



**BCSP-064**

**TITLE OF THE PROJECT**

**VIRTUAL KIRANA STORE**

**BY**

**SAGAR ARORA**

**196323647**

**Under Guidance of: Upasana Jain**

**Submitted to the School of Computer and Information Science, IGNOU**  
in partial fulfilment of the requirements for the award of the degree

**Bachelor in Computer Application (BCA)**

**Year of Submission**

**2021-2022**

**Indira Gandhi National Open University**

Gandhi Smriti & Darshan Samiti Rajghat  
New Delhi - 110002 INDIA

# TABLE OF CONTENT

Title of the Project -----	4
Introduction -----	5
Objective -----	6
Tools and Languages Used -----	7
System Analysis -----	12
Identification of Need -----	15
Preliminary Investigation -----	16
Requirement Analysis -----	20
Feasibility Study -----	21
Project Planning -----	26
Project Scheduling -----	27
Software Engineering Paradigm Applied -----	28
Software Requirement Specification -----	34
Data Flow Diagrams-----	39
Entity-Relation Diagram -----	48
Class Diagrams -----	50
System Design -----	54
Module Description -----	57
Database Design -----	59
Coding Efficiency -----	66

Optimization of Code-----	67
Coding-----	69
Output Screens -----	147
Testing -----	171
Implementation -----	182
System Security -----	191
Gantt Chart -----	196
Pert Chart -----	198
Limitations -----	200
Future Scope -----	201
Bibliography -----	202

# TITLE OF THE PROJECT



*Virtual Kirana Store*

## INTRODUCTION

Virtual Kirana Store is a web-based software. This application is use to buy Grocery and Essential products in this pandemic time so that we can maintain the social distancing and can be live safer.

To access this software, the user must have to register him/her self to the website and get a unique user id to access the account and the features.

Registered Customer under grocery stores can check they product delivery, order status and the mode of payment that they have selected.

This application is used by store Admin cum seller who register themselves to get a profile and then they can access the site, add groceries and manage the stock and also can check the profit and earning status from the product and their date of expire and more details.

The operators of the store deals with query and complaints of the customer for the help and maintain the standard of customer obsession and satisfaction.

The Store admin can also register customers and provide them their unique credentials through which they can login to their store. Customer uses this credential to login into the respective store, choose Products and Place order. Payment is done by Cash on delivery.

## OBJECTIVE

Virtual Kirana Store makes your grocery shopping even simpler. No more hassles of sweating it out in crowded markets-now shop from the comfort of your home, office or on the move.

We offer you convenience of shopping everything that you need for your home – be it fresh fruits and vegetables, rice, dals, oils, packaged food, dairy item, frozen, households cleaning items & personal care products from single virtual store.

- This virtual store helps customer to get grocery at home.
- This will provide easy management for operators as well as customers.
- This virtual store helps to keep the record of their regular customers.
- This Virtual store will save your time and money to get the product shipped faster and easier with best price.
- The Software UI is comfortable for the age group 8-60.

# **TOOLS/PLATFORMS, HARDWARE, SOFTWARE REQUIREMENT SPECIFICATIONS**

## **PYTHON**

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but rarely used. It has fewer syntactic exceptions and special cases than C or Pascal

## VISUAL STUDIO CODE

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

Visual Studio Code features a lightning-fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization and community-contributed keyboard shortcut mappings let you navigate your code with ease.

For serious coding, you'll often benefit from tools with more code understanding than just blocks of text. Visual Studio Code includes built-in support for IntelliSense code completion, rich semantic code understanding and navigation, and code refactoring.

And when the coding gets tough, the tough gets debugging. Debugging is often the one feature that developers miss most in a leaner coding experience, so we made it happen. Visual Studio Code includes an interactive debugger, so you can step through source code, inspect variables, view call stacks, and execute commands in the console.

VS Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster. VS Code has support for Git so you can work with source control without leaving the editor including viewing pending changes and diffs.



## IDLE PYTHON

IDLE is Python's Integrated Development and Learning Environment.

IDLE has the following features:

- coded in 100% pure Python, using the Tkinter GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and macOS
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

## XAMPP:

- **Apache:**
  - (Application Server) Apache, often referred to as Server, is an open-source Java Servlet Container developed by the Apache Software Foundation.
- **MySQL Server:**
  - It handles large databases much faster than existing solutions.
  - It consists of multi-threaded SQL server that supports different back ends, several different client programs and libraries,

administrative tools, and application programming interfaces (APIs)

- MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.
- Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.
- MySQL is a registered trademark of MySQL AB.

- ❖ **Sublime Text** - Sublime Text is a sophisticated text editor for code, markup and prose. You'll love the slick user interface, extraordinary features and amazing performance.
- ❖ **Web browsers:** Google Chrome, Mozilla Firefox, Opera and Internet Explorer.

## **SOFTWARE REQUIREMENTS**

Following software are required for developing Web based application:

<b>I.</b>	<b>Operating System</b>	Windows 10.
<b>II.</b>	<b>Environment</b>	Python VS Code/Idle
<b>III.</b>	<b>Front-end Tool</b>	Django, Html, Css
<b>IV.</b>	<b>Technology</b>	Python, Django
<b>V.</b>	<b>Backend Tool</b>	MYSQL5.0

## **HARDWARE INTERFACES**

It's a web – based project, so a robust hardware configuration is required. The hardware requirements are:

<b>I.</b>	<b>Processor</b>	I3 CPU 2.40ghz and above.
<b>II.</b>	<b>Motherboard</b>	Intel P4 or 2.5 ghz
<b>III.</b>	<b>RAM</b>	1GB or above
<b>IV.</b>	<b>Hard Disk</b>	80 GB or Above
<b>V.</b>	<b>Network Card</b>	Standard Ethernet card for networking.
<b>VI.</b>	<b>I/O Devices</b>	Keyboard, mouse and Color monitor
<b>VII.</b>	<b>Wires</b>	Twisted pair for networking.

## SYSTEM ANALYSIS

For solving any problem, I develop a system i.e. a computer system. Therefore, in order to enhance the relationship between the customer and the company I had to develop a reliable system i.e. **(Virtual Kirana Store)**. Therefore, in order to develop any system I need to analyze the requirements of the users. One can think of the systems approach as an organized way of dealing with a problem. In this application, the System Analysis mainly deals with the software development requirements.

System Analysis by definition is a process of systematic investigation for the purpose of gathering data, interpreting the facts, diagnosing the problem and using this information either to build a completely new system or to recommend the improvements to the existing system.

A satisfactory system analysis involves the process of examining a business situation with the intent of improving it through better methods and procedures. In its core sense, the analysis phase defines the requirements of the system and the problems which user is trying to solve irrespective of how the requirements would be accomplished.

The main points to be discussed in system analysis are:

- ❖ Specification of what the new system is to accomplish based on the user requirements.
- ❖ Faults in the previous system

- ❖ Functional hierarchy showing the functions to be performed by the new system and their relationship with each other.
- ❖ Function network, which are similar to function hierarchy but they highlight, those functions which are common to more than one procedure.

List of attributes of the entities - these are the data items which need to be held about each entity.

**After the detailed study of above points, we came to following Conclusion:**

- ❖ Specifications for the tasks achieved by the **grocery management system**, here we highlighted the main task i.e., to handle the grocery and provide feasibility and hassel free life for user.
- ❖ We designed the relationship between various tasks to be achieved by the system and the hierarchy of all tasks in order of their priority.
- ❖ List of various attributes of various entities

In my project firstly I need to analyze the working of organization and evaluate the most feasible requirements of the organization. In this phase I would analyze the main issue to be resolved by this application. So in order to perform all such activities I divided such process into various sub parts.

### Problem identification

- Preliminary Investigation
  - Techniques for fact gathering
- Requirement Analysis
- Feasibility Study
- Project Planning
- Project Scheduling

## **IDENTIFICATION OF NEEDS**

Identification of need is very first step of system analysis. The analyst meets in this phase my main focus is on evaluate the central problem that the organization is facing currently and to know why the previous system fails to provide such facilities.

With the Admin and the end user. Here in my case both users and admin are the common people. In this step, I am interested to understand the product's objective and defined the goals required to meet the objective. After identifying overall goals, I moved towards evaluation of supplementary information during preliminary investigation. I also focused on "does the technology exist to build the system?" and "what bounds have been placed on costs and schedule". To perform it we scanned followings:

- The performance of the system
- The information being supplied and its form
- The economy of processing
- The control of the information Processing
- The efficiency of the existing system
- The security of the data & software

## **PRELIMINARY INVESTIGATION**

The basic purpose behind Preliminary Investigation is to first clarify, understand and evaluate the **Project Request**.

As my application “**Virtual Kirana Store**”. The first step in the development of a system life cycle is identification of the needs. In this phase, I tried to investigate the most common and feasible requirements of users. In these phase I will try to get the answers of the following questions:

- Is the application reliable?
- Weather they wanted to bind their data with certain restrictions?
- Is the storage area is persistent?
- What is being done through this application?
- How it is being done?
- What are the problems in previous system?
- How the systems inter-relate the various kinds of users i.e. (administrator and user)?
- What are the facilities provided by this system to various levels of management i.e. higher level, Middle level and lower level?
- What kind if facilities are provided by the organization to the customers?



To know the answers to these questions I need to perform an intensive investigation. For doing this I need to select an appropriate technique so that I can evaluate a proper solution to all such issues.

## **Techniques for Fact Gathering:**

Fact Gathering means learning as much as possible about the present system and organizational workings.

To do fact gathering, the analyst has the following options:

- Interviews personnel
- Prepares questionnaires
- Observes the current system
- Gathers forms and documents currently in use
- Determines the flow of data through the system, and

Clearly defines the system requirements.

**But I had used the interviewing technique for getting all answers to my questions.**

## **INTERVIEWING**

By studying this organization chart, the analyst can confidently schedule interviews with key personnel involved with the system. Of course, there should be preliminary interviews. Later he/she will conduct a detailed interview with all the people who actually operate the system. Not only will these people use the

newly developed system, but they also may be the ones most afraid of change, especially if they feel the computer might replace them. Like an investigative reporter trying to discover the who, what, when, why and how of a story, the analyst should conduct the interview in such a way that people provide honest descriptions of their jobs. The following questions can help accomplish this goal.

- Who is involved with what you do?
- What do you do?
- What are the main functions?
- Where do you do it?
- When do you do it?
- Why do you do it the way you do?
- How do you do it?
- Do you have suggestions of change?

Interviews help gather vital facts about existing problems, such as lack of quality control or sufficient security, but they also allow the analyst to involve people in change, easing them into it. After all, it is the users' system, not the analyst's.

After interviewing a large number of individual I came to following answers to these questions:

- In this software I will present a web application the main theme is to develop this application, to establish a strong relationship with the customer and organization through providing service this application will

gather data /complaints from Users. After getting the data from customer the problem will have solve by service engineer or administrator.

- The project Virtual Kirana Store must be reliable so that user can conclude decision on the basis of the output generated by the system.
- Furthermore users also wanted to bind their data with certain restrictions i.e. only a valid user can access the data through the application.
- The users also wanted that their data must be secure i.e. their data must not be hacked.
- The system must be able to handle multiple requests at same time. If the number of request is far more than the expectations then also system must generate reliable output.
- The application need to be developed such as, the Users who wants to register them to website to place an order, Then the Users have permitted to register them after taking all the information. After getting registered Users would able to cast their orders from adding items in cart and view the order lists and results.
- In the previous system I faced - network failure problem, speed problems etc. but in this system we handled all the problems.

After receiving all the requests I need to conclude that weather all the requests are possible to achieve or not. So I need to perform a proper feasibility study. But before feasibility study we need to perform requirement analysis.

## **REQUIREMENT ANALYSIS**

Requirement analysis is a software engineering task that bridges the gap between system levels requirements engineering and software design.

### **Requirement Specification of the Project from admin point of view:-**

- Updating of stocks, users and payment records
- Creation of users and user profiles
- Marinating of reports:
  - Report generation after shopping
  - Result report
  - Daily stock Report
- Viewing the size of the grocery and user.

### **Requirement Specification of the Project from user point of view:-**

- Can register with the project
- Can order grocery.
- Can view the status of their record.
- Can view the details about grocery.

### **Requirement Specification of the Project from User's point of view:-**

- Viewing his or her Information.
- Order grocery.
- View the grocery list
- Saving his/her information document.
- View above generated reports

## **FEASIBILITY STUDY**

A feasibility study determines whether the proposed solution is feasible based on the priorities of the requirements of the organization. A feasibility study culminates in a feasibility report that recommends a solution. It helps us to evaluate the cost-effectiveness of a proposed system.

The feasibility study is carried out to test if the proposed system is worth being implemented. Given unlimited resources and infinite time, all projects are feasible.

After performing a Preliminary Investigation, gathering and interpreting data and details concerning the project, a Feasibility Check is done which involves a series of steps to check the Technical, Financial and Operational feasibilities.

During this phase, various solutions to the existing problems were examined.

For each of these solutions the Cost and Benefits were the major criteria to be examined before deciding on any of the proposed systems.

### **These solutions would provide coverage of the following:**

- a) Specification of information to be made available by the system.
- b) A clear-cut description of what tasks will be done manually and what needs to be handled by the automated system.
- c) Specifications of new computing equipment needed.

A system that passes the feasibility tests is considered a feasible system. The feasibility study is an investigative phase where a variety of reasonable solutions are presented. Each feasible alternative is described, the advantages and

disadvantages are identified and the cost is estimated. Both long- and short-term advantages and disadvantages should be considered from all viewpoints: operational, technical, scheduling. Developing a number of feasible solutions and gathering enough information on each alternative to provide an indication of cost can be a very difficult exercise. It may well exceed the work required to design and implement the chosen alternative. Systems analysts must, in a relatively short time, come to understand the constraints and operations of the organization and add to this their own knowledge and experience of software design and development. If their solutions and advice is to be relevant, the systems analyst must maintain an up-to-date knowledge of the developments in the IT industry.

In feasibility study I will try to sort out some points that's evaluate whether the System are feasible: -

- Yes, the project Virtual Kirana Store is reliable with user and system also.
- Furthermore, users also wanted to bind their data with certain restrictions i.e. only a valid user can upload or download data through the application.
- The users also wanted that their data must be secure i.e. their data must not be hacked.
- In this software I will present a web application the main theme is to develop this application, to provide a platform where Users can place an order easily without any problems
- The application has been done such as , the Users who wants to register his/her names for voting , must have register itself online , then the Users have permitted to register their names and post it to that firm from he/her bought

such product that's he/she complaining. After getting complaints the operator views or complaints and filtered the complaints according to their locality and severity. The service engineer fills a daily complaints service form for User's complaints, and solve these complaints.

- In the previous system I faced - network failure problem, speed problems etc. but in this system, we handled all the problems.

In the feasibility study a **cost-benefit analysis** of each option is undertaken. In cost-benefit analysis each option must be understood in sufficient detail to assign a monetary value to components. For each option the systems analyst must describe the technical feasibility and determine the impact on, and the cost to, the organization from a number of viewpoints: operational; Technical; schedule.

In my Application **Virtual Kirana Store** have performed following types of feasibility study:

➤ **Technical feasibility:**

It is related to the software and equipment specified in the design for implementing a new system. **Technical feasibility** is a study of function, performance and constraints that may affect the ability to achieve an acceptable system. During technical analysis, the analyst evaluates the technical merits of the system, at the same time collecting additional information about performance, reliability, maintainability and productivity. Technical feasibility is frequently the most difficult areas to assess.

The main technical issue raised during feasibility is the existence of necessary technology and whether the proposed equipment has the capacity to hold

required data. The technical guarantee of accuracy, reliability, ease and data were also investigated.

### **Assessing System Performance:**

It involves ensuring that the system responds to user queries and is efficient, reliable, accurate and easy to use. Since I have the excellent network setup which is supported and excellent configuration of servers with 80 GB hard disk and 1GB RAM, it satisfies the performance requirement.

After conducting the technical analysis I found that my project fulfills all the technical pre-require Software's, the network environments if necessary are also adaptable according to the project.

### **➤ Scheduling Feasibility:**

Here I evaluate the effect of each option on my scheduling constraints. Every organization has a schedule of events it must meet; for example: In this phase, I will try to make estimates about each and every task that I need to accomplish in order to develop the application and I will analyze whether all such task is feasible to accomplish with in the given time limit.

### **➤ Operational Feasibility:**

Operation feasibility is a measure of how people feel about the system. Operational Feasibility criteria measure the urgency of the problem or the acceptability of a solution. Operational Feasibility is dependent upon determining human resources for the project. It refers to projecting whether the system will operate and be used once it is installed.



If the ultimate users are comfortable with the present system and they see no problem with its continuance, then resistance to its operation will be zero.

Behaviorally also the proposed system is feasible. A particular application may be technically and but may fail to produce the forecasted benefits, because the company is not able to get it to work. For the system, it is not necessary that the user must be a computer expert, but any computer operator given a little bit of knowledge and training can easily operate.

My Project is operationally feasible since there is no need for special training of staff member and whatever little instructions on this system is required can be done so quite easily and quickly. This project is being developed keeping in mind the general people who one have very little knowledge of computer operation, but can easily access their required database and other related information. The redundancies can be decreased to a large extent, as the system will be fully automated. Here I answer questions like:

- Is the system is easy to use?
- Weather existing premises need renovation?
- Will users accept these changes?
- Weather organization operations will be interrupted while I set up?
- Will any changes in one section of the organization adversely affect another section?

## **PROJECT PLANNING**

### **Phase 1 Requirement Gathering and Analysis**

Gather system requirements and prepare a System Requirement Specification document. After collect the information, it is analyzed that the available resources can fulfill all the requirements. In addition, it be examined that what resource will be used.

### **Phase 2 System Design**

Make a detailed analysis of the system and prepare a System Design Document based on SRS.

### **Phase 3 Prepare UTC & STC for testing the software**

Prepare Unit Test Cases document. This document will be used to verify whether the functional requirements of the system have been met.

### **Phase 4 Develop the software**

Develop the planned system.

### **Phase 5 Test the software using the prepared UTC/STC and Rework if needed**

Run your software programs using the respective UTC to verify & test the software.

### **Phase 6 Demonstrate the software to users & implement it**

## **PROJECT SCHEDULE**

<b>Step #</b>	<b>Time Frame</b>
<b>Phase 1.</b> Requirement Gathering	10 Days
<b>Phase 2.</b> System Design	15 Days
<b>Phase 3.</b> Prepare UTC & STC for Testing the software	10 Days
<b>Phase 4.</b> Develop the software	30 Days
<b>Phase 5.</b> Test the software using UTC & STC	20 Days
<b>Phase 6.</b> Demonstrate the software to users & implement it	4 Day

# **SOFTWARE ENGINEERING PARADIGM APPLIED**

## **Introduction**

Software engineering is a layered technology, which comprises of four independent layers as a quality focus layer, the process layer, the methods layer, and the tools layer. The foundation for the software engineering is the process layer and together with the technology layer helps the rational and timely development of computer software. The method layer provides a technical way to perform a broad array of tasks regarding requirements analysis, design, program construction, testing, and maintenance. The tools layer encompasses the software engineering tools to provide automated or semi-automated support for the process and the methods.

In real world, the software development is a teamwork that incorporates a development strategy that encompasses the process, methods, and tools layers and so the strategy is termed as a process model or Software Engineering Paradigms.

According to Fritz Bauer “Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines”. There are several models suggested to go through the software process. The one most suitable & we opted the linear sequential model to develop the software.

## **System Engineering**

The software is always part of a large system and is built using modular approach. The very scratch work begins by establishing requirements for all system elements and then allocating some subset of these requirements for all system elements and then allocating some subset of these requirements to the software. This system view is essential whenever the software is intended to interact with other system elements such as hardware, people, and database. In due course the system engineering and the analysis encompass requirements gathering at the system level with a small amount of top-level design and analysis. Information engineering encompasses requirements gathering at the strategic business level and at the business area level.

