Atma Ram Sanatan Dharma College

University of Delhi

Software Engineering Project

Paper Code **32341402**

Submitted By:

Aditya Sethi	Chandni Kumari	Parleen Ranhotra	Kumar Sanskar
20/88080	20/88023	20/88054	20/88039

BSc (Hons)Computer Science, Sem -IV

Submitted To

Ms. Uma Ojha

Department of Computer Science

ACKNOWLEDGMENT

The contentment that is achieved on the successful completion of any task is incomplete without mentioning the names of the people who made it possible with their consistent guidance, support and indispensable encouragement.

The Project was jointly undertaken by CHANDNI, SANSKAR, PARLEEN and ADITYA as their fourth semester Software Engineering Project, under the proper guidance and supervision of “Mrs. Uma Ojha”. Our primary thanks to her, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project.

It’s our privilege to acknowledge our deepest sense of gratitude to her for her inspiration which has helped us immensely. We are extremely grateful to her for unstilted support and encouragement in the preparation of this project.

INDEX

PROBLEM STATEMENT

Grocery shopping is a routine buying behaviour, as it not only involves regular decision making but also includes consumer behaviour which is automatic, habitual and unthinking. Online grocers face a number of challenges. The major challenge is lack of handy experience in consumer demands. Online market has developed its space in the virtual world but is this market worth it for all kinds of products? Thus, there is need to study consumer perception towards the online grocery market.

The purpose of this idea is the fact that people's routine is becoming hectic day by day because of which they do not have time to go to different shops physically or visit different websites virtually. SHOP ON provides customers with the list of items along with their price that are available in shops near by them. The system basically establishes a method through which they can survey the price and details of the items they need in their nearby shops.

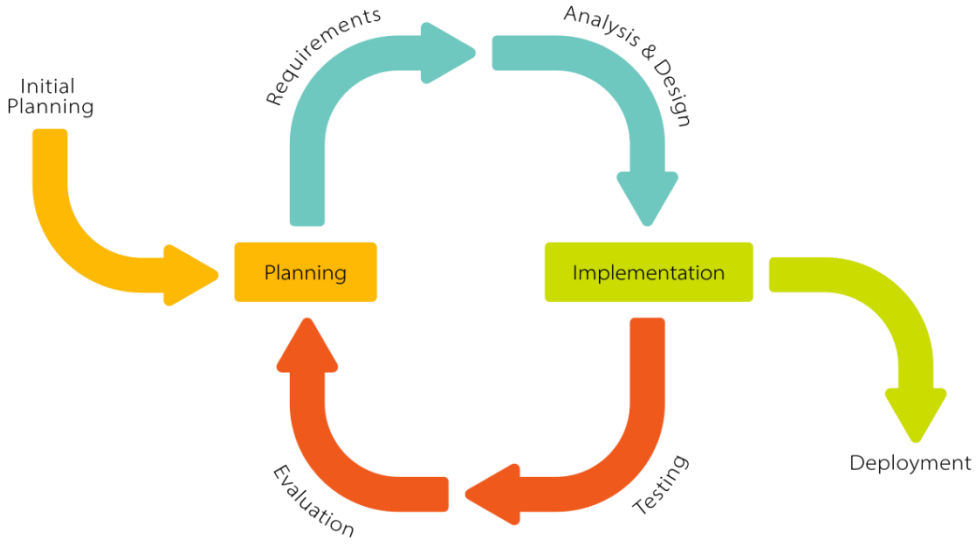
The system of the App would not only benefit the customers but would also benefit the shopkeepers as it would increase their sales. Consumers will get advantage of place, time and product. On the other hand, retailers will get advantage of all time connectivity with their consumers.

It also consists of a Shopkeeper’s login panel where shopkeepers can login to the system and then see the customer requests for items. The system then schedules those requests and serves them one after another. At the end of purchase, the system will email the user an online receipt in a printable word format so that the user can directly print it for their record purpose.

PROCESS MODEL

For our project shopon, we have decided to use the iterative method. This will help us review and improve the rough individual parts of the software in the next iterations. We will be able to provide enough time in planning and evaluation and can even help us make unforeseen changes/updates throughout the software lifecycