The following figure6.8 show details about Available Products .

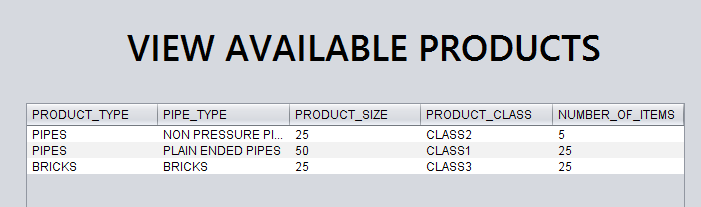


Figure 6.8: Screenshot ofView Available Products

**6.9 View Customer Details on Date:**

The following figure6.9 show details about customer details according to date.

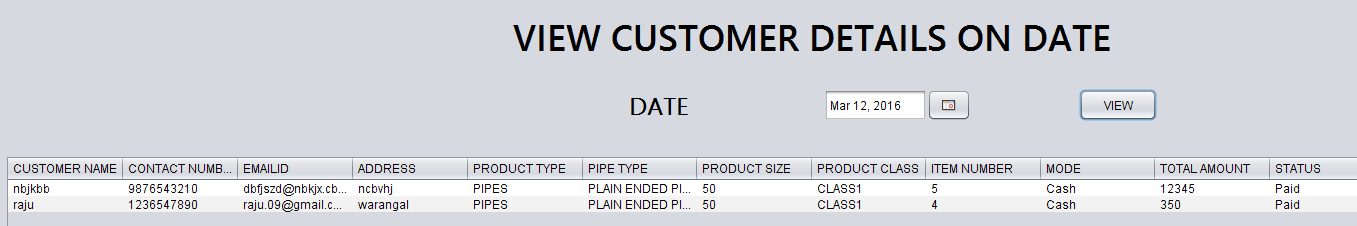


Figure 6.9: Screenshot ofView Customer Details on Date

**6.10 View customer Details:**

The following figure6.10 show details about customer details.

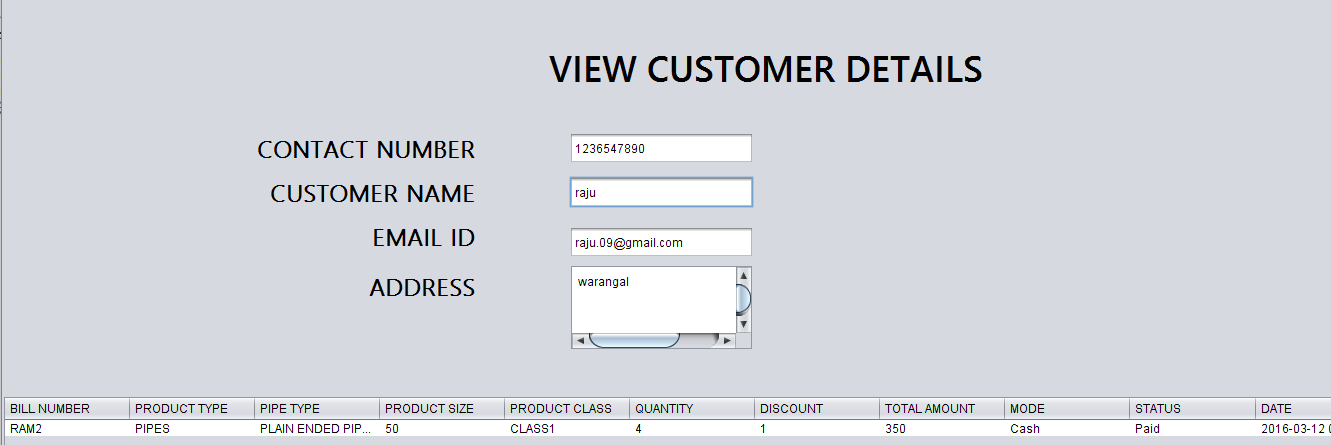


Figure 6.10: Screenshot ofView customer Details

**6.11 View Stock Name:**

The following figure6.11 show details about Stock Name.

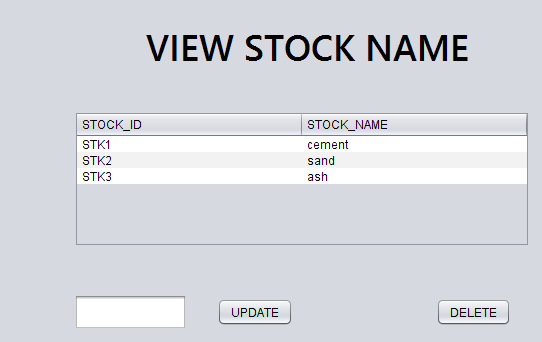


Figure 6.11: Screenshot ofView Stock Name

**6.12 View Transportation Expenditure Name:**

The following figure6.12 show details about Transportation Expenditure Type.