## **Software Requirement Analysis**

In the very first approach of software development, the requirement's gathering process is performed and later intensified and focused on the software. To understand the various characteristics of the software to be built, first we must have to understand the information domain for the software, and the required function, behavior, performance, and interface. Requirements for both the system and the software are documented and must be reviewed with the customer. Keeping this software engineering paradigm, we are having a proper documentation on our software so that Librarian's requirement should be reviewed with the user. It helped us to minimize the distance between the actual requirement and the software required and hence kept and will keep the customer's satisfaction very high.

## **Design**

The design is almost and always a multi-step process and focuses on four distinct attributes of a program:

- Data Structure
- Software Architecture
- Interface Representations and
- Procedural Detail.

The design process translates requirements into a representation of the software that can be assessed for quality before coding begins. Like requirements, the design is documented and becomes part of the software configuration.

## **Code Generation**

The design is translated into a machine-readable form using some programming tools.

## **Testing**

Once the design is converted into machine-readable form the program testing phase starts. The testing process on our software mainly focused on the logical internals of the software, ensuring that all statements have been tested properly, and on the functional externals. Through testing we mainly intended to uncover errors and ensured that the defined input produces the desired results that agree with required results in the specification.

## **Maintenance**

Software will undoubtedly undergo change after it is delivered to the customer. Change will occur due to either errors have been encountered or the software have been adapted to accommodate in its external environment or the customer requires functional or performance enhancements. Software support/maintenance reapplies each of the preceding phases to an existing program rather than a new one. The linear nature of the classic life cycle worked perfect without any “blocking states” as the project is to be developed by me under the guidance of the project coordinator only & hence, we started the next phase as soon as the current phase is finished without waiting for anyone else.

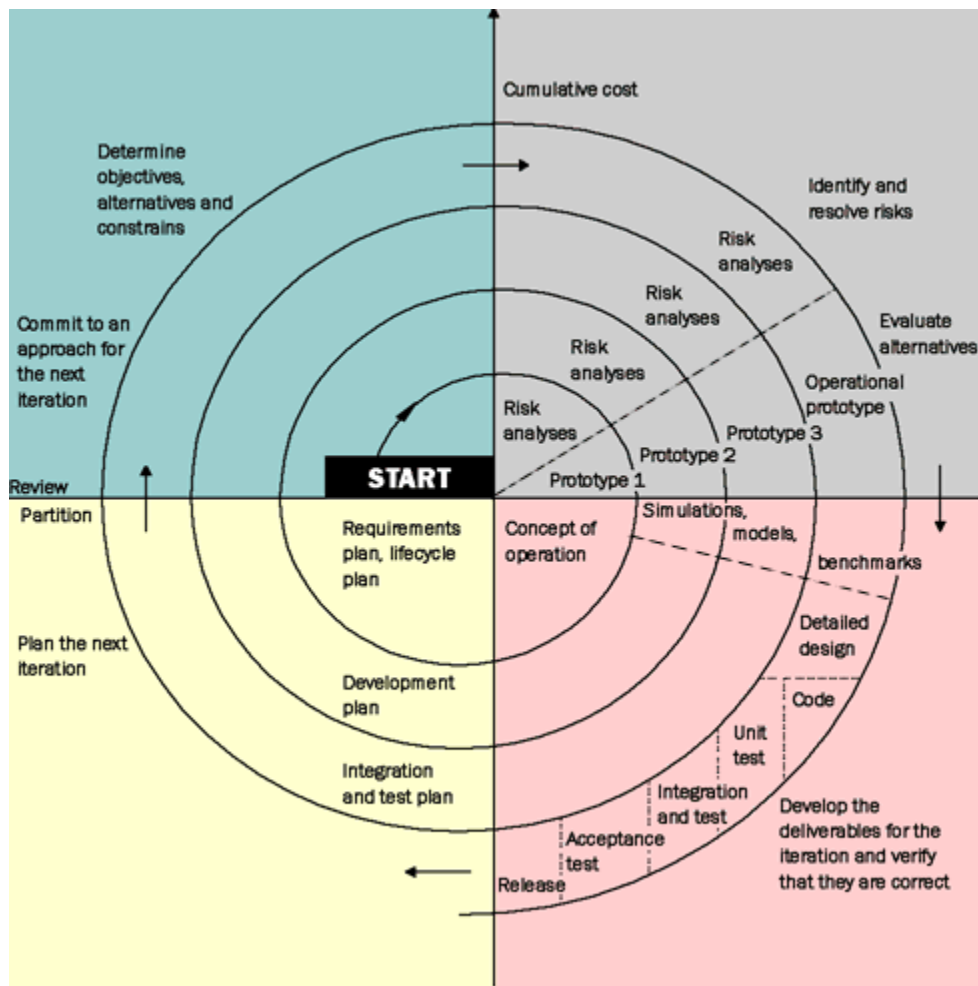
## THE SPIRAL MODEL

This is developed to encompass the best features of both the classical life cycle and prototyping, at the same time adding a new element – risk analysis and hence known as a risk-oriented software life cycle. Each spiral addresses major risks that have been identified. After all the risks have been addressed, the spiral model terminates as a waterfall life cycle.

The **Spiral model** uses an “evolutionary” approach and is currently the more realistic approach, but like other paradigms the spiral approach is not a panacea. This approach requires a considerable risk assessment expertise. If a major risk is unidentified, problems will undoubtedly occur. But this approach has the primary advantage that its range of options accommodates the good features of existing software process models, while its risk driven approach avoids many of their difficulties.

The Spiral Model, the Classic life cycle, or the Waterfall model suggests a systematic and spiral approach to the software development. The very first phase starts from the system level and spirally progresses i.e. whenever the developer wants to change design, coding etc., can easily perform the work. A pictorial view of opted model to develop the software is depicted in figure 1.





## Spiral Model

# **SOFTWARE REQUIREMENT SPECIFICATION**

## **Overview**

The *software requirement specification (SRS)* is very important part of the software building process, which describes the actual user level requirement from technical point of view. i.e. what the user exactly wants? or for what purpose we are making everything The objective of preparing the software requirement specification is to represent the requirements of the software in such a manner that ultimately leads to successful software implementation. It is the result of the analysis process of the software development. It should contain all the data the software is going to process, the function it will provide, and the behavior it will exhibit. This Software Requirements Specifications (SRS) is defined in IEEE Std. 830-1993, IEEE Recommended Practice for Software Requirements Specifications. The document is organized in the following structure:

- Introduction
- Information Description
- Functional Description
- Behavior Description
- Validation Criteria
- Bibliography
- Appendix

**Introduction**

The introduction section describes the goals and objective the software under going development in context of computer-based system. It mainly deals with the software scope. i.e. it will bind the software application domain.

**Information Description**

This section of the SRS provides a detailed of the problem that the software must solve. It should describe the core of the software – i.e. *The Data*. The data or information the software is going to work on is the most basic part of the software. The description of each data or information entity is described here. It also gives details of the relationships between the data elements of the software. The information description helps the software designers in their designing purpose.

**Functional Description**

This section of the SRS describes each functions required to solve the problem. It emphasizes on the core of the software on which the data will be processed – i.e. *The Function*. It also includes the process specification of each function, design constraints, and performance characteristics. The DFD or any other graphical diagram can also be added to describe the functionality of the system.

**Behavioral Description**

This section of the SRS describes the behavior of the software will exhibit. It is based on definition of the events and the operations that it will perform because of events.

**Validation Criteria**

This section of the SRS contains the details of the tests that should be performed to validate functions, performance, and behavior of the software. It is one of the most important aspect of the software which decides how much robust our software is.

**Bibliography**

This section contains references to all the related documents related with the software. This may include any technical document, standards document or software engineering paper.

**Appendix**

This section is supplementary and can include the matters that are important for the software development. It may include the statistical data, graphs or algorithm details.

# **SRS Document**

## **(Software Requirements Specification Document)**

### **Introduction**

This document is meant to delineate the features of OSS, so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other. Virtual Kirana Store (VKS) for electronics item shop web application is intended to provide complete solutions for vendors as well as customers through a single get way using the internet. It will enable vendors to setup online shops, customer to browse through the shop and purchase them online without having to visit the shop physically. The administration module will enable a system administrator to approve and reject requests for new shops and maintain various lists of shop category.

### **Information**

By registering on this Website user can cast their Orders. When Administrator starts the website Server, he will be given the right to user to create id and password so that he can be verified to use the software and its benifits.

### **Function**

VIRTUAL KIRANA STORE(VKS) application enables vendors to set up online shops, customers to browse through the shops, and a system administrator to approve and reject requests for new shops and maintain lists of shop categories. Also, the developer is designing an online shopping site to manage the items in the shop and also help customers to purchase them online without visiting the shop

physically. The online shopping system will use the internet as the sole method for selling goods to its consumers.

## **Behavior**

In order to maintain an acceptable speed at maximum number of uploads allowed from a particular customer as any number of users can access to the system at any time. Also, the connections to the servers will be based on the attributes of the user like his location and server will be working 24X7 times.

## **Validation criteria**

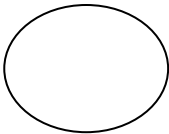
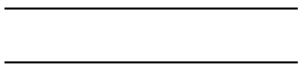

In this we describe the successful implementation, testing, functions, performance and constraints related to our software. The testing of the software will be done on various levels by providing the input constraints to check the validity of the required function. The overall performance of the software will be parameterized by above mentioned validation criteria.

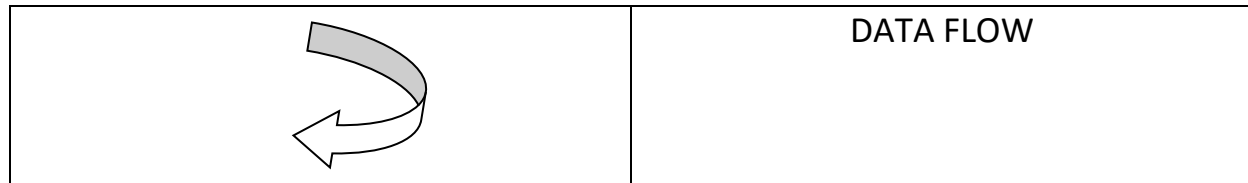
## **Bibliography**

- Python: The Complete Reference: Martin C.Brown
- HTML-The Complete Reference: Mc Graw-Hill Edition
- SQLLITE (TM): The Complete Reference: Vikram Vaswani
- IGNOU Study Materials
- DJANGO :(YOUTUBE) CODE WITH HARRY

## **DATA FLOW DIAGRAM**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kinds of data 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 does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

<b>SYMBOLS</b>	<b>REPRESENTATIVE NAMES</b>
	PROCESS
	DATABASE/FILE
	INPUT/OUTPUT



**Rules of making DFD: - there are following seven rules for the construction of data flow diagram**

- 1) Arrow should not cross each other.
- 2) Square, circles, and files must bear names.
- 3) Decomposed data flow must balance.
- 4) No two data flows, square or circles can have the same name.
- 5) Draw all data flows around the outside of the diagram.
- 6) Choose meaningful names for data flows processes and data stores.
- 7) Control information such as record counts, passwords and validation requirements are not pertinent to a data flow diagram.

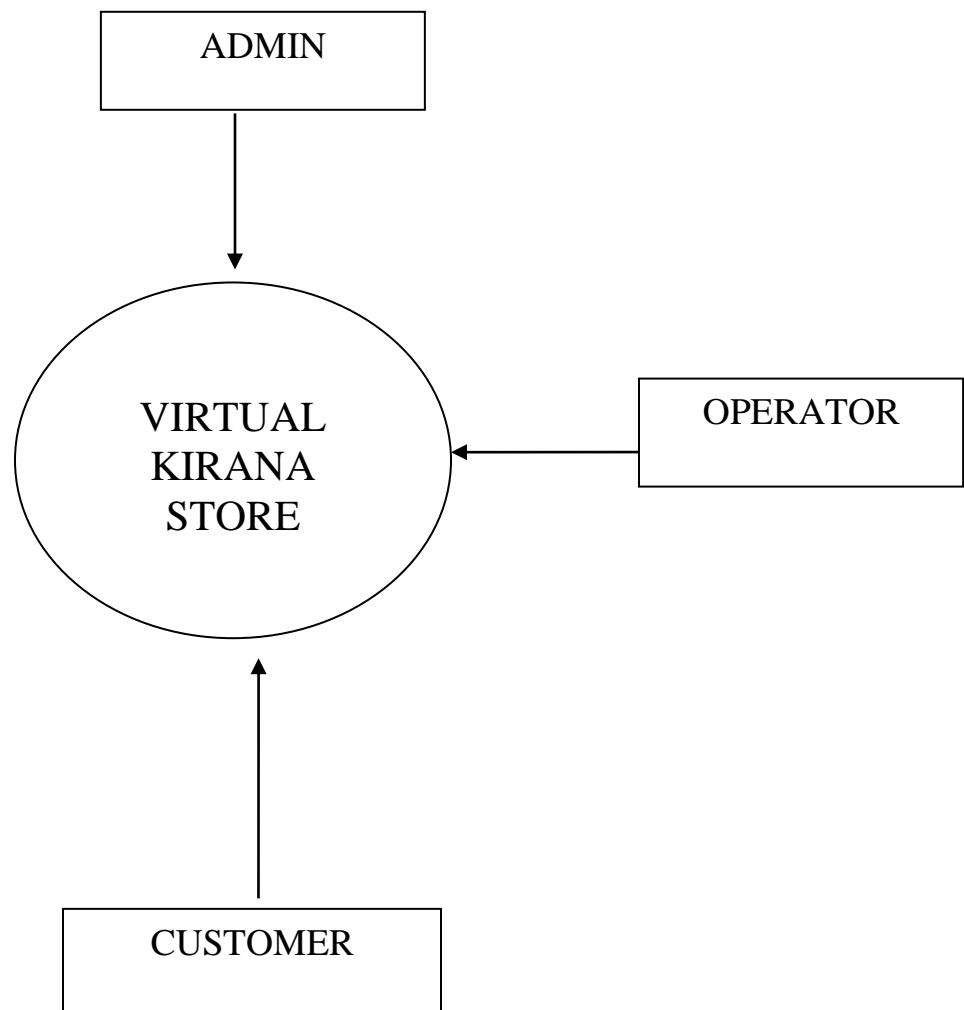
**DFD are describe for different level**

e.g.:

- i. 0 level DFD
- ii. 1 level DFD
- iii. 2 level DFD



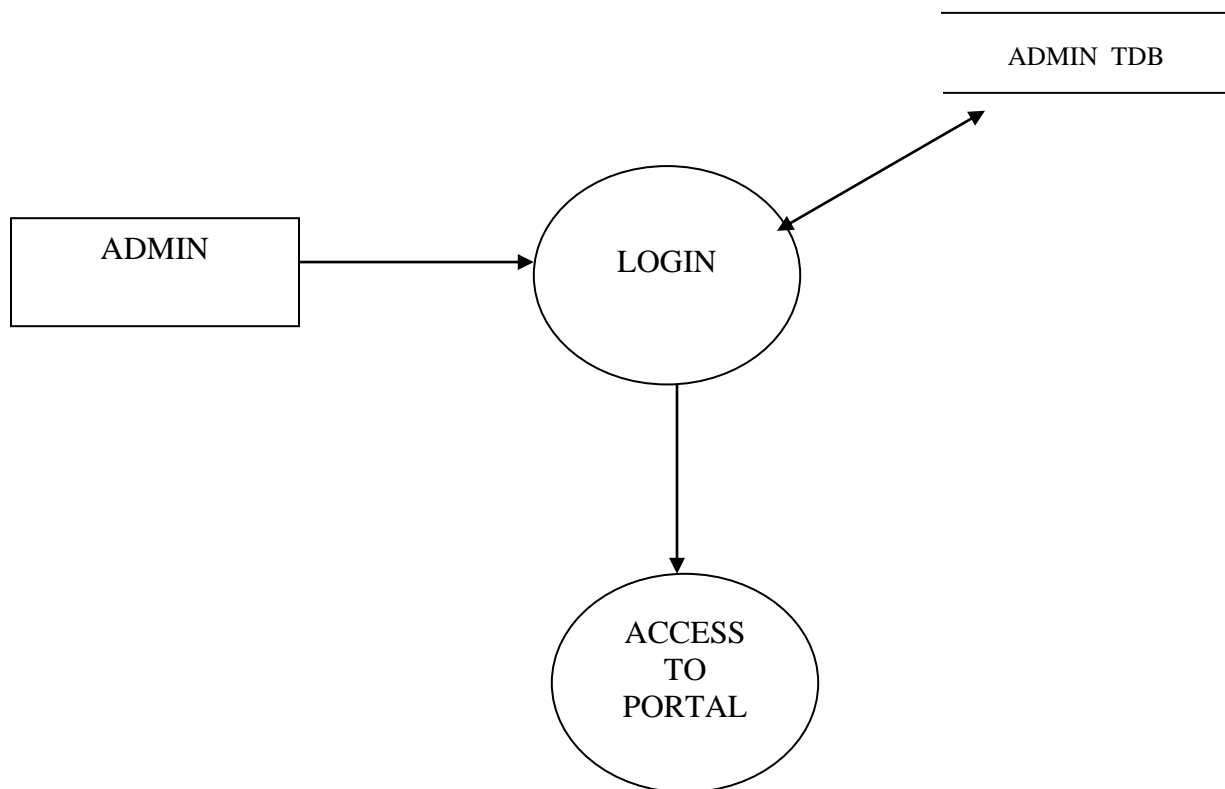
## LEVEL 0 DFD



\

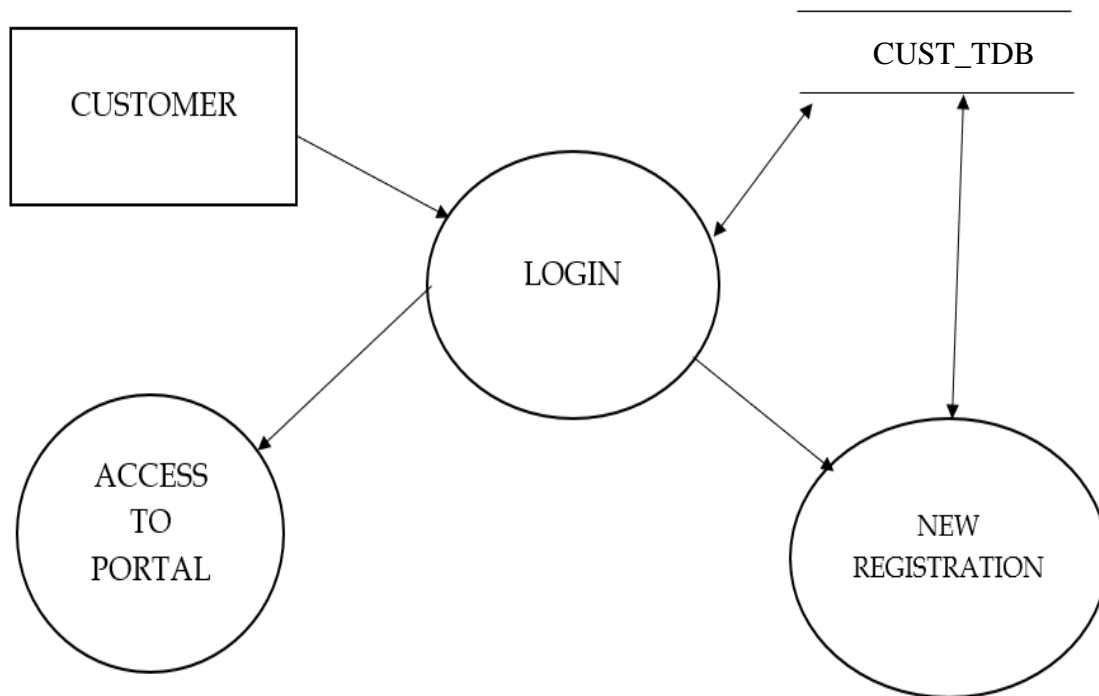
## LEVEL 1 DFD

### ADMIN MODULE

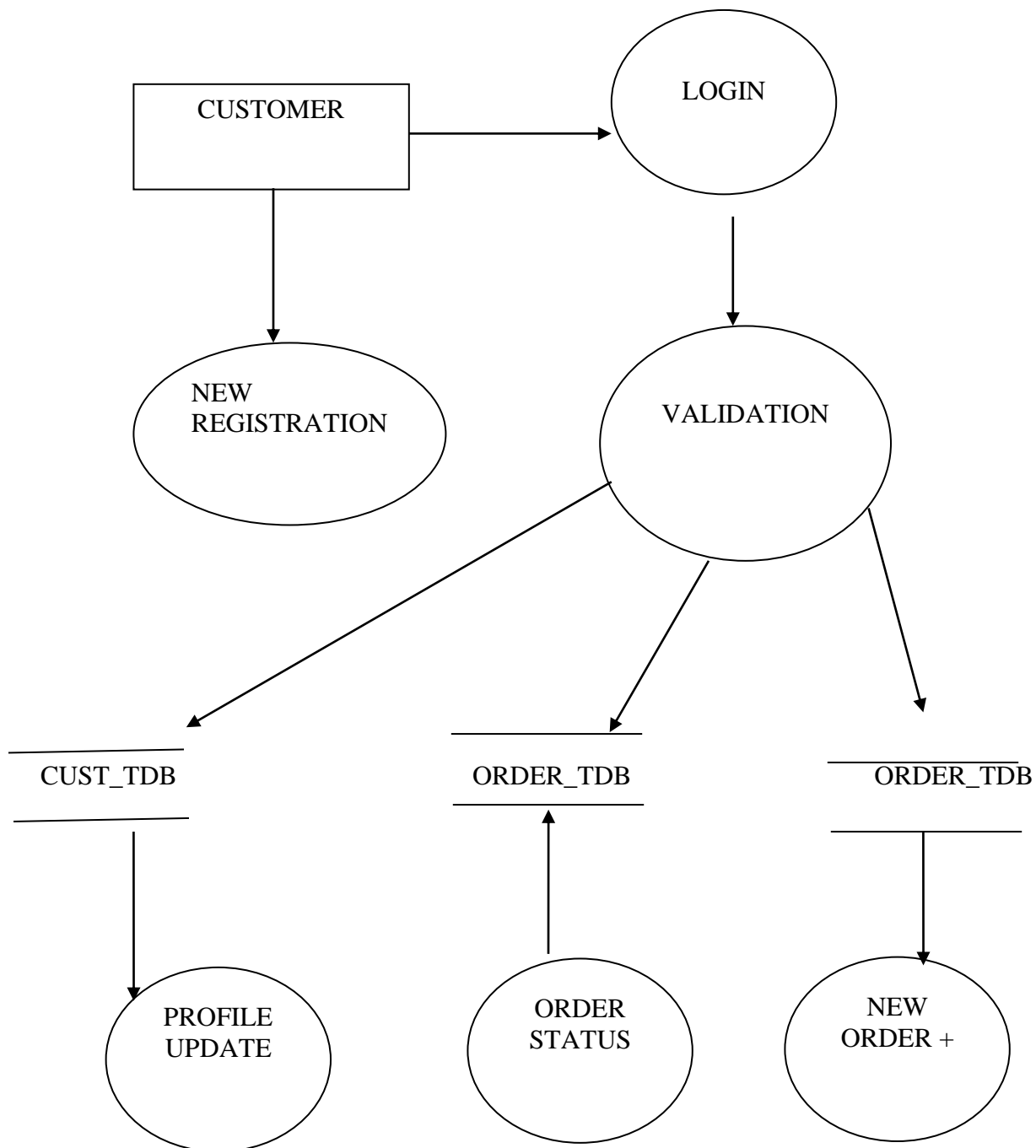


## LEVEL 1 DFD

### CUSTOMER MODULE

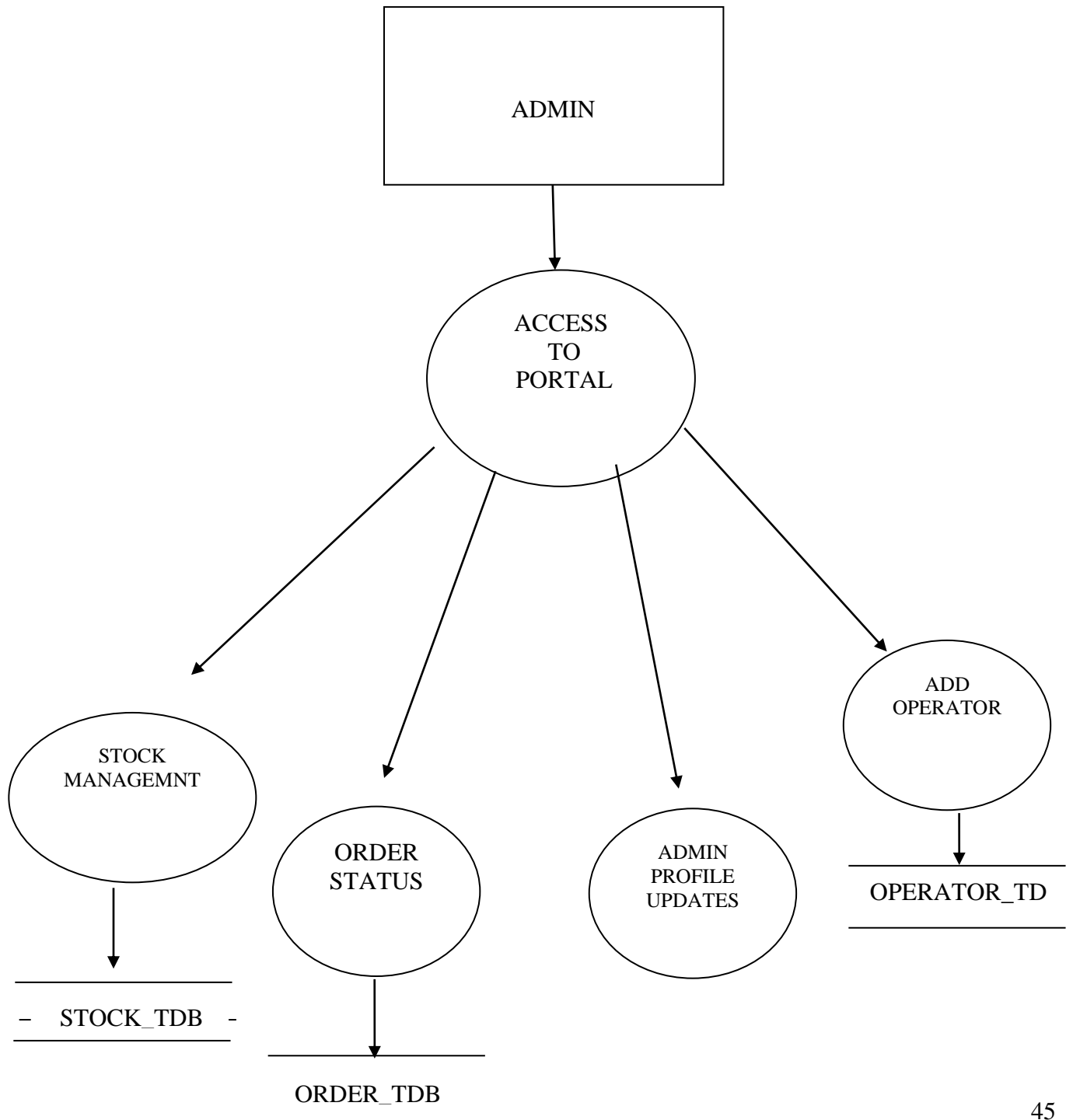


## LEVEL 2 DFD

CUSTOMER MODULE

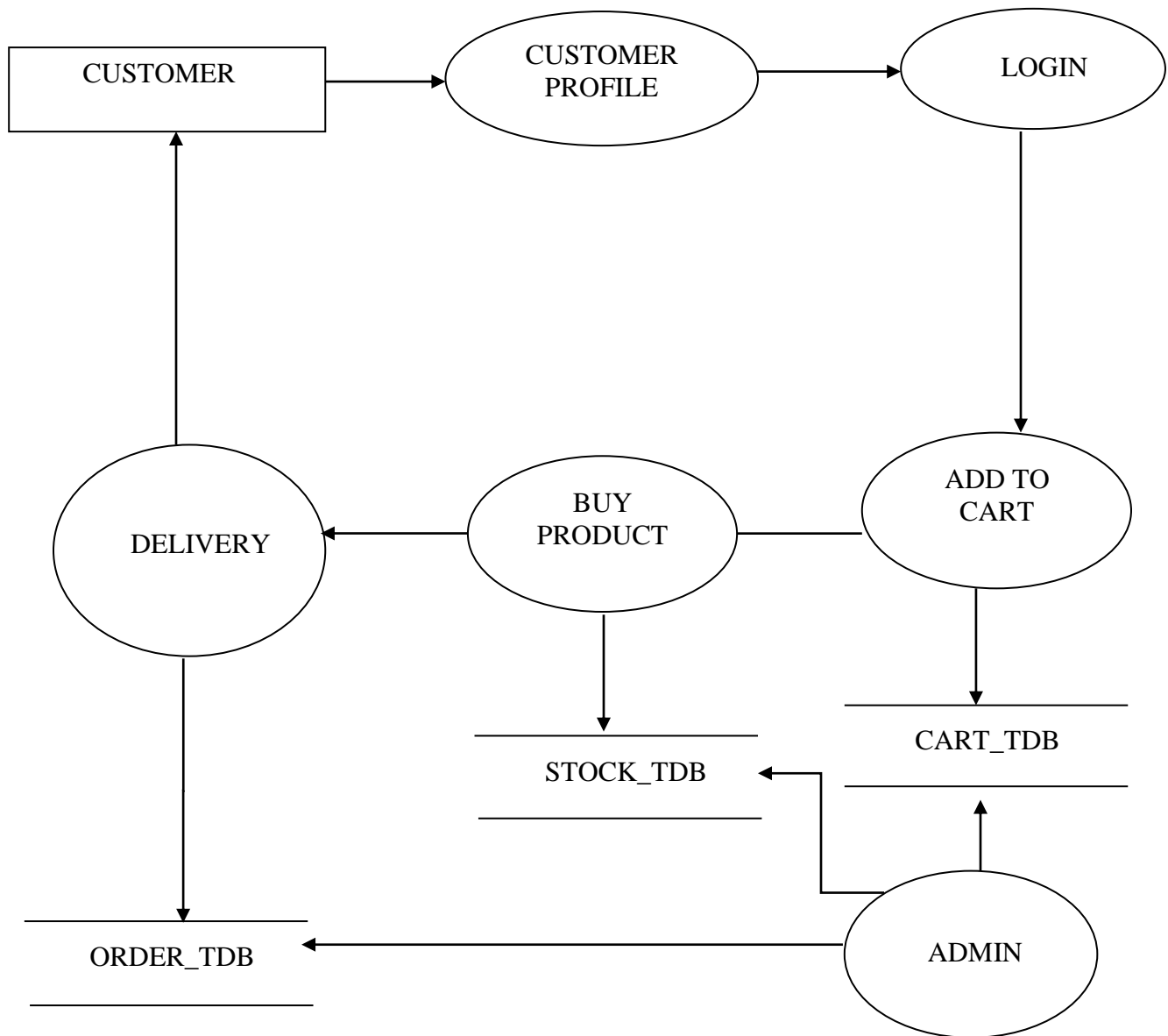
## LEVEL 2 DFD

### ADMIN MODULE



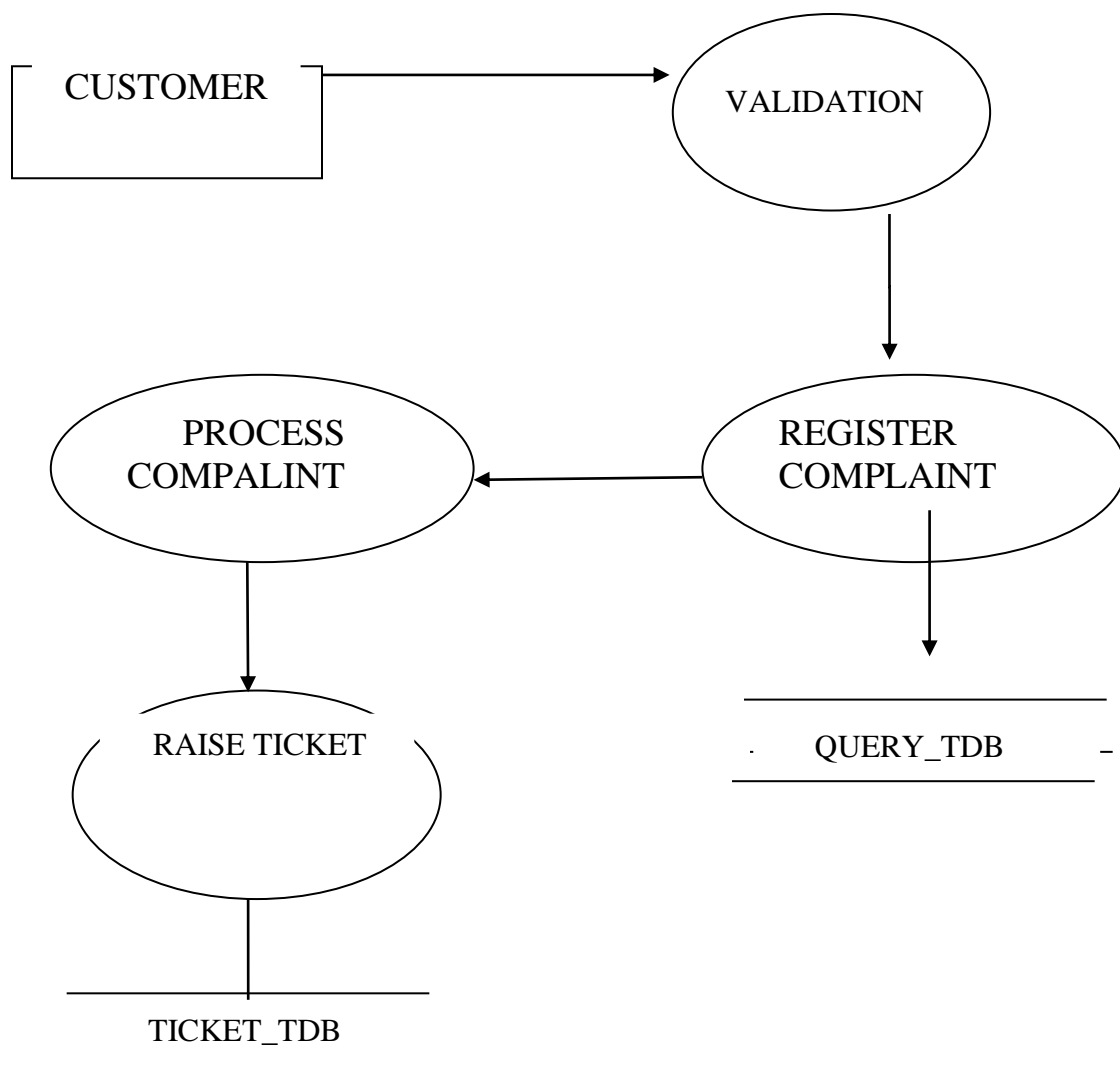
## LEVEL 2 DFD

### NEW ORDER MODULE



## LEVEL 2 DFD

### QUERY COMPLAINT



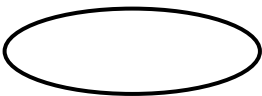
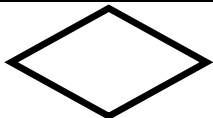


## ER DIAGRAM

**Entity Relationship Diagram (ERD)** is a visual representation of underlying data model in the database. It depicts various entities along with its attributes and relationships with other entities and helps visualize the structure of database objects.

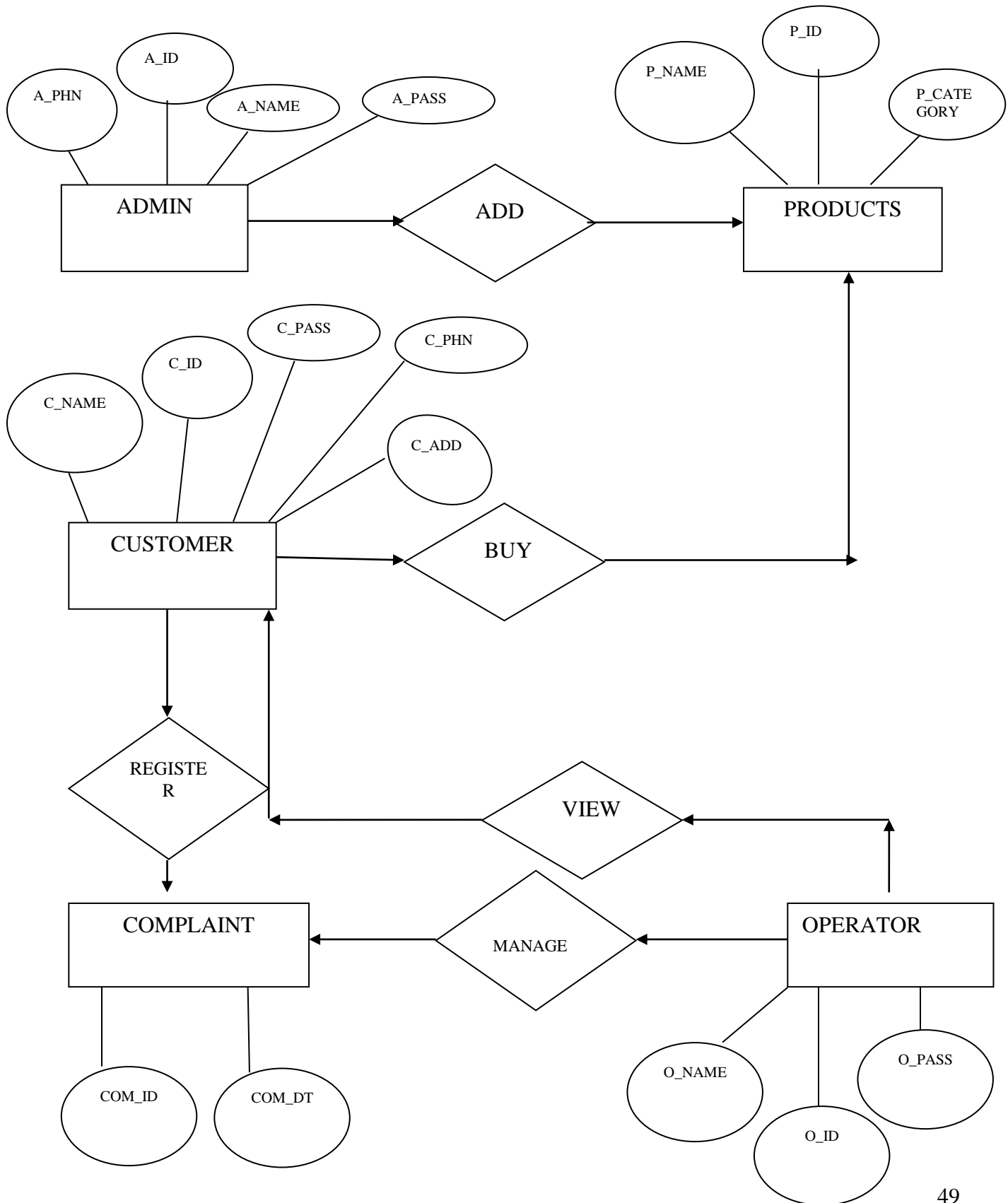
(1) It depicts the complexities of underlying database in visual structure. This helps in structured depiction and communication.

(2) ERD helps in design of data model by identifying different elements and relationships among the elements

Symbols	Representative Names
Entity	
Weak Entity	
Attributes	
Key Attributes	
Multivalued Attributes	
Relationship	



## ER DIAGRAM



## USECASE DIAGRAM

A **use case diagram** is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

It shows the relationship between the user and the system and creates the boundary of the system. **This diagram is used to:**

- Understand the requirements clearly.
- Enable users to understand the system.
- Generate test case to validate the system.
- Build project schedule.