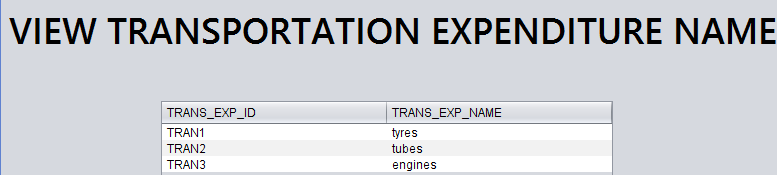


Figure 6.12: Screenshot ofView Transportation Expenditure Name

**6.13 View Transportation Expenditure Details:**

The following figure6.13 show details about Transportation Expenditure Details .

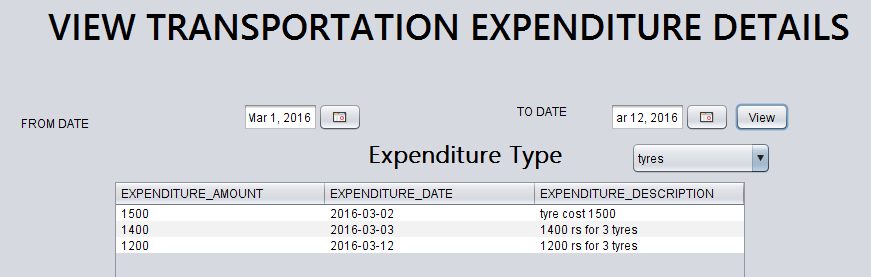


Figure 6.13: Screenshot ofView Transportation Expenditure Details

**6.14 View Transportation Details:**

The following figure6.14 show details about Transportation Details.

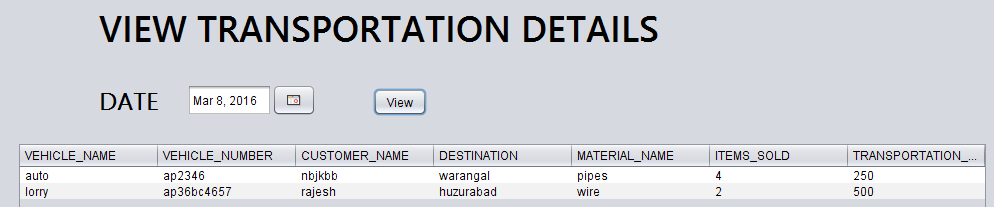


Figure 6.14: Screenshot ofView Transportation Details

**CHAPTER-7**

**CONCLUSION**

* Using JAVA technologywe have developed a database for Ramamurthy cement spun pipes and brick factory, which contains details about customer, products, employee, stock and transportation.
* The application works well and satisfies the needs.
* The application is tested very well and errors are properly debugged.
* It also acts as the management system for valuable resources.
* The usability of the application increases due to the features that were added.

**7.1 LIMITATION:**

According to our client's requirements we have fulfilled their needs. So there are no limitations in our project.

**CHAPTER-8**

**FUTURE SCOPE AND ENHANCEMENTS**

**Future Scope and Enhancements:**

As the current software has to be worked only on the system it is installed in.

* In future we can make this software be available on internet, where as the owner can login and perform action from anywhere through a PDA.
* We can provide login for customer and online transactions.

**BIBLIOGRAPHY:**

[1].The java complete reference to java, Herbert Schildt 7th edition -McGraw-Hill,

ISBN: 978-0-07-163177-8, 2007

[2].Modern Control systems, Dorf &Bishop –Princeton Hall, ISBN: 0133-70-285, 2008

[3].A Practitioners Guide to Software, Roger S.Pressman8th Edition –McGraw-Hill, ISBN: 978-0-07-8022128, 2014

[4].Database Management System, Raghu Ramakrishna/Johannes Gehrke, 2nd Edition, ISBN: 0-67116898-2,2, 2000

[5].Database System Concepts Silberschatz Korth, Sudarshan, 6th Edition - McGraw-Hill ISBN: 0-07-352332-1, 2010

[6].The Unified Modeling Language User Guide, Low Price Edition Grady Booch, James Rumbaugh, Ivar Jacob**,** ISBN: 81-7808-769-5, 91, 1997

[7].The Elements of UMLTM 2.0, Scott Ambler-Cambridge University Press Newyork@2005, ISBN: 978-0-07-52616-7-82, 2005

[8].Software Testing, Cem Kaner, James beach, Bret Pettiehord, ISBN: 978-0-471-120-940, 2001

[9].A Practitioner’s Approach Roger S.Pressman, Software Engineering, 3rd Edition, ISBN: 978-007-126782-3,482, 2014

[10]. Black\_Box Testing: Techniques for Functional Testing of Software and Systems

Boris Beizer - Wiley Publications, ISBN: 978-0-471-120-940, 1995