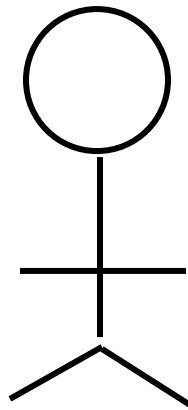
**In Use case diagram, we use seven symbols. They are given below.**

**System:** It shows the boundary of the system and is represented by a rectangle.

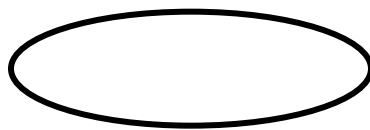


**Actor:** It represents the user or people or device which interacts with the system. It gives input to the system. They are outside of the system

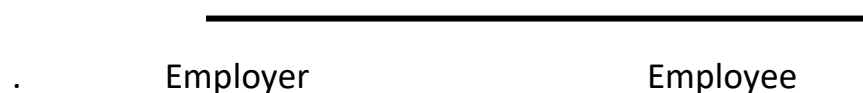
boundary. Actor is represented as given below:



**Use Case:** It shows the desired function of the system. It defines the requirements of the system. Actors gives input to the use case and use case provides output for the given input. It can be represented as below:



**Association:** It is a structural relationship that describes a set of links (a link being a connection among objects). Aggregation is a special kind of association, representing a structural relationship between a whole and its parts. It is represented by a line (may be directed), including labels such as multiplicity and role name as given below:



**Dependency:** This is used to show the relationship between use cases and is represented by a directed dashed line and occasionally includes label



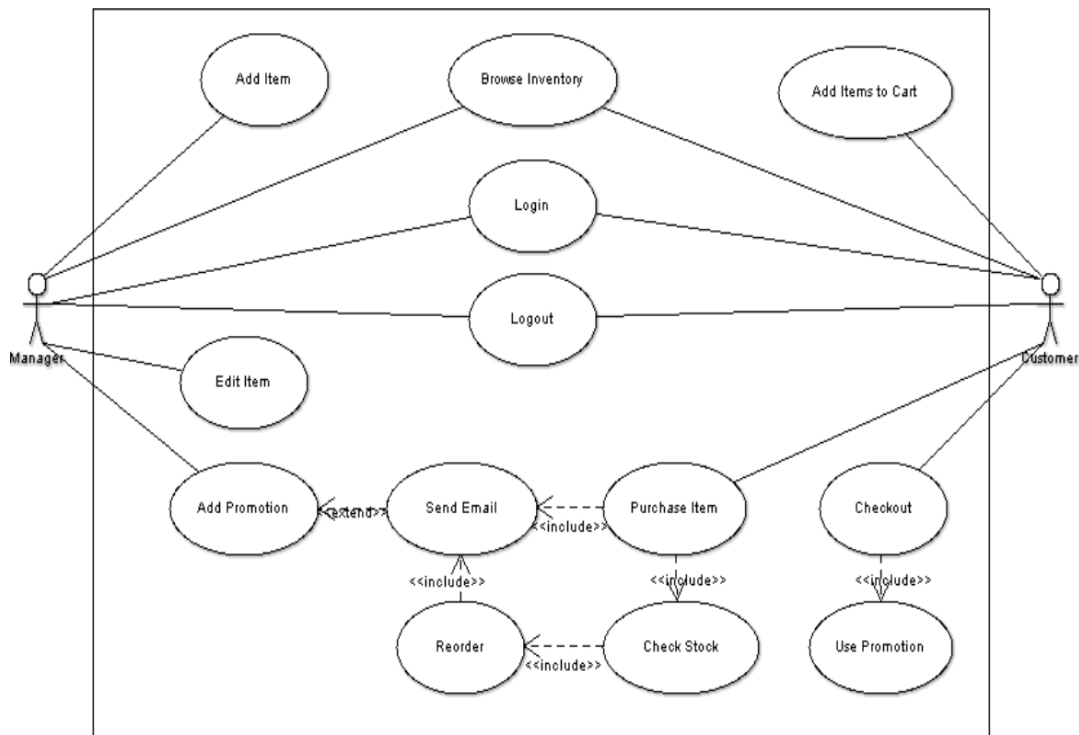
**Generalization:** It shows the concept of inheritance among use cases and actors. A generalization is a relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent). In this way, child shares the structure and behavior of the parent. It is represented by a line with a hollow arrowhead pointing towards the parent.



**Realization:** It is a semantic relationship between classifiers, wherein one Classifier specifies a contract that another classifier guarantees to carry out. It is represented by a dashed line with a hollow arrowhead pointing towards the parent.



# USE CASE DIAGRAM OF VIRTUAL KIRANA STORE



## **SYSTEM DESIGN**

### **SCOPE**

In this section we define the scope of the design effort. The design phase is an important part of the system development phase. A good design of the system needs creativity and flair from the designer and is the key to effective and successful engineering. The following are the basic objectives of the software design process:

- To describe the process of software design where informal ideas are transformed into detailed implementation description.
- Introduction of different stages in the design process.
- Understanding whether an Object Oriented or a Functional Oriented approach or both should be applied to the software.
- Determining and improving, cohesion control and coupling within subsystems.

In analysis of the system, we have seen what a system should do. In system design phase the emphasis will be on how to do what a system should do. There are two main approaches:

→ Data centered approach.

→ Process centered approach

**“Virtual kirana Store”** main function is to provide service to users by monitoring and making analysis of the information about the grocery. **Virtual kirana Store** helps in Grocery management and helps in serving the grocery in a better way by adding more functionality, and making desired and necessary changes in the management. Grocery management system helps to fill the communication gap between the seller and customer. It works as follows:

Processes are managed in following manner in grocery management system:

- First of all the administrator creates various users there are basically 2 types of users which uses the system, i.e. customers, operator.
- Administrator creates various users and assigns them various roles i.e. Manipulators, Handler or a Field Engineer. By providing all the necessary details to the system, then system reviews these details if these details are valid then details are stored in the database.
- A log file is created every time user logs in or log out an entry is made in this log file.
- Administrator can view this log file any time he wants to. If he finds any invalid login then he informs the admin.
- After the user creation, the customers visits the web site, if he has the desire to purchase grocery he would be able to purchase.

- In order to view the information regarding grocery i.e availability, price, the customer must register himself to the application, only then he can view the information of the grocery and other details.
- If a user is registered only then he/she can, view the details of grocery.
- The Operator views the complaints daily. Then filters the complaints according to locality and severity. These Manipulators will then assign a operator to service a set of complaints in a particular locality. The employee can also see the status of complaints assigned to a Field engineer. The view shown to the operator will contain daily complaints, classify complaints, assign complaints and view status.
- Operator view will comprise of view complaints assigned, severity of complaints and fill daily complaint service form. She/he daily fills Service detail form in response to a type of complaint which serves as source/input/feedback information to Branch Office and Head Office. If the engineer has visited the customer and the complaint is closed then he makes a closing entry in the daily complaint service form.
- The admin who want to assess the performance of employees and the complaints related to products will be provided with reports view. The reports will be displayed in graphical and percentage format.



## **MODULE DESCRIPTION**

In the project, the design phase has been identified as one of the most crucial documents. In this phase, we have identified the various aspects of the “VKS”, which have to be implemented as subsystems and their further components. The VKS project has been divided into following modules.

- **CUSTOMER PROFILE.**

It is design for the purpose of creating the Customer's Profile. If Any Customer Wants to buy Grocery, they must first ask stores to register them. Customers Have to Provide their Basic Details Like Name, Phone number and full Address.

- **ADMIN PROFILE.**

Store admin register themselves as admin. They can login and then keep track of Groceries. They track their stocks and add groceries.

- **LOGIN.**

After Registration Customer/store admin will Login to the next module. Here Buyer can Search for their Products from their respective stores.

- **PRODUCT**

Product are Added by the Store admin. Providing details of the product Prices and Company name. Customers Place orders for these products.

- **ORDERS**

Customers login to the Application. They select the grocery and make their Order. Payment is collected in money on cash on delivery.

- **QUERY/COMPLAINT**

This module is designed for Customers who have any Problem or query. Customers can submit their queries which is further solved by Store operator.

- **ADD TO CART**

This module is designed to add the product to the cart before making a final purchase in the software.

## DATA STRUCTURE

### 1) ADMIN\_TDB

Sno	Attribute	Data type	Size	Description
1	A_Id	Int	40	It is the Primary key. It contains the Store.
2	A_name	varchar	255	It contains the name of the Store Admin.
3	A_phn	varchar	Max	It contains the phone of the Store. admin
4	A_pass	Varchar	20	It contains the password of the Store.

**2)CUST\_TDB**

<b>Sno</b>	<b>Attribute</b>	<b>Data type</b>	<b>Size</b>	<b>Description</b>
<b>1</b>	<b>C_Id</b>	<b>Int</b>	<b>40</b>	<b>It is the Primary key. It contains the customers.</b>
<b>2</b>	<b>C_Name</b>	<b>Varchar</b>	<b>50</b>	<b>It contains the name of the User.</b>
<b>3</b>	<b>C_Add</b>	<b>Varchar</b>	<b>max</b>	<b>It contains the address of the User.</b>
<b>4</b>	<b>C_Phn</b>	<b>Varchar</b>	<b>20</b>	<b>It contains the Phone number of the User.</b>
<b>5</b>	<b>C_Email</b>	<b>Varchar</b>	<b>20</b>	<b>It contains the email Id of the User.</b>
<b>6</b>	<b>C_Pass</b>	<b>Varchar</b>	<b>20</b>	<b>It contains Password of the user.</b>

**3)STOCK\_TDB**

<b>Sno</b>	<b>Attribute</b>	<b>Data type</b>	<b>Size</b>	<b>Description</b>
<b>1</b>	<b>P_Id</b>	<b>int</b>	<b>40</b>	<b>It contains the Grocery unique Id. PRIMERY KEY</b>
<b>2</b>	<b>P_Name</b>	<b>varchar</b>	<b>100</b>	<b>It contains the name of the Grocery.</b>
<b>3</b>	<b>P_Price</b>	<b>Decimal</b>	<b>18,2</b>	<b>It contains the Price of Grocery.</b>
<b>4</b>	<b>P_CATEGORY</b>	<b>varchar</b>	<b>45</b>	<b>It contains PRODUCT Category.</b>

**4)ORDER\_TDB**

<b>Sno</b>	<b>Attribute</b>	<b>Data type</b>	<b>Size</b>	<b>Description</b>
<b>1</b>	<b>Order Id</b>	<b>Numeric</b>	<b>18</b>	<b>This contains the Order Id.</b>
<b>2</b>	<b>P_Id</b>	<b>int</b>	<b>40</b>	<b>This contains the PRODUCT ID foreign key.</b>
<b>3</b>	<b>Cust_id</b>	<b>Numeric</b>	<b>18</b>	<b>This contains the customer Id. As a foreign key</b>
<b>4</b>	<b>Qty</b>	<b>Numeric</b>	<b>18</b>	<b>This Contains the Qty.</b>
<b>5</b>	<b>Total_Cost</b>	<b>Numeric</b>	<b>18</b>	<b>This Contains the total cost of order.</b>
<b>6</b>	<b>Order_On</b>	<b>Datetime</b>	<b>40</b>	<b>This Contains the Date On which Order is Placed.</b>

**5)CART\_TDB**

<b>Sno</b>	<b>Attribute</b>	<b>Data type</b>	<b>Size</b>	<b>Description</b>
<b>1</b>	<b>Cart_Id</b>	<b>Int</b>	<b>40</b>	<b>This Contains the InCart Id.</b>
<b>2</b>	<b>P_Id</b>	<b>Int</b>	<b>40</b>	<b>This Contains the Id of product.</b>
<b>3</b>	<b>Cust_Id</b>	<b>Int</b>	<b>40</b>	<b>This Contains the CustomerId.</b>

**6)COMPLAINT\_TDB**

<b>Sno</b>	<b>Attribute</b>	<b>Data type</b>	<b>Size</b>	<b>Description</b>
<b>1</b>	<b>Com_Id</b>	<b>Int</b>	<b>50</b>	<b>This Contains the complaint Id.</b>
<b>2</b>	<b>Cust_Id</b>	<b>Int</b>	<b>50</b>	<b>This Contains the Customer Id.</b>
<b>3</b>	<b>Com_DT</b>	<b>VarcharP</b>	<b>max</b>	<b>This Contains the complaint Made by Customer.</b>
<b>4</b>	<b>O_ID</b>	<b>Int</b>	<b>50</b>	<b>This contains the operator ID.</b>



**7)OPERATOR\_TDB**

<b>Sno</b>	<b>Attribute</b>	<b>Data type</b>	<b>Size</b>	<b>Description</b>
<b>1</b>	<b>O_Id</b>	<b>Int</b>	<b>50</b>	<b>This Contains the OPERATOR Id.</b>
<b>2</b>	<b>O_NAME</b>	<b>varchar</b>	<b>50</b>	<b>This Contains the name</b>
<b>3</b>	<b>O_PASS</b>	<b>Varchar</b>	<b>Max</b>	<b>This Contains the password</b>
<b>4</b>	<b>A_ID</b>	<b>Int</b>	<b>20</b>	<b>This contains the operator ID.</b>

## **CODE EFFICIENCY**

For the code to be efficient to the requirement we used loops and constructs. Visual interfaces have proper validation for invalid inputs using alert and prompt message dialog boxes. Message boxes interact directly with the end user and the appropriate event is fired according to that. For example, in the registration page when user tries to do the same mistake, he is stopped to do so by halting the browser and before that as in the user-id case he is properly suggested for the uniqueness of user id.

An update interface for the control panel we used the generic code but with proper validations to take action necessary to it. In update form the end user is always prompted to select the data which he should update which ensures that only the entry made will be modified if anything has been committed wrong. As soon as the user/ administrator login the user is displayed with details of the work his id related to do. That is what the user is going to access or modify. As the cookies and hidden data transfer from page to page is used the user ones login has to do minimum work to get his required worked done as most of the processes are atomized unto the maximum extent. The data being inputted in to the table or being output on the form from the table is validated properly show that referential as well as data integrity is maintained. The interface is designed in the way the user should have fewer problems in filling the form as well as to do the job on the software. So the code is efficient enough for the person as per the security is concerned the software will be a more robust one as it is worth to say hacker free but to a far extent we are in the way to do the same.

## **OPTIMIZATION OF CODE**

The code is optimized using loops and constructs. Run time controls are generated and variables are declared and assigned values whenever required. Generating run time controls does memory usage and management. Memory is released whenever a control goes out of scope and not required anymore. Memory stack size is also taken into account consider the cost of a query from database. Whenever the connection is not needed for each instance, the connection is being freed and memory is released so that the server should not go down. Only one time one user can update the database. The Functions and procedures are made and called whenever required. A class module is designed for database connectivity. Module is called only whenever a proper connection with database is required. Permission policies taken into account while opening the database and lock is granted whenever required. To optimize the code-naming convention is followed. Prefixes are used to identify the control and interface. Since in coding controls are referred by their name so a complicate name would create more confusion and problem while coding. We used the name based on the function and property of the control. Frames and command buttons are to enable the user to refresh the interface. Code is written in the appropriate event of a control.

The event of the control is decided to optimize the code efficiency. Validation like date is also done using functions and checks on controls. That is a text box prompting to enter the number accepts only numeric values. The code optimized

to check and ensure that the proper value is given as input. Date input box can only contain date format prescribed. Indexing is used for fast search of a record. Code optimization evaluates the software performance and handles errors smartly. A large number of KLOC must be avoided and emphasis must be on the logical aspects rather than physical structure of the program. For example, in the online examination case the data is handled efficiently first the logic in fetching the data randomly from the database and then it is transfer in to array which is further randomized with the answer to output the question paper. Runtime record sets are also defined and controls are unloaded which are generated at runtime. We tried a lot to optimize the code but still are less confident whether the code will handle all possible errors in future that were not in scope during testing. It will be optimized more and more by getting feedback from the end user. The end user will tell the problems encountered and an updated version will be developed with more optimized performance.

## **SOURCE-CODE**

### **ESHOP PROJECT FOLDER**

#### **1. ESHOP-MAIN (PROJECT)**

- **\_\_INIT\_\_.PY**
- **ASGI.PY**
- **URLS.PY**
- **SETTINGS.PY**
- **WSGI.PY**

#### **2. STORE(APPLICATION)**

##### **2.1. MODELS**

- **\_\_INIT.PY\_\_**
- **CATEGORY.PY**
- **CONTACT.PY**
- **CUSTOMER.PY**
- **ORDERS.PY**
- **PRODUCTS.PY**

##### **2.2 TEMPLATES**

- **BASE.HTML**
- **CART.HTML**
- **CONTACT.HTML**
- **INDEX.HTML**
- **LGOIN.HTML**
- **ORDERS.HTML**
- **SEARCH.HTML**
- **SIGNUP.HTML**

##### **2.3 TEMPLATETAGS**

- **CART.PY**
- **CUSTOMER\_FILTER.PY**

##### **2.4 VIEWS**

- **\_\_INIT\_\_.PY**

- **CART.PY**
- **CHECKOUT.PY**
- **CONTACT.PY**
- **HOME.PY**
- **LOGIN.PY**
- **ORDERS.PY**
- **SINGUP.PY**

**2.5 \_\_INIT\_\_.PY**

**2.6 ADMIN.PY**

**2.7 APP.PY**

**2.8 TEST.PY**

**2.9 URLS.PY**

### **3. MIGRATIONS**

**3.1 0001\_initial.PY**

**3.2 0002\_auto\_20210423\_1934.py**

**3.3 0003\_order\_address.py**

**3.4 0004\_order\_phone.py**

**3.5 0005\_order\_date**

**3.6 0006\_order\_status**

**3.7 0007\_contact.py**

**3.8 0008\_customer\_addss.py**

**3.9 0012\_remove\_contacts\_subject\_alter\_contacts\_email\_and\_more.py**

### **4. MANAGE.PY**

## MANAGE.PY

```
#!/usr/bin/env python

"""Django's command-line utility for administrative tasks."""

import os

import sys

def main():

    """Run administrative tasks."""

    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'Eshop.settings')

    try:

        from django. core.management import
execute_from_command_line

    except ImportError as exc:

        raise ImportError(

            "Couldn't import Django. Are you sure it's installed and "

            "available on your PYTHONPATH environment variable? Did you

            "forget to activate a virtual environment?"

        ) from exc

    execute_from_command_line(sys.argv)

if __name__ == '__main__':

    main()
```

## E-SHOP(WSGI.PY)

```
"""
```

WSGI config for Eshop project.

It exposes the WSGI callable as a module-level variable named  
`application`.

For more information on this file, see  
<https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/>  
"""

```
import os
```

```
from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Eshop.settings')
```

```
application = get_wsgi_application()
```



## E-SHOP(URLS.PY)

```
from django.contrib import admin
```

```
from django.urls import path , include
```

```
from django.conf.urls.static import static
```

```
from . import settings
```

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path("", include('store.urls'))  
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## E-SHOP(SETTINGS.PY)

"""

Django settings for Eshop project.

Generated by 'django-admin startproject' using Django 3.1.7.

For more information on this file, see

<https://docs.djangoproject.com/en/3.1/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/3.1/ref/settings/>

"""

import os

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See

<https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/>

# SECURITY WARNING: keep the secret key used in production secret!

```
SECRET_KEY = '-95t%=#4o3@l-(-%ok9*h%n3!0(sdchjn%+_ $5#umaj-  
!3bg*7'
```

```
# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'store'  
]
```

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]
```

```
ROOT_URLCONF = 'Eshop.urls'
```

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  
                'django.template.context_processors.debug',  
                'django.template.context_processors.request',  
                'django.contrib.auth.context_processors.auth',  
                'django.contrib.messages.context_processors.messages',  
            ],  
        },  
    ],  
]
```

```
WSGI_APPLICATION = 'Eshop.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases
```

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```

```
# Password validation
```

```
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [  
    {  
        'NAME':  
        'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',  
    },  
    {  
        'NAME':  
        'django.contrib.auth.password_validation.MinimumLengthValidator',  
    },  
    {  
        'NAME':  
        'django.contrib.auth.password_validation.CommonPasswordValidator',  
    },  
    {  
        'NAME':  
        'django.contrib.auth.password_validation.NumericPasswordValidator',  
    },  
]
```

```
# Internationalization
```

```
# https://docs.djangoproject.com/en/3.1/topics/i18n/
```

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_L10N = True
```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/3.1/howto/static-files/
```

```
STATIC_URL = '/static/'
```

```
STATICFILES_DIRS = [  
    os.path.join(BASE_DIR, "static")  
]
```

```
MEDIA_URL = "/image/download/"
```

```
MEDIA_ROOT = BASE_DIR
```

## **E-SHOP(ASGI.PY)**

"""

ASGI config for Eshop project.

It exposes the ASGI callable as a module-level variable named  
`application`.

For more information on this file, see  
<https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/>  
"""

```
import os
```

```
from django.core.asgi import get_asgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Eshop.settings')
```

```
application = get_asgi_application()
```

## **MODELS(\_\_INIT\_\_.PY)**

```
from .product import Products
from .category import Category
from .customer import Customer
from .orders import Order
```



## MODELS(CUSTOMER.PY)

```
from django.db import models

class Customer(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    phone = models.CharField(max_length=10)
    email=models.EmailField()
    password = models.CharField(max_length=100)

    #to save the data
    def register(self):
        self.save()

    @staticmethod
    def get_customer_by_email(email):
        try:
            return Customer.objects.get(email= email)
        except:
            return False

    def isExists(self):
        if Customer.objects.filter(email = self.email):
            return True

        return False

    def __str__(self):
        return self.first_name
```

## **MODELS(CONTACT.PY)**

```
from django.db import models
```

```
class Contacts(models.Model):
```

```
    name = models.CharField(max_length=50)
```

```
    email = models.CharField(max_length=50)
```

```
    # subject = models.CharField(max_length=10)
```

```
    msg = models.CharField(max_length=100)
```

```
    # sub = models.CharField(max_length=100)
```

```
    # pic = models.ImageField(upload_to="uploads/products")
```

```
    def __str__(self):
```

```
        return self.name
```

## **MODELS(CATEGORY.PY)**

```
from django.db import models
```

```
class Category(models.Model):  
    name= models.CharField(max_length=50)
```

```
    @staticmethod
```

```
    def get_all_categories():  
        return Category.objects.all()
```

```
    def __str__(self):  
        return self.name
```

## MODELS( INIT.PY )

```
from .product import Products
from .category import Category
from .customer import Customer
from .orders import Order
```

## MODELS(PRODUCT.PY)

```
from django.db import models
```

```
from .category import Category
```

```
class Products(models.Model):
```

```
    name = models.CharField(max_length=60)
```

```
    price= models.IntegerField(default=0)
```

```
    category=
```

```
models.ForeignKey(Category,on_delete=models.CASCADE,default=1 )
```

```
    description= models.CharField(max_length=250, default="",  
blank=True, null= True)
```

```
    image= models.ImageField(upload_to='uploads/products/')
```

```
    @staticmethod
```

```
    def get_products_by_id(ids):
```

```
        return Products.objects.filter (id__in=ids)
```

```
    @staticmethod
```

```
    def get_all_products():
```

```
        return Products.objects.all()
```

```
    @staticmethod
```

```
    def get_all_products_by_categoryid(category_id):
```

```
if category_id:

    return Products.objects.filter (category=category_id)
else:

    return Products.get_all_products();
```

## MODELS(ORDER.PY)

```
from django.db import models
```

```
from .product import Products
```

```
from .customer import Customer
```

```
import datetime
```

```
class Order(models.Model):
```

```
    product = models.ForeignKey(Products,  
                                on_delete=models.CASCADE)
```

```
    customer = models.ForeignKey(Customer,  
                                on_delete=models.CASCADE)
```

```
    quantity = models.IntegerField(default=1)
```

```
    price = models.IntegerField()
```

```
    address = models.CharField (max_length=50, default="", blank=True)
```

```
    phone = models.CharField (max_length=50, default="", blank=True)
```

```
    date = models.DateField (default=datetime.datetime.today)
```

```
    status = models.BooleanField (default=False)
```

```
    def placeOrder(self):
```

```
        self.save()
```

```
@staticmethod
```

```
def get_orders_by_customer(customer_id):  
    return Order.objects.filter(customer=customer_id).order_by('-  
date')
```



## **MODELS (CATEGORY.PY)**

```
from django.db import models
```

```
class Category(models.Model):
```

```
    name= models.CharField(max_length=50)
```

```
    @staticmethod
```

```
    def get_all_categories():
```

```
        return Category.objects.all()
```

```
    def __str__(self):
```

```
        return self.name
```

## **MODELS(CONTACT.PY)**

```
from django.db import models
```

```
class Contacts(models.Model):
```

```
    name = models.CharField(max_length=50)
```

```
    email = models.CharField (max_length=50)
```

```
    # subject = models.CharField(max_length=10)
```

```
    msg= models.CharField(max_length=100)
```

```
    # sub=models.CharField(max_length=100)
```

```
    # pic=models.ImageField(upload_to="uploads/products")
```

```
    def __str__(self):
```

```
        return self.name
```

## TEMPLATES(CART.HTML)

```
{% extends 'base.html' %}
```

```
{% block content %}
```

```
{% load cart %}
```

```
{% load custom_filter %}
```

```
<div class="container">
```

```
    <div class="border rounded p-4 m-4">
```

```
        {% comment %} <p class="display-4 pl-4 ml-4 btn btn-warning
rounded-pill">Your Cart</p> {% endcomment %}
```

```
        <p class="display-4 pl-4 ml-4 btn btn-warning rounded-pill"><a
href="store">Add Items(+)</a></p>
```

```
    <hr>
```

```
    <table class="table">
```

```
        <thead>
```

```
            <tr>
```

```
                <th>Sno.</th>
```

```
                <th>Image</th>
```

```
                <th>Product</th>
```

```
                <th>Price</th>
```

```
                <th>Quantity</th>
```

```
                <th>Total</th>
```

```
            </tr>
```

```
        </thead>
```

```
        <tbody>
```

```
{% for product in products %}
<tr>
  <td>{{forloop.counter}}</td>

  <td></td>

  <td>{{product.name}}</td>

  <td>{{product.price|currency}}</td>
  <td>{{product|cart_quantity:request.session.cart}}</td>

<td>{{product|price_total:request.session.cart|currency}}</td>
</tr>

{% endfor %}

</tbody>

<tfoot>
<tr>
  <th colspan="4"></th>
  <th class="" colspan="">Total</th>

<th>{{products|total_cart_price:request.session.cart|currency}}</th>
</tr>
</tfoot>
</table>
<hr>
<div class="m-3 p-3">
```

```
<a href="#" data-toggle="modal" data-target="#exampleModal"
class="btn btn-outline-success border rounded col-lg-3 float-
right">Check out</a>
```

```
</div>
</div>
</div>
```

```
<!-- modal -->
```

```
<!-- Modal -->
```

```
<div class="modal fade" id="exampleModal" tabindex="-1"
role="dialog" aria-labelledby="exampleModalLabel" aria-
hidden="true">
```

```
<div class="modal-dialog" role="document">
```

```
<div class="modal-content">
```

```
<div class="modal-header">
```

```
<h5 class="modal-title" id="exampleModalLabel">
```

```
Check Out Form
```

```
</h5>
```

```
<hr>
```

```
<button type="button" class="close" data-dismiss="modal" aria-
label="Close">
```

```
<span aria-hidden="true">&times;</span>
```

```
</button>
```

```
</div>
```

```
<div class="modal-body">
```

```
<div class="m-2 p-3">
```

```
<form action="/check-out" method="POST">
```

```
{% csrf_token %}
<div class="form-group">
  <label for="">Address</label>
  <input type="text" name="address" id="" class="form-
control" placeholder="" aria-describedby="helpId"required>

</div>
<div class="form-group">
  <label for="">Phone</label>
  <input type="text" name="phone" id="" class="form-
control" placeholder="" aria-describedby="helpId"required>
</div>

  <input type="submit" class="btn float-right btn-outline-
success col-lg-6" value="Check out">
</form>
</div>
</div>
</div>

{% endblock %}
```

## TEMPLATES(BASE.HTML)

```
<!doctype html>
<html lang="en">

<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1,
  shrink-to-fit=no">
  <link rel="stylesheet" href="static/contact.css">
  <link rel="stylesheet" href="static/new.css">
  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap
  p.min.css"
    integrity="sha384-
  Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/
  dAiS6JXm" crossorigin="anonymous">
  {% comment %} mdn kit {% endcomment %}

  <title>Virtula Kirana Store</title>

  <style>
    nav{
      background-color: #8FCE00;
    }
    .one-edge-shadow {
      box-shadow: 0 8px 2px -5px rgb(246, 245, 245);
    }
```

```
.display-8{
  font-weight: 200;
  font-size: 30px;
  font-colour: $teal-200;
}
</style>
</head>

<body>
  <!-- navbar -->

  <nav class="one-edge-shadow
  navbar navbar-light
  sticky-top navbar
  navbar-expand-lg
  text-white">
    <a class="navbar-brand ms-3 text-dark " href="/">
      <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.0/css/all.css"
integrity="sha384-
lZN37f5QGtY3VHgisS14W3ExzMWZxybE1SJSEsQp9S+oqd12jhcu+A56Eb
c1zFSJ" crossorigin="anonymous">

     Virtual Kirana
Store</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent"
    aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
```



```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item active">

      <a class="nav-link" href="/store">(Hurry up get Delivery in 30
min)<span class="sr-only">(current)</span></a>
    </li>

  </ul>
```

```
<ul class=" navbar-nav my-2 my-lg-0">

  <li class="nav-item active">
    <a class="nav-link" href="/cart">Cart
      <span class="badge badge-
success">{{request.session.cart.keys|length}}</span>
      <span class="sr-only">(current)</span></a>
    </li>
```

```
{% if request.session.customer %}
```

```
<li class="nav-item active ">
  <a class="nav-link" href="/orders">Orders<span class="sr-
only">(current)</span></a>
</li>
<li class="nav-item active ">
  <a class="nav-link" href="#">User<span class="sr-
only">(current)</span></a>
```

```
</li>
<li class="nav-item active ">
  <a class="nav-link" href="/logout">Logout<span class="sr-
only">(current)</span></a>
</li>

<li class="nav-item active ">
  <a class="nav-link" href="contact">Contact us <span class="sr-
only">(current)</span></a>
</li>
{% else %}

<li class="nav-item active">
  <a class="nav-link" href="/signup">Signup<span class="sr-
only">(current)</span></a>
</li>
<li class="nav-item active">
  <a class="nav-link" href="/login">Login<span class="sr-
only">(current)</span></a>
</li>
<li class="nav-item active ">
  <a class="nav-link" href="contact">Contact us <span class="sr-
only">(current)</span></a>
</li>
{% endif %}

</ul>

</div>
</nav>
<!-->
```

```
{% block content %}{% endblock %}

<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
  integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KCKRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93
hXpG5KkN"
  crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/pop
per.min.js"
  integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusv
fa0b4Q"
  crossorigin="anonymous"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.
min.js"
  integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76P
VCmYI"
  crossorigin="anonymous"></script>
<div>

</body>
{% comment %} <footer class="ct-footer">
  <div class="container-fluid">
    <form name="contentForm" enctype="multipart/form-data"
method="post" action="">
      <div class="ct-footer-pre text-center-lg">
        <div class="inner">
```

```
<span>Join WebCorpCo to receive updates, news &
events!</span>
</div>
<div class="inner">
  <div class="form-group">
    <input name="formfields[nl_email]" id="nl_email"
class="validate[required]" placeholder="Enter your email address"
type="text" value=""> <label for="nl_email" class="sr-only">Email
Address</label> <button type="submit" class="btn btn-
motive">Join</button>
  </div>
</div>
</div>
</form>
```

```
<h2 class="ct-footer-list-header">About</h2>
<ul>
  <li>
    <a href="">FAQ</a>
  </li>
  <li>
    <a href="">Our Board</a>
  </li>
  <li>
    <a href="">Our Staff</a>
  </li>
  <li>
    <a href="">Contact Us</a>
  </li>
</ul>
</li>
</ul>
```

```
<div class="inner-right">
  <p>Copyright © 2016 WebCorpCo.&nbsp;<a href="">Privacy
Policy</a></p>
  <p><a class="ct-u-motive-color" href="" target="_blank">Web
Design</a> by DigitalUs on <a href="" target="_blank">Solodev
CMS</a></p>
</div>
</div>
</div>
</footer> {% endcomment %}

</html>
```

## TEMPLATES(SIGNUP.HTML)

```
{% extends 'base.html' %}
```

```
{% block content %}
```

```
<div class="container mt-4">
```

```
    <div class="p-3 m-3">
```

```
        <div class="col-lg-5 rounded mx-auto border pt-4">
```

```
            <div class="text-center col">
```

```
                
```

```
        <!-- <hr> -->
```

```
        <hr>
```

```
    </div>
```

```
    <h3 class="alert alert-success rounded-pill" style="text-
align:center">Create an Account</h3>
```

```
    <form action="/signup" method="POST">
```

```
        {% csrf_token %}
```

```
        {% if error%}
```

```
        <div class="alert alert-danger" role="alert">
```

```
            {{error}}
```

```
        </div>
```

```
        {% endif %}
```

```
<!-- firstname -->
<div class="form-group">
  <label for="">First Name</label>
  <input type="text" name="firstname"
    id="" value="{{values.first_name}}" class="form-control
form-control-sm"
    placeholder="">
</div>
```

```
<!-- last name -->
<div class="form-group">
  <label for="">Last Name</label>
  <input type="text" name="lastname"
    id="" value="{{values.last_name}}" class="form-control
form-control-sm"
    >
</div>
```

```
<!-- phone -->
<div class="form-group">
  <label for="">Phone</label>
  <input type="text" name="phone"
    id="" class="form-control form-control-sm"
    value="{{values.phone}}"
    placeholder="9876543210"
    >
</div>
```

```
<!-- email -->
<div class="form-group">
  <label for="">Email</label>
  <input required type="email" name="email" id=""
    value="{{values.email}}"
```

```
        class="form-control-sm form-control"
placeholder="abc@gmail.com">
    </div>
```

```
<!-- password -->
<div class="form-group">
    <label for="">Password</label>
    <input type="password"
name="password"
id=""
class="form-control form-control-sm">
</div>
```

```
<!--Checkme button-->
<div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input"
id="exampleCheck1">
```

```
        <label class="form-check-label d-grid gap-2"
for="exampleCheck1">I accept the <a
href="https://drive.google.com/file/d/197OlicSwbSzKvbt9K6e_s81hGAt
E8o9M/view?usp=sharing" target="_blank">privacy policy</a></label>
    </div>
```

```
    <hr>
    <button type="submit" class="btn btn-sm btn-success col-lg-
12">Create an account</button>
```

```
    </form>
</div>
</div>
</div>
```

```
{% endblock %}
```



## TEMPLATES(SEARCH.HTML)

```
{% extends 'base.html' %}
{% block title %} Search {% endblock title %}
{% block content %}

{% load cart %}
{% load custom_filter %}
<!-- body -->
<div class="container-fluid mt-3">
    <div class="row">

        <!-- filter -->

        <div class="col-lg-3 mx-auto">
            <div class="list-group">

                <a href="/" class="list-group-item list-group-item-
action btn btn-outline-success">All Products</a>

                {% for category in categories %}
                <a href="/?category={{category.id}}"
                    class="list-group-item list-group-item-action
btn btn-outline-success ">{{category.name}}</a>
                {% endfor %}
            </div>
        </div>
    </div>

    <!-- all products -->
    <div id='products' class="col-lg-9 mx-auto">
```

```

<div class="row mx-auto">
  {% for product in products %}
    <div class="card mx-auto mb-3" id="{{product.id}}"
style="width: 18rem;">
      
      <div class="card-body">
        <p class="card-
title">{{product.name}}</p>
        <p class="card-
text"><b>{{product.price | currency}}</b></p>
        <!-- {{product |
is_in_cart:request.session.cart }} -->
      </div>

      <div class="card-footer p-0 no-gutters">

        {% if
product|is_in_cart:request.session.cart %}
          <div class="row no-gutters">
            <form action="/#{{product.id}}"
class="col-2 " method="post">
              {% csrf_token %}
              <input hidden type="text"
name='product' value='{{product.id}}'>
              <input hidden type="text"
name='remove' value='True'>
              <input type="submit"
value=" - " class="btn btn-block btn-success border-right">
            </form>
            <div class="text-center col btn
btn-success">{{product|cart_quantity:request.session.cart}} in
Cart</div>

```

```
<form action="/#{{product.id}}"
class="col-2 " method="post">
    {% csrf_token %}
    <input hidden type="text"
name='product' value '{{product.id}}'>
        <input type="submit"
value="" + " class="btn btn-block btn-success border-left">
    </form>
</div>
{% else %}
<form action="/#{{product.id}}"
method="POST" class="btn-block">
    {% csrf_token %}
    <input hidden type="text"
name='product' value '{{product.id}}'>
        <input type="submit"
class="float-right btn btn-success form-control"
value="Add To Cart">
    </form>
    {% endif %}
</div>
</div>
</div>
{% endfor %}
</div>
</div>
</div>
</div>
</div>
{% endblock %}
```

## TEMPLATES(ORDERS.HTML)

```
{% extends 'base.html' %}

{% block content %}
{% load cart %}
{% load custom_filter %}
<div class="container">
  <div class="border rounded p-4 m-4">
    <p class="display-4 pl-4 ml-4">Your Orders Status | <a
href="store">HOME</a></p>
    {% comment %} <p class="display-5 pl-3 ml-0">Home</p> {%
endcomment %}
    <hr>
    <table class="table">
      <thead>
        <tr>
          <th>Sno.</th>
          <th>Image</th>
          <th>Product</th>
          <th>Date</th>
          <th>Price</th>
          <th>Quantity</th>
          <th>Total</th>
          <th>Status</th>
        </tr>
      </thead>
      <tbody>

        {% for order in orders %}
        <tr>
```

```
<td>{{forloop.counter}}</td>
<td></td>
<td>{{order.product.name}}</td>
<td>{{order.date}}</td>
<td>{{order.price | currency}}</td>
<td>{{order.quantity}}</td>
<td>{{order.quantity | multiply:order.price | currency}}</td>
{% if order.status %}
<td><small class="badge badge-
success">Completed/Delivered</small></td>
{%else%}
<td><small class="badge badge-warning">Pending/In
Transit</small></td>
{% endif %}
</tr>

{% endfor %}

</tbody>

</table>

</div>
</div>

{% endblock %}
```

## TEMPLATES(LOGIN.HTML)

```
{% extends 'base.html' %}
```

```
{% block content %}
```

```
<div class="container">
```

```
  <div class="p-3 m-3">
```

```
    <div class="col-lg-5 rounded mx-auto border pt-4">
```

```
      <div class="text-center col">
```

```
        
```

```
    <hr>
```

```
  </div>
```

```
  <h3 class="alert alert-light rounded-pill" style="text-align:center"
```

```
>Welcome to VKS Login</h3>
```

```
  <form action="/login" method="POST">
```

```
    {% csrf_token %}
```

```
    {% if error%}
```

```
      <div class="alert alert-danger" role="alert">
```

```
        {{error}}
```

```
      </div>
```

```
    {% endif %}
```

```

<!-- email -->
<div class="form-group">
  <label for="">Email</label>
  <input required type="email" name="email" id=""
    value="{{values.email}}"
    class=" form-control-sm form-control"
placeholder="abc@gmail.com">
</div>
<!-- password -->
<div class="form-group">
  <label for="">Password</label>
  <input type="password"
    name="password"
    id=""
    class="form-control form-control-sm" >
</div>
<!--Checkme button-->
<div class="mb-3 form-check">
  <input type="checkbox" class="form-check-input"
id="exampleCheck1" required>
  <label class="form-check-label d-grid gap-2"
for="exampleCheck1">I accept the <a
href="https://drive.google.com/file/d/197OlicSwbSzKvbt9K6e_s81hGAt
E8o9M/view?usp=sharing" target="_blank">privacy policy</a></label>
</div>
<hr>
  <button type="submit" class="btn btn-sm btn-success col-lg-
12">Login</button>
</form>
</div>
</div>
</div>
{% endblock %}

```

## TEMPLATES(INDEX.HTML)

```
{% extends 'base.html' %}
```

```
{% block content %}
```

```
{% load cart %}
```

```
{% load custom_filter %}
```

```
<!-- body -->
```

```
<div class="container-fluid mt-3">
```

```
    <div class="row">
```

```
        <!-- filter -->
```

```
        <div class="col-lg-3 mx-auto">
```

```
            <div class="list-group">
```

```
                <a href="/" class="list-group-item list-group-item-  
action btn btn-outline-success">All Products</a>
```

```
                {% for category in categories %}
```

```
                <a href="/?category={{category.id}}"
```

```
                    class="list-group-item list-group-item-action  
btn btn-outline-success ">{{category.name}}</a>
```

```
                {% endfor %}
```

```
            </div>
```

```
        </div>
```

```
    <!-- all products -->
```

```
    <div id='products' class="col-lg-9 mx-auto">
```

```
        <div class="row mx-auto">
```



```

        {% for product in products %}
        <div class="card mx-auto mb-3" id="{{product.id}}"
style="width: 18rem;">
            
            <div class="card-body">
                <p class="card-
title">{{product.name}}</p>
                <p class="card-
text"><b>{{product.price | currency}}</b></p>
                <!-- {{product |
is_in_cart:request.session.cart }} -->
            </div>

            <div class="card-footer p-0 no-gutters">

                {% if
product|is_in_cart:request.session.cart %}
                <div class="row no-gutters">
                    <form action="/#{{product.id}}"
class="col-2 " method="post">
                        {% csrf_token %}
                        <input hidden type="text"
name='product' value='{{product.id}}'>
                        <input hidden type="text"
name='remove' value='True'>
                        <input type="submit"
value=" - " class="btn btn-block btn-success border-right">
                    </form>
                    <div class="text-center col btn
btn-success">{{product|cart_quantity:request.session.cart}} in
Cart</div>

```

```

                                <form action="/#{{product.id}}"
class="col-2 " method="post">
                                {% csrf_token %}
                                <input hidden type="text"
name='product' value='{{product.id}}'>
                                <input type="submit"
value=" " + " class="btn btn-block btn-success border-left">
                                </form>
                                </div>
                                {% else %}
                                <form action="/#{{product.id}}"
method="POST" class="btn-block">
                                {% csrf_token %}
                                <input hidden type="text"
name='product' value='{{product.id}}'>
                                <input type="submit"
class="float-right btn btn-success form-control"
                                value="Add To Cart">
                                </form>
                                {% endif %}

                                </div>

                                </div>
                                {% endfor %}

{% endblock %}
```

## TEMPLATES(CONTACT.HTML)

```
{% extends 'base.html' %}
{%load static%}
{% block content %}
<!--Section: Contact v.2-->
<div class="container" my="4" py="4">
<section class="mb-4">

    <!--Section heading-->
    <h2 class="h1-responsive font-weight-bold text-center my-4">Contact us</h2>
    <!--Section description-->
    <p class="text-center w-responsive mx-auto mb-5">Do you have any
questions? Please do not hesitate to contact us directly. Our team will
come back to you within
    a matter of hours to help you.</p>

    <div class="row">

        <!--Grid column-->
        <div class="col-md-9 mb-md-0 mb-5">
            <form id="contact-form" name="contact-form" action="contact"
method="POST">
                {%csrf_token %}
                <!--Grid row-->
                <div class="row">

                    <!--Grid column-->
                    <div class="col-md-6">
                        <div class="md-form mb-0">
```

```
        <label for="name" class="">Name</label>
        <input type="text" id="name" name="name"
class="form-control" required>

    </div>
</div>
<!--Grid column-->

<!--Grid column-->
<div class="col-md-6">
    <div class="md-form mb-0">
        <label for="email" class=""> Email</label>
        <input type="text" id="email" name="email"
class="form-control" required>

    </div>
</div>
<!--Grid column-->

</div>
<!--Grid row-->

<!--Grid row-->
{% comment %} <div class="row">
    <div class="col-md-12">
        <div class="md-form mb-0">
            <label for="subject" class="">Subject</label>
            <input type="text" id="subject" name="subject"
class="form-control">

        </div>
    </div>
</div> {% endcomment %}
```

```
<!--Grid row-->

<!--Grid row-->
<div class="row">

    <!--Grid column-->
    <div class="col-md-12">

        <div class="md-form">
            <label for="message">Message</label>
            <textarea type="text" id="message" name="msg"
rows="2" class="form-control md-textarea"></textarea required>

        </div>

    </div>
</div>
<!--Grid row-->

<br>
<!--Grid row-->
<div class="row">

    <!--Grid column-->
    {% comment %} <div class="col-md-12">

        <div class="md-form">
            <label for="file">Upload Image (Optional)</label>
            <input type="file" id="file" name="msg" rows="2"
class="form-control md-">

        </div>
```

```
        </div> {% endcomment %}
    </div>
    <!--Grid row-->
<br>
```

```
    <div class="text-center text-md-left">
        <div class="container" ><a class="btn btn-primary"
onclick="document.getElementById('contact-
form').submit();">Send</a>
        </div></div>
        <div class="status"></div>
    </div> </form>
    <!--Grid column-->
```

```
    <!--Grid column-->
    <div class="col-md-3 text-center">
        <ul class="list-unstyled mb-0">
            <li><i class="fas fa-map-marker-alt fa-2x"></i>
                <p><a
href="https://www.google.com/maps/place/Laxmi+Nagar,+Block+S1,+
Nanakpura,+Shakarpur,+New+Delhi,+Delhi+110092/@28.6275489,77.2
697558,15z/data=!3m1!4b1!4m5!3m4!1s0x390ce35319b6a7ff:0x127dc
a80423ad527!8m2!3d28.6275609!4d77.2784081"
target="_blank">laxmi Nagar, Delhi-110092,INDIA</a> </p>
                </li>

            <li><i class="fas fa-phone mt-4 fa-2x"></i>
                <p><a href="tel:+917834864894">+91 7834864894</a></p>
            </li>

            <li><i class="fas fa-envelope mt-4 fa-2x"></i>
```

```
        <p><a
href="mailto:vks@gmail.com">vks@gmail.com</a></p>
    </li>
</ul>
</div>
<!--Grid column-->
```

```
</div>
```

```
</section>
```

```
</div>
```

```
<!--Section: Contact v.2-->
```

```
{% endblock %}
```

## TAMPLATETAGS(CART.PY)

```
from django import template
register = template.Library ()
@register.filter (name='is_in_cart')
def is_in_cart(product, cart):
    keys = cart.keys ()
    for id in keys:
        if int (id) == product.id:
            return True
    return False;
@register.filter (name='cart_quantity')
def cart_quantity(product, cart):
    keys = cart.keys ()
    for id in keys:
        if int (id) == product.id:
            return cart.get (id)
    return 0;
@register.filter (name='price_total')
def price_total(product, cart):
    return product.price * cart_quantity (product, cart)

@register.filter (name='total_cart_price')
def total_cart_price(products, cart):
    sum = 0;
    for p in products:
        sum += price_total (p, cart)

    return sum
```



## Tamplatetags (Custom\_filter.py)

```
from django import template
```

```
register = template.Library()
```

```
@register.filter(name='currency')
```

```
def currency(number):
```

```
    return "Rs "+str(number)
```

```
@register.filter(name='multiply')
```

```
def multiply(number , number1):
```

```
    return number * number1
```

## VIEWS(HOME.PY)

```
from django.shortcuts import render , redirect , HttpResponseRedirect
from store.models.product import Products
from store.models.category import Category
from django.views import View
```

# Create your views here.

```
class Index(View):
```

```
    def post(self , request):
        product = request.POST.get('product')
        remove = request.POST.get('remove')
        cart = request.session.get('cart')
        if cart:
            quantity = cart.get(product)
            if quantity:
                if remove:
                    if quantity<=1:
                        cart.pop(product)
                    else:
                        cart[product] = quantity-1
                else:
                    cart[product] = quantity+1
            else:
                cart[product] = 1
        else:
            cart = {}
            cart[product] = 1
```

```
request.session['cart'] = cart
print('cart' , request.session['cart'])
return redirect('homepage')
```

```
def get(self , request):
    # print()
    return HttpResponseRedirect(f'/store{request.get_full_path()[1:]}')
```

```
def store(request):
    cart = request.session.get('cart')
    if not cart:
        request.session['cart'] = {}
    products = None
    categories = Category.get_all_categories()
    categoryID = request.GET.get('category')
    if categoryID:
        products = Products.get_all_products_by_categoryid(categoryID)
    else:
        products = Products.get_all_products();

    data = {}
    data['products'] = products
    data['categories'] = categories

    print('you are : ', request.session.get('email'))
    return render(request, 'index.html', data)
```

## VIEWS(CONTACT.PY)

```
import email
from django.shortcuts import render, redirect
from django.http import HttpResponseRedirect

# from django.contrib.auth.hashers import check_password
from store.models.contact import Contacts
from django.views import View
# from store.models.product import Products
# from store.models.orders import Order

class Contact(View):
    def get(self, request):
        return render(request, 'contact.html')
s
    def post(self, request):
        if request.method == 'POST':
            name=request.POST.get('name')
            email=request.POST.get('email')
            # sub=request.POST.get('subject')
            msg=request.POST.get('msg')
            # pic=request.POST.get('pic')

            con=Contacts(name=name,email=email,msg=msg)
            con.save()
            return redirect('contact')
```

## VIEWS(CHECKOUT.PY)

```
from django.shortcuts import render, redirect
from django.contrib.auth.hashers import check_password
from store.models.customer import Customer
from django.views import View
```

```
from store.models.product import Products
from store.models.orders import Order
```

```
class CheckOut(View):
    def post(self, request):
        address = request.POST.get('address')
        phone = request.POST.get('phone')
        customer = request.session.get('customer')
        cart = request.session.get('cart')
        products = Products.get_products_by_id(list(cart.keys()))
        print(address, phone, customer, cart, products)

        for product in products:
            print(cart.get(str(product.id)))
            order = Order(customer=Customer(id=customer),
                           product=product,
                           price=product.price,
                           address=address,
                           phone=phone,
                           quantity=cart.get(str(product.id)))
            order.save()
        request.session['cart'] = {}
        return redirect('store')
```

## VIEWS(CART.PY)

```
from django.shortcuts import render , redirect
```

```
from django.contrib.auth.hashers import check_password
```

```
from store.models.customer import Customer
```

```
from django.views import View
```

```
from store.models.product import Products
```

```
class Cart(View):
```

```
    def get(self , request):
```

```
        ids = list(request.session.get('cart').keys())
```

```
        products = Products.get_products_by_id(ids)
```

```
        print(products)
```

```
        return render(request , 'cart.html' , {'products' : products} )
```

## VIEWS(SIGNUP.PY)

```
from django.shortcuts import render, redirect

from django.contrib.auth.hashers import make_password

from store.models.customer import Customer
from django.views import View
class Signup (View):
    def get(self, request):
        return render (request, 'signup.html')
    def post(self, request):
        postData = request.POST
        first_name = postData.get ('firstname')
        last_name = postData.get ('lastname')
        phone = postData.get ('phone')
        email = postData.get ('email')
        password = postData.get ('password')
        # validation
        value = {
            'first_name': first_name,
            'last_name': last_name,
            'phone': phone,
            'email': email
        }
        error_message = None

        customer = Customer (first_name=first_name,
                              last_name=last_name,
                              phone=phone,
                              email=email,
```

```
        password=password)
    error_message = self.validateCustomer (customer)
    if not error_message:
        print (first_name, last_name, phone, email, password)
        customer.password = make_password (customer.password)
        customer.register ()
        return redirect ('homepage')
    else:
        data = {
            'error': error_message,
            'values': value
        }
        return render (request, 'signup.html', data)

def validateCustomer(self, customer):
    error_message = None
    if (not customer.first_name):
        error_message = "Please Enter your First Name !!"
    elif len (customer.first_name) < 3:
        error_message = 'First Name must be 3 char long or more'
    elif not customer.last_name:
        error_message = 'Please Enter your Last Name'
    elif len (customer.last_name) < 3:
        error_message = 'Last Name must be 3 char long or more'
    elif not customer.phone:
        error_message = 'Enter your Phone Number'
    elif len (customer.phone) < 10:
        error_message = 'Phone Number must be 10 char Long'
    elif len (customer.password) < 5:
        error_message = 'Password must be 5 char long'
    elif len (customer.email) < 5:
        error_message = 'Email must be 5 char long'
    elif customer.isExists ():
```



```
    error_message = 'Email Address Already Registered..'
    # saving

    return error_message
```

## VIEWS(ORDERS.PY)

```
from django.shortcuts import render, redirect
from django.contrib.auth.hashers import check_password
from store.models.customer import Customer
from django.views import View
from store.models.product import Products
from store.models.orders import Order
from store.middlewares.auth import auth_middleware

class OrderView(View):

    def get(self , request ):
        customer = request.session.get('customer')
        orders = Order.get_orders_by_customer(customer)
        print(orders)
        return render(request , 'orders.html' , {'orders' : orders})
```

## VIEWS(LOGIN.PY)

```
from django.shortcuts import render , redirect , HttpResponseRedirect
from django.contrib.auth.hashers import check_password
from store.models.customer import Customer
from django.views import View
```

```
class Login(View):
```

```
    return_url = None
```

```
    def get(self, request):
```

```
        Login.return_url = request.GET.get ('return_url')
```

```
        return render (request, 'login.html')
```

```
    def post(self, request):
```

```
        email = request.POST.get ('email')
```

```
        password = request.POST.get ('password')
```

```
        customer = Customer.get_customer_by_email (email)
```

```
        error_message = None
```

```
        if customer:
```

```
            flag = check_password (password, customer.password)
```

```
            if flag:
```

```
                request.session['customer'] = customer.id
```

```
            if Login.return_url:
```

```
                return HttpResponseRedirect (Login.return_url)
```

```
            else:
```

```
                Login.return_url = None
```

```
                return redirect ('homepage')
```

```
        else:
```

```
            error_message = 'Invalid !!'
```

```
    else:
        error_message = 'Invalid !!'

    print (email, password)
    return render (request, 'login.html', {'error': error_message})

def logout(request):
    request.session.clear()
    return redirect('login')
```

## Migrations (0001\_initial.py)

# Generated by Django 3.1.7 on 2021-04-23 12:47

```
import datetime
```

```
from django.db import migrations, models
```

```
import django.db.models.deletion
```

```
class Migration(migrations.Migration):
```

```
    initial = True
```

```
    dependencies = [  
    ]
```

```
    operations = [  
        migrations.CreateModel(  
            name='Category',  
            fields=[  
                ('id', models.AutoField(auto_created=True,  
primary_key=True, serialize=False, verbose_name='ID')),
```

```
        ('name', models.CharField(max_length=50)),
    ],
),
migrations.CreateModel(
    name='Customer',
    fields=[
        ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
        ('first_name', models.CharField(max_length=50)),
        ('last_name', models.CharField(max_length=50)),
        ('phone', models.CharField(max_length=10)),
        ('email', models.EmailField(max_length=254)),
        ('password', models.CharField(max_length=100)),
    ],
),
migrations.CreateModel(
    name='Products',
    fields=[
        ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
        ('name', models.CharField(max_length=60)),
        ('price', models.IntegerField(default=0)),
```

```
        ('description', models.CharField(blank=True,
default='', max_length=250, null=True)),
        ('image',
models.ImageField(upload_to='uploads/products/')),
        ('category', models.ForeignKey(default=1,
on_delete=django.db.models.deletion.CASCADE,
to='store.category')),
    ],
),
migrations.CreateModel(
    name='Order',
    fields=[
        ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
        ('quantity', models.IntegerField(default=1)),
        ('price', models.IntegerField()),
        ('address', models.CharField(blank=True, default='',
max_length=50)),
        ('phone', models.CharField(blank=True, default='',
max_length=50)),
        ('date',
models.DateField(default=datetime.datetime.today)),
```

```
        ('status', models.BooleanField(default=False)),
        ('customer',
models.ForeignKey(on_delete=django.db.models.deletion.C
ASCASCADE, to='store.customer')),
        ('product',
models.ForeignKey(on_delete=django.db.models.deletion.C
ASCASCADE, to='store.products')),
    ],
),
]
```



## Migrations (0002\_auto\_20210423\_1934.py)

# Generated by Django 3.1.7 on 2021-04-23 14:04

```
from django.db import migrations
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [  
        ('store', '0001_initial'),  
    ]
```

```
    operations = [  
        migrations.RemoveField(  
            model_name='order',  
            name='address',  
        ),  
        migrations.RemoveField(  
            model_name='order',  
            name='date',  
        ),  
        migrations.RemoveField(  

```

```
        model_name='order',  
        name='phone',  
    ),  
    migrations.RemoveField(  
        model_name='order',  
        name='status',  
    ),  
]
```

## Migrations (0003\_order\_address.py)

# Generated by Django 3.1.7 on 2021-04-23 14:04

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [  
        ('store', '0002_auto_20210423_1934'),  
    ]
```

```
    operations = [  
        migrations.AddField(  
            model_name='order',  
            name='address',  
            field=models.CharField(blank=True, default="",  
max_length=50),  
        ),  
    ]
```

## Migrations (0004\_order\_phone.py)

# Generated by Django 3.1.7 on 2021-04-23 14:05

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [  
        ('store', '0003_order_address'),  
    ]
```

```
    operations = [  
        migrations.AddField(  
            model_name='order',  
            name='phone',  
            field=models.CharField(blank=True, default="",  
max_length=50),  
        ),  
    ]
```

## Migrations (0005\_order\_date.py)

# Generated by Django 3.1.7 on 2021-04-23 14:05

```
import datetime
```

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [
```

```
        ('store', '0004_order_phone'),
```

```
    ]
```

```
    operations = [
```

```
        migrations.AddField(
```

```
            model_name='order',
```

```
            name='date',
```

```
            field=models.DateField(default=datetime.datetime.today),
```

```
        ),
```

```
    ]
```

## Migrations (0006\_order\_status.py)

# Generated by Django 3.1.7 on 2021-04-23 14:06

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [  
        ('store', '0005_order_date'),  
    ]
```

```
    operations = [  
        migrations.AddField(  
            model_name='order',  
            name='status',  
            field=models.BooleanField(default=False),  
        ),  
    ]
```

## Migrations (0007\_contact.py)

# Generated by Django 4.0.4 on 2022-06-03 06:56

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [
```

```
        ('store', '0006_order_status'),
```

```
    ]
```

```
    operations = [
```

```
        migrations.CreateModel(
```

```
            name='Contact',
```

```
            fields=[
```

```
                ('id', models.AutoField(auto_created=True,
```

```
                primary_key=True, serialize=False, verbose_name='ID')),
```

```
                ('first_name', models.CharField(max_length=50)),
```

```
                ('last_name', models.CharField(max_length=50)),
```

```
                ('phone', models.CharField(max_length=10)),
```

```
        ('email', models.EmailField(max_length=254)),  
        ('password', models.CharField(max_length=100)),  
    ],  
),  
]
```



## Migrations (0008\_customer\_addss.py)

# Generated by Django 4.0.4 on 2022-06-04 06:22

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [  
        ('store', '0007_contact'),  
    ]
```

```
    operations = [  
        migrations.AddField(  
            model_name='customer',  
            name='addss',  
            field=models.CharField(default='null',  
max_length=100),  
        ),  
    ]
```

## Migrations

### (0012\_remove\_contacts\_subject\_alter\_contacts\_email\_and\_more.py)

# Generated by Django 4.0.4 on 2022-06-06 06:29

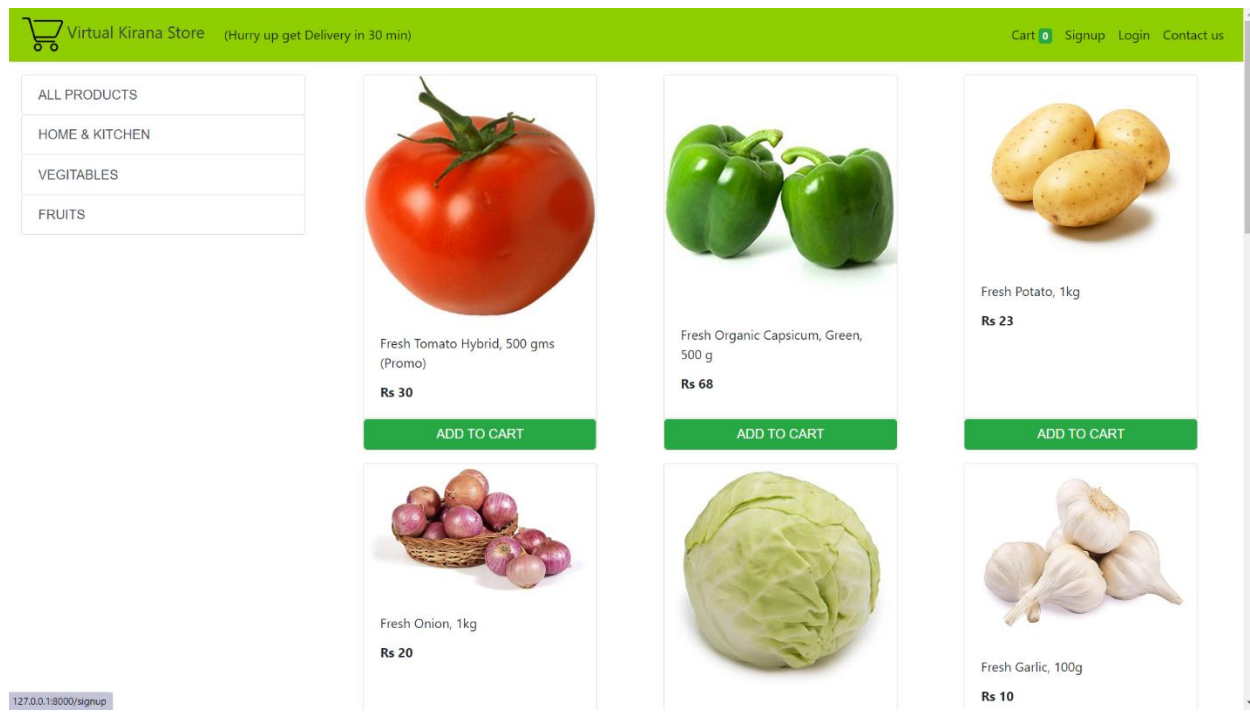
```
from django.db import migrations
```


```
class Migration(migrations.Migration):
```

```
    dependencies = [
        ('store', '0017_alter_contacts_pic'),
    ]
```


```
    operations = [
        migrations.RemoveField(
            model_name='contacts',
            name='pic',
        ),
    ]
```

## Output Screen



 Virtual Kirana Store (Hurry up get Delivery in 30 min)

Cart **0** [Signup](#) [Login](#) [Contact us](#)



Create an Account

First Name

Last Name

Phone

9876543210


Email

abc@gmail.com


Password

☐ I accept the [privacy policy](#)

CREATE AN ACCOUNT

 Virtual Kirana Store (Hurry up get Delivery in 30 min)

Cart **0** Signup Login Contact us



Create an Account

First Name

Sagar

Last Name

Arora

Phone

7834864894

Email

sagararora@gmail.com

Password

\*\*\*\*\*

☒ I accept the [privacy policy](#)

CREATE AN ACCOUNT



### Welcome to VKS Login

Email

abc@gmail.com

Password

☐ I accept the [privacy policy](#)

LOGIN



### Welcome to VKS Login

Email


sagararora@gmail.com


Password

\*\*\*\*\*

☒ I accept the [privacy policy](#)

LOGIN

 Virtual Kirana Store (Hurry up get Delivery in 30 min)


Cart  Orders User Logout Contact us

ALL PRODUCTS

HOME & KITCHEN


VEGITABLES

FRUITS




Fresh Tomato Hybrid, 500 gms  
(Promo)  
**Rs 30**

ADD TO CART




Fresh Organic Capsicum, Green,  
500 g  
**Rs 68**

ADD TO CART





Fresh Potato, 1kg  
**Rs 23**

ADD TO CART



Fresh Onion, 1kg  
**Rs 20**





Fresh Garlic, 100g  
**Rs 10**





### Contact us

Do you have any questions? Please do not hesitate to contact us directly. Our team will come back to you within a matter of hours to help you.

Name

Email

Message

SEND




[Iaxmi Nagar, Delhi-110092,INDIA](#)



[+91 7834864894](#)



[vks@gmail.com](#)

 Virtual Kirana Store (Hurry up get Delivery in 30 min)


Cart 0 Signup Login Contact us

ALL PRODUCTS

HOME & KITCHEN


VEGITABLES

FRUITS




Fresh Tomato Hybrid, 500 gms (Promo)  
Rs 30

ADD TO CART




Fresh Organic Capsicum, Green, 500 g  
Rs 68

ADD TO CART




Fresh Potato, 1kg  
Rs 23

ADD TO CART




Fresh Onion, 1kg  
Rs 20

ADD TO CART




ADD TO CART



Fresh Garlic, 100g  
Rs 10

ADD TO CART

 Virtual Kirana Store (Hurry up get Delivery in 30 min)


Cart 0 Signup Login Contact us

ALL PRODUCTS

HOME & KITCHEN


VEGETABLES

FRUITS




Fresh Papaya Raw, 1Pc (500-700g)  
Rs 43

ADD TO CART




Fresh Orange, Valencia, 4 Pcs  
Rs 60

ADD TO CART




Fresh Mosambi, 1kg  
Rs 69


ADD TO CART



Fresh Banana Robusta, 1kg



155

 Virtual Kirana Store (Hurry up get Delivery in 30 min)


Cart 0 Signup Login Contact us

ALL PRODUCTS

HOME & KITCHEN

VEGETABLES


FRUITS



Pigeon Polypropylene Mini Handy and Compact Chopper with 3 B

Rs 545


ADD TO CART



Scotch-Brite Bathroom Brush with abrasive scrubber for super

Rs 150


ADD TO CART



Scotch-Brite Stainless Steel Utensil Scrubber


Rs 40

ADD TO CART




PSB 01 Deluxe Mops

Rs 1595



PSB 01 Deluxe Mops

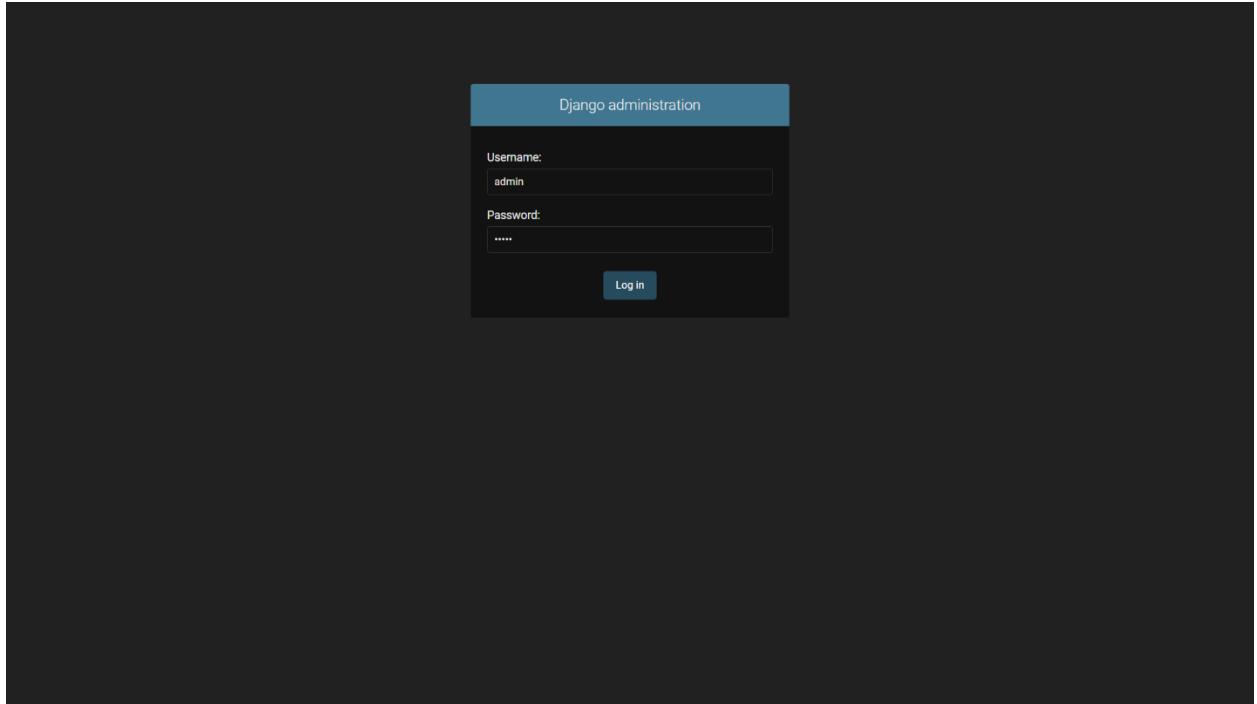
Rs 1595



PSB 01 Deluxe Mops

Rs 1595

127.0.0.1:8000/?category=28



Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Change

Users

+ Add

Change

STORE

Categories

+ Add

Change

Contacts

+ Add

Change

Customers

+ Add

Change

Orders

+ Add

Change

Productss

+ Add

Change

Recent actions

My actions

✖ Sagar

Contacts

+ Products object (137)

Products

+ Products object (136)

Products

+ Products object (135)

Products

+ Products object (134)

Products

+ Products object (133)

Products

+ Products object (132)

Products

+ Products object (131)

Products

+ Products object (130)

Products

+ Products object (129)

Products

Django administration

WELCOME **ADMIN** [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Store > Categorys

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

- Groups [+ Add](#)
- Users [+ Add](#)

**STORE**

- Categorys** [+ Add](#)
- Contacts [+ Add](#)
- Customers [+ Add](#)
- Orders [+ Add](#)
- Productss [+ Add](#)

«

Select category to change

Action:  Go 0 of 3 selected

☐ CATEGORY

☐ Fruits

☐ Vegetables

☐ Home & Kitchen

3 categorys

[ADD CATEGORY +](#)

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home » Store » Categorys » Fruits

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categorys + Add

Contacts + Add

Customers + Add

Orders + Add

Productss + Add

Change category

HISTORY

Fruits

Name: Fruits

Delete

Save and add another

Save and continue editing

SAVE



The screenshot displays the Django administration interface for a project named 'Virtual Kirana Store'. The top navigation bar is dark blue with the text 'Django administration' on the left and 'WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT' on the right. Below the navigation bar, the breadcrumb trail reads 'Home > Store > Contactss'. The left sidebar contains a search bar and a list of menu items under two categories: 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'STORE' (Categories, Contactss, Customers, Orders, Products). The 'Contactss' item is highlighted. The main content area is titled 'Select contacts to change' and features an 'ADD CONTACTS +' button. Below this, there is an 'Action:' dropdown menu, a 'Go' button, and a status '0 of 1 selected'. A table lists the contacts, with one entry 'sagar' under the 'CONTACTS' header. At the bottom of the table, it says '1 contacts'.

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Store > Contactss

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

- Groups + Add
- Users + Add

STORE

- Categories + Add
- Contactss + Add
- Customers + Add
- Orders + Add
- Products + Add

Select contacts to change

ADD CONTACTS +

Action: Go 0 of 1 selected

CONTACTS
sagar

1 contacts

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Store > Contactss > sagar

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categorys + Add

Contactss + Add

Customers + Add

Orders + Add

Productss + Add

Change contacts

HISTORY

sagar

Name: sagar

Email: SAGARARORA.A63@GMAIL.COM

Msg: probelm

Delete

Save and add another

Save and continue editing

SAVE

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Store > Customers

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categorys + Add

Contacts + Add

Customers + Add

Orders + Add

Productss + Add

Select customer to change

ADD CUSTOMER +

Action: Go 0 of 3 selected

☐ CUSTOMER

☐ Sagar

☐ deepak

☐ sagar

3 customers

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Store > Customers > Sagar

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categorys + Add

Contacts + Add

Customers + Add

Orders + Add

Products + Add

Change customer

HISTORY

Sagar

First name: Sagar

Last name: Arora

Phone: 7834864894

Email: sagararora@gmail.com

Password: pbkdf2\_sha256\$320000\$ocBAR6kLHHCH02

Delete

Save and add another

Save and continue editing

SAVE

The screenshot displays the Django administration interface for a project named 'Virtual Kirana Store'. The top navigation bar is dark blue with the text 'Django administration' on the left and 'WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT' on the right. Below the navigation bar, a breadcrumb trail shows 'Home > Store > Orders'. The left sidebar contains a search bar and a list of menu items categorized under 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'STORE' (Categorys, Contacts, Customers, Orders, Productss). The 'Orders' menu item is highlighted. The main content area is titled 'Select order to change' and features an 'ADD ORDER +' button. It includes an 'Action:' dropdown menu, a 'Go' button, and a status '0 of 1 selected'. Below this, there are checkboxes for 'ORDER' and 'Order object (63)'. At the bottom, it indicates '1 order'.

Django administration

WELCOME, ADMIN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Store > Orders > Order object (63)

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

STORE

Categorys [+ Add](#)

Contacts [+ Add](#)

Customers [+ Add](#)

Orders [+ Add](#)

Products [+ Add](#)

Change order

Order object (63) [HISTORY](#)

Product: Products object (105) [✎](#) [+](#)

Customer: sagar [✎](#) [+](#)

Quantity: 1

Price: 30

Address: Laxmi nagar delhi 110092

Phone: 1234567890

Date: 2022-06-06 [Today](#) [📅](#)  
Note: You are 5.5 hours ahead of server time.

☐ Status

Delete

Save and add another

Save and continue editing

SAVE

Django administration

WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Store > Products

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categorys + Add

Contacts + Add

Customers + Add

Orders + Add

Products + Add

Select products to change

ADD PRODUCTS +

Action:  Go 0 of 30 selected

NAME	PRICE	CATEGORY
Aashirvaad Shudh Chakki Atta, 10kg Pack, 100 % Whole Wheat A	368	Home & Kitchen
Harpic Disinfectant Toilet Cleaner Liquid, Original - 1 L	178	Home & Kitchen
Lizol Disinfectant Surface & Floor Cleaner Liquid, Citrus -	356	Home & Kitchen
Gala Spin Mop Extendable Handle/Stick Rod with Microfiber Re	600	Home & Kitchen
PSB 01 Deluxe Mops	1595	Home & Kitchen
Gala Double Lip Wiper (Multicolour)	220	Home & Kitchen
Scotch-Brite Stainless Steel Utensil Scrubber	40	Home & Kitchen
Scotch-Brite Bathroom Brush with abrasive scrubber for super	150	Home & Kitchen
Pigeon Polypropylene Mini Handy and Compact Chopper with 3 B	545	Home & Kitchen
Fresh Coconut - Medium, 1 Pc (350g-450g)	29	Fruits
Fresh Papaya, 1 Piece	58	Fruits
Fresh Banana Robusta, 1kg	66	Fruits
Fresh Mosambi, 1kg	69	Fruits
Fresh Orange, Valencia, 4 Pcs	60	Fruits
Fresh Produce Lemon 3 Pieces (90g - 110g)	12	Vegetables
Fresh Bottle Gourd - Round, 500 g	42	Vegetables
Fresh Parwal, 500g	36	Vegetables
Fresh Papaya Raw, 1Pc (500-700g)	43	Fruits
Fresh Produce Drum Stick, 4 Pieces	40	Vegetables

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Store > Productss > Products object (137)

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categorys + Add

Contacts + Add

Customers + Add

Orders + Add

Productss + Add

Change products

Products object (137)

HISTORY

Name: Aashirvaad Shudh Chakki Atta, 10kg Pack, 10

Price: 368

Category: Home & Kitchen

Description: Aashirvaad Shudh Chakki Atta, 10kg Pack, 10

Image: Currently: uploads/products/41hJfyaExDL.jpg  
Change: Choose File No file chosen

Delete

Save and add another

Save and continue editing

SAVE



Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Store > Productss > Products object (128)

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categorys + Add

Contacts + Add

Customers + Add

Orders + Add

Productss + Add

Change products

Products object (128)

HISTORY

Name:

Fresh Coconut - Medium, 1 Pc (350g-450g)

Price:

29

Category:

Fruits

Description:

Fresh Coconut - Medium, 1 Pc (350g-450g)

Image:

Currently: uploads/products/811ThUQW/AZL\_AC\_LU480\_FMwebp\_QL65\_webp

Change: Choose File No file chosen

Delete

Save and add another

Save and continue editing

SAVE

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Store > Productss > Products object (122)

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categorys + Add

Contacts + Add

Customers + Add

Orders + Add

Productss + Add

Change products

Products object (122)

HISTORY

Name:

Fresh Produce Lemon 3 Pieces (90g - 110g)

Price:

12

Category:

Vegitables

Description:

Rich in vitamin C

Image:

Currently: uploads/products/81k83pTVP9L\_SX522.jpg

Change: Choose File No file chosen

Delete

Save and add another

Save and continue editing

SAVE

## TESTING AND DEBUGGING

Testing is the filter to catch defects before they are “discovered” by the user or end user. Every time a user open the websiet, he/she generates a “test-case”. We tried the test cases to find the defects first since software development is a human activity and there may be potential errors. So testing before the product is delivered helped us to assure the quality and saved resources.

### Testing Objective

Objective of testing is to find errors, demonstrate that software functions that satisfy the specification, reduce the number of errors detected by user, have “confidence” in the software system. A good test always tries to discover undetected errors and is successful when errors are found since zero defects is not possible. There is always a condition or usage that can lead to an incorrect behavior.

### Testing Steps

We started testing each individual new component and worked out as unit test, integration test, high order test, user, order acceptance testing and different testing techniques are used at different times in the process.

### Testing Techniques

Following testing techniques are used.

**White Box Testing**

In white box testing we exercises the internal logic of a program, examine internal program structure and derived tests from an examination of the program's logic, tested internal paths and working of the software. To perform this we used to develop test cases for unit and integration testing.

**Black Box Testing**

In black box testing we exercises the input and output requirements of a program, tested by conducting specifications of requirement for which the software should do. This test is derived from the I/O specification and used in most functional tests.

**White Box Strategies**

White box testing strategies uses the control structure of the program and design to derive test cases.

**Basis Path Testing**

Test cases are derived to exercise the basis set that guarantee that every statement in the program is executed at least once during testing. It is applied in series of steps.

**Test case for login button of the login form:**

**Test case 1:** All the following combinations are tested

    Password is valid and Username is valid

Password is valid and Username is invalid

Password is invalid and Username is invalid

Password is invalid and Username is valid

**Expected results:** No error crossing this stage when Password & Username are both permissible combination of characters. In all other case error message is displayed.

**Test case 2:** Following input combination is tried where permissible set of characters for username & password are entered which are non-existent in table.

- Password is valid, existent and Username is valid, existent
- Password is valid, existent and Username is valid, nonexistent
- Password is valid, nonexistent and Username is valid, existent
- Password is valid, nonexistent and Username is valid, nonexistent

**Expected Results:** If no entry into the system then error message displayed and user is linked to the registration page.

**Test Case 3:** To test this both username & password to be entered must be valid & existent in the registration table.

Password = Valid, Existent and Username = Valid, Existent. Similarly, all the other internal paths & logic can be tested.

## Control Structure Testing

### Condition testing

Condition testing is a test case design method that exercises the logical conditions contained in a program module. If a condition is incorrect, then at least one of the conditions is incorrect. So, types of errors in a condition include Boolean operator error (incorrect/missing/extra Boolean operators), Boolean variable error, Boolean parenthesis error, relational operator error and arithmetic expression error. It mainly focuses on testing each condition in the program. The condition testing strategies to be employed are:

**Branch testing:** For a compound condition C, the true and false branches of C and every simple condition in C need to be executed at least once.

**Domain testing:** It requires three or four tests to be derived for a relational expression. For a relational expression of the form

**E1 <relational operator> E2**

Three tests are required to make the value of E1 greater than, equal to, or less than that of E2.

**Boolean Expression testing:** A condition without relational expression with n variables, all of  $2^n$  possible tests are required provided  $n > 0$ . This strategy can detect Boolean operator, variable, and parenthesis errors, but it is practical only if n is small.

**Loop testing**

Loops are cornerstone for the vast majority of all algorithms implemented in software. Loop testing focuses exclusively on the validity of loop constructs. Testing approach for different types of loops is listed below.

**Simple loops:** The following set of tests will be applied to simple loops, where  $n$  is the maximum number of allowable passes through the loop.

- Skip the loop entirely
- Only one pass through the loop
- Two passes through the loop
- $m$  passes through the loop where  $m < n$
- $n - 1, n, n + 1$  passes through the loop.

**Nested loops:** Simple loop testing strategy if applied to the nested loops will increase the no. of possible tests geometrically. Approach for nested loops are:

- Start at the innermost loop. Set all other loops to minimum values
- Conduct simple loop tests for the innermost loop while holding the outer loops at their minimum iteration parameter values. Add other tests for out-of-range or excluded values.
- Work outward, conducting tests for the next loop, but keep all other outer loops at minimum values and other nested loops to typical values.
- Continue till all loops have been tested.

**Concatenated loops:** When concatenated loops are independent then approach for simple loop was applied & if they happen to be dependent the nested loop approach was applied.

**Unstructured loops:** This class of loops was redesigned to reflect the use of the structured programming constructs.

### **Black Box Strategies**

Black Box testing strategy focuses on the functional requirements of the software without any regard to the internal structure. It is used in most system level testing where functional, performance, recovery and security & stress are main issues. This applied strategy finds errors in incorrect or missing functions (compared to white box), interface errors, errors in external data structures and in behavior performance problems (Combinations of input make it behave poorly). It also finds errors in initialization and termination (Sensitive to certain inputs (e.g. performance). It is performed much later in process than white box...

A test strategy that uses every possible input condition as a test case would be ideal but is not possible. We apply following Black-Box Testing Strategies.

### **Graph Based Testing**

The first step in black box testing is to understand the objects that are modeled in software and the relationships that connect the objects. Once this has been



accomplished, the next step is to define a series of tests that verify all objects that have the expected relationship with another.

## Equivalence Partitioning

It is a black-box testing method that divides the input domain of a program into classes of data from which test cases can be derived. It reduces, by more than one, the number of test cases that must be developed since one test case might uncover a class of errors. It covers a large set of other possible test cases hence helps reduce the number of inputs. Incorrect processing of all integer number is detected.

**Identifying Equivalence Classes:** Each input condition (a sentence or phrase in the specification) is taken & divided into 2 or more classes as valid equivalence class and invalid equivalence class.

### **Rules for Equivalence Classes:**

**Range** - If an input condition specifies a range (i.e. n is an integer from 1 to 10) then

- i) 1 valid (if  $1 \leq n$  and  $\leq 1000$ )
- ii) 2 invalid (if  $n < 1$  and  $> 1000$ )

**Specified Value** - An input condition specifies a specific value e.g. 10 character string then

- i) 1 valid (10 character string)
- ii) 2 invalid (any character string less than 10)

**Value Set** - If the input specifies a set of valid values, then

- i) 1 valid condition within the set.
- ii) 1 invalid condition outside the set.

**Boolean** - If an input condition specifies a “must be” situation (e.g. either Yes or No) then

- i) 1 valid (if Yes).
- ii) 1 invalid (else No).

### **Boundary Value Analysis**

It is a test case design technique that complements equivalence partitioning, by selecting test cases at the “edges” of the class. Greater no. Of errors tends to occur at the boundaries of the input domain rather than in the “Center”. Rather than *any* element in class, BVA selects tests at the *edge* of the class. In addition to input condition, test cases can be derived for output conditions. Similar to Equivalence partitioning, BVA, equivalence classes are identified first then boundaries.

### **Some guidelines for BVA:**

1. If an input condition specifies a range bounded by values a and b then test cases will be designed with values a and b and just above and just below a and b.

2. If an input condition specifies a number of values, test cases will be developed that exercise the minimum and maximum numbers. Values just above and just below minimum and maximum are also tested.

These 2 guidelines are also applied to the output conditions. Designing test case to exercise the data structure at its boundary will test boundaries prescribed on internal program data structures.

## **UNIT TESTING:**

It is based on the test plan for various screens and their outputs.

<b>S.No.</b>	<b>Test Condition</b>	<b>Expected Result</b>	<b>Remarks</b>
<b>1.</b>	Enter a username with a blank space.	Should prompt for not permitting blank space and bring focus on that field.	Done
<b>2.</b>	Enter an invalid password	Should prompt for invalid password after verifying from database and refocus that field.	Done
<b>3.</b>	Press Login without entering the username or password.	Should prompt for username or password.	Done

## **SYSTEM TESTING:**

**It is based on the system test plan for concurrency and performance issues.**

<b>S.No.</b>	<b>Test Condition</b>	<b>Expected Result</b>	<b>Remarks</b>
<b>1.</b>	Log on to the system using same username and password from two different machines.	Should give message that somebody is already logged in.	Done
<b>2.</b>	Go to Daily Report section without logging in.	Should redirect you back to the login page.	Done
<b>3.</b>	Log on to the system using different user-id's from different machines and try editing the same inquiry.	Should allow but do sequentially.	Done
<b>4.</b>	Log on to the system using different user-id and password from different machines & do same inquiry.	Should take minimum time for processing the same inquiry.	Done

## **IMPLEMENTATION**

Implementation is undertaken once the design phase is complete. In this phase, every module identified and specified in the design document is independently coded and unit tested. Unit testing is testing of different units or modules of a system in isolation.

The objective of the implementation phase is

- To define the organization of the code, in terms of implementation subsystems organized in layers,
- To implement classes and objects in terms of components (source files, binaries, executables, and others),
- To test the developed components as units, and
- To integrate the results produced by individual implementers (or teams), into an executable system.

The Implementation phase limits its scope to how individual classes are to be unit tested. System test and integration test are described in the Testing phase

### **Coding Guidelines**

This Section defines the coding guidelines to be adopted during development of IMAR tool. Coding guidelines are essential for any software the development, maintenance and enhancements of which includes multiple programmers. In addition to providing for better team development, a standard also allows for the development of software tools to aid in the creation, testing, and documentation of code.

## **Need of Code Conventions**

Code conventions are important to programmers for a number of reasons:

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its entire life span by the original author.
- Code conventions improve the readability of the software, allowing engineers to understand new codes more quickly and thoroughly.
- By following coding conventions the development time is reduced tremendously.
- It gradually develops a discipline among S/W Engineers. And this discipline amounts to minimum number of bugs in the S/W you write.

## **Python Source Files**

Each Python source file contains a single public or default class or interface. Interfaces associated with a class must be in different source file. The public or default class should be the first class in the file.

Python source files have the following ordering:

- 1) Package and Import statements.
- 2) Beginning comments.
- 3) Class/ interface implementation comment (## ... ##)
- 4) Specific Comments (## for both single and multiple line comments);
- 5) Class and interface declarations
- 6) Class/ interface statement
- 7) Static variables (in sequence)

## 8) Constructors

## 9) Method Comments

### **Beginning comments**

All source files should begin with a PythonDoc-style comment that lists the following things:

Project name:

Module name:

Program name:

Program description:

Author:

Date of Creation:

Modified Date:

Modified By:

### **Package and Import Statements**

The first non-comment line of most python source files is a package statement.

After that, import statements can follow. For example:

Import tkinter

Class/ interface implementation comment (##...##)

Class name:

Class description:

Author name @ author:

Date of creation:



## Methods

### *Comments*

Method name:

Method description:

Parameter @param:

Returns @return:

Exception @exception:

Author Name @ author:

Date @date:

## Method Definition

Given below sequence is to be followed when defining methods:

private methods

protected methods

public methods

***Note: All variables/ objects of same type should be clubbed together i.e. all int should be at one place and strings type in one place.***

## Naming Convention

Naming conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier-for example, whether it's a constant, package, or class-which can be helpful in understanding the code.

➤ Naming Convention Will Be Uniform for the whole project:

- The PYTHON CLASS names Must Be Self Explanatory.
- First letter in lowercase, with the first letter of each internal word in uppercase.
- Member Variables must start with m\_ e.g. m\_myChar
- For static variables the First Letter must be 's\_'
- The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("\_").

**Coding Convention:**

This Coding convention is uniform through out the project and it is mandatory to follow these steps for all Classes we write:

These are the recommendations for code readability.

- The term *compute* can be used in methods where something is computed.
- The term *find* can be used in methods where something is looked up.
- The terms *get/set* must be used where an attribute is accessed directly.
- The term *initialise* can be used where an object or a concept is established.
- *List* suffix should be used on names representing a list of objects.
- n prefix should be used for variables representing a number of objects.
- *No* suffix should be used for variables representing an entity number.
- Iterator variables can be called *i, j, k* etc.
- *is* prefix should be used for Boolean variables and methods.
- Complement names must be used for complement entities.

*get/set, add/remove, create/destroy, start/stop, insert/delete, increment/decrement, old/new, begin/end, first/last, up/down, min/max, next/previous, old/new, open/close, show/hide*

- Abbreviations in names should be avoided.
- Negated Boolean variable names must be avoided
- Associated constants (final variables) should be prefixed by a common type name.
- Exception classes should be suffixed with *Exception*.
- Default interface implementations can be prefixed by *Default*.

## **Files**

- Classes should be declared in individual files with the file name matching the class name.
- File content must be kept within 80 columns.
- Special characters like TAB and page break must be avoided.
- The incompleteness of split lines must be made obvious.
- Split lines occurs when a statement exceed the 80 column limit given above. It is difficult to give rigid rules for how lines should be split. In general:
  - Break after a comma.
  - Break after an operator.
  - Align the new line with the beginning of the expression on the previous line.
- Variables should be initialised where they are declared and they should be declared in the smallest scope possible.

- Variables must never have dual meaning.
- Class variables should never be declared public.
- Related variables of the same type can be declared in a common statement.
- Unrelated variables should not be declared in the same statement.
- Variables should be kept alive for as short a time as possible.

**Comments**

- Tricky code should not be commented but rewritten.
- Making the code self-documenting by names and an explicit logical structure should in general minimize the use of comments.
- Use `//` for all comments, also multi line comments for description only.
- Comments should be indented relative to their position in the code.
- The declaration of *collection* variables should be followed by a comment stating the common type of the elements of the collection

**Code Statistics**

Statistic	Value
Number of .py files	38
Number of Classes	10
Lines of Code	11659
Non Commented Lines of Code	7610
Number of statements	5669
Reusable Lines of Code	

**Unit Testing**

Unit testing (or module testing) is the testing of different units or modules of a system in isolation.

**Verification and Validation**

Verification is the process of determining whether one phase of a software product conforms to its previous phase, whereas validation is the process of determining whether a fully developed system conforms to its requirements specification. Thus while verification is concerned with phase containment of errors, the aim of validation is to make the final product error free.

**Conclusion**

System is implemented as per guidelines and handed over to the testing team for Integration and System testing.

The Virtual Kirana Store (VKS) application enables vendors to set up online shops, customers to browse through the shops, and a system administrator to approve and reject requests for new shops and maintain lists of shop categories. Also, the developer is designing an online shopping site to manage the items in the shop and also help customers to purchase them online without visiting the shop physically. The online shopping system will use the internet as the sole method for selling goods to its consumers.

# **SYSTEM SECURITY**

## **Introduction**

Software security applies information security principles to software development. Information security is commonly defined as "the protection of information systems against unauthorized access to or modification of information, whether in storage, processing or transit, and against the denial of service to authorized users of the provision of service to unauthorized users, including those measures necessary to detect, document, and counter such threats." Many questions regarding security are related to the software life cycle itself. In particular, the security of code and software processes must be considered during the design and development phase. In addition, security must be preserved during operation and maintenance to ensure the integrity of a piece of software.

The mass of security functionality employed by today's networked world, might deceive us into believing that our jobs as secure system designers are already done. However, computers and networks are incredibly insecure. The lack of security stems from two fundamental problems. Systems, which are theoretically secure, may not be secure in practice. Furthermore, systems are increasingly complex; complexity provides more opportunities for attacks. It is much easier to prove that a system is insecure than to demonstrate that one is secure -- to prove insecurity, one simply exploits certain system vulnerability. On the other hand, proving a system secure requires demonstrating that all possible exploits can be defended against (a very daunting, if not impossible, task).

There is presently no single solution for secure software engineering. However, there are specific approaches, which improve the likelihood that a system is secure. In particular, we can improve software reliability. We can also improve our understanding of what must be required to trust a given piece of software.

### **Good Practice**

Security requires more managing and mitigating risk than it does technology. When developing software one must first determine the risks of a particular application. For example, today's typical web site may be subject to a variety of risks, ranging from defacement, to distributed denial of service (DDoS, described in detail later) attacks, to transactions with the wrong party.

Once the risks are identified, identifying appropriate security measures becomes tractable. In particular, when defining requirements, it is important to consider how the application will be used, who will be using the application, etc. With that knowledge, one can decide whether or not to support complex features like auditing, accounting, no repudiation, etc.

Another potentially important issue is how to support naming. The rise of distributed systems has made naming increasingly important. Naming is typically handled by rendezvous: a principal exporting a name advertises it somewhere, and someone wishing to use that name searches for it (phone books and directories are examples). For example, in a system such as a resource discovery system, both the resources and the individuals using those resources must be named. Often there are trade-offs with respect to naming: while naming can provide a level of indirection, it also can create additional problems if the names



are not stable. Names can allow principals to play different roles in a particular system that can also be useful.

### **Security in EVS**

EVS is fully secured system. Each user has a unique user-id and password. Password is stored in encrypted form. Further users are categorised into three classes:

#### **Administrator**

Administrator user has all the rights. He/she has full access to the system. The main rights of the system administrator are:

- Login
- Manage Users
- Manage Stock
- Manage Orders
- Change Delivery Status

#### **User**

User has following rights:

- Registration
- Login
- View products details
- Choose cart to check items
- Make check out as COD
-

**User****Maintenance**

Software maintenance traditionally denotes the process of modifying a software product after it has been delivered to the customer. Maintenance is inevitable for almost any kind of product. Most products need maintenance due to wear and tear caused by use. On the other hand, software products do not need maintenance on this count, but need maintenance on account of the following three main reasons:

**Corrective Maintenance**

Corrective maintenance of a software product may be necessary either to rectify some bugs observed while the system is in user or to enhance the performance of system.

**Adaptive Maintenance**

A software product might need maintenance to support the new features that the users want or to change different functionalities of the systems, or when they need the product to interface with new hardware or software.

**Perfective Maintenance**

A software product needs maintenance to support the new features that the users want or to change different functionalities of the system according to customer demands.

**EVS Maintenance**

EVS is developed in Python, which is a interpreted language. There is need of adaptive maintenance. EVS needs Corrective and perfective maintenance, because it is a new system and has very high scope of extension.

**Conclusion**

EVS is a secured system, as all measures are taken into account so that unauthorised person could not access the system.

## **GANTT CHART:-**

A Gantt chart is a type of bar chart that illustrates a project schedule.<sup>[5]</sup> This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements constitute the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line.

Gantt charts are sometimes equated with bar charts.

Gantt charts are usually created initially using an early start time approach, where each task is scheduled to start immediately when its prerequisites are complete. This method maximizes the float time available for all tasks.

The Gantt chart uses horizontal bars to show the durations of actions or tasks. The left end marks the beginning of the task while the right end its finish. Earlier tasks appear in the upper left and later ones in the lower right.

## **GANTT CHART OF VIRTUAL KIRANA STORE**

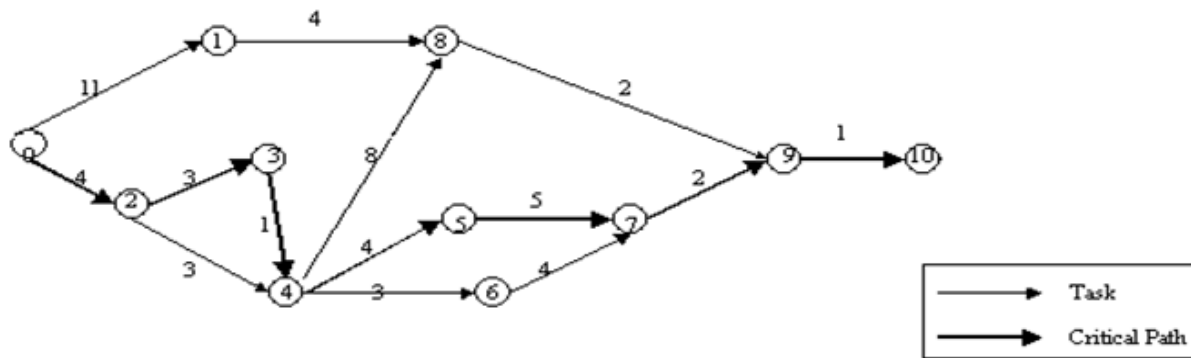
Weeks Tasks	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
Coding									
System Design									
Requirement Analysis									
Testing & Debugging									
Installation									
User Training									
Application Development									

## **PERT CHART:-**

The PERT chart makes use of tasks. It shows achievements. These achievements however are not task achievements. They are terminal achievements, called events. Arrows are used to represent tasks and circles represent the beginning or completion of a task. The PERT chart uses these paths and events to show the interrelationships of project activities. In this project there are twelve activities/tasks.

1. Need Identification
2. Complete Project Proposal
3. Feasibility Study
4. Software Requirement Analysis
5. System Design
6. Coding
7. Test Run (Lab testing)
8. Modifications (if any)
9. Final Launch
10. Documentation

### PERT CHART



The list of tasks and events is networked in a PERT chart in above figure. The arrow length is not significant, but the sequence and interconnections must give a true picture of the precedence of activities to be completed. The numbers on the activity lines are the time (in weeks) required between events.

**Note: No. in circle denotes events serial no., while those with the arrow denote taken in weeks.**

PERT chart is valuable when a project is being planned. When the network is finished, the next step is to determine the *critical path*. According to above diagram the critical path is:

Critical Path = Start # – 0 – 2 – 3 – 4 – 5 – 7 – 9 – 10 – 11 – 12

Duration (in weeks)       $4 + 4 + 1 + 4 + 5 + 2 + 1 + 2 + 1 = 24$  weeks

The critical path is the longest path through the network. No task on the critical path can be held up without delaying the start of the next tasks and, ultimately, the completion of the project. So the critical path determines the project completion date.

## **LIMITATION OF THE PROJECT**

- The system requires persistent data.
- The system can only be deployed on server operating system.
- An IBM xSeries or equivalent or higher configuration server is required.
- The system uses HTTP 1.1 protocols.
- IE 6.0 or compatible browser is required to access the system.
- A minimum of 1024 MB RAM is required for running an Application Server along with Oracle or MS SQL Server 2000 or IBM DB2 or MySQL.



## **FUTURE SCOPE/FURTHER ENHANCEMENT**

The future scope of the application is very vast. It can be web based. The project may include the graphical tools, charts graph etc. It gives a visual effect and one can easily find out one of things with just one view easily. This application is very useful for any organization, which can be access on the internet. All the project details are stored in the database and displays to the manager very easily, by which he or she can manage his task and team members very easily.

These requirements will be made available

- This project will Provide Online payment facility.
- This project will help them to track the delivery status.
- This project will help seller to create their own seller profile
- This software will get proper updates related to the UI and optimized patches.

## **BIBLIOGRAPHY**

- **Head First Python: A Brain-Friendly Guide:** Paul Barry
- **Django :**Youtube CODE WITH HARRY & TELESKU
- **Microsoft visual studio 2021** Unleashed Publication (Third Edition)
- **Head First HTML with CSS & XHTML:** Elisabeth Robson
- **SQL Server 2012, Database Design Study Guide:** Kevin Hough
- **IGNOU STUDY MATERIAL**
- Some websites:
  - a. [https://www.youtube.com/channel/UCeVMnSShP\\_Iviwkknt83cw](https://www.youtube.com/channel/UCeVMnSShP_Iviwkknt83cw)
  - b. <https://www.w3schools.com/>
  - c. <https://egyankosh.ac.in/>
  - d. <https://www.udemy.com/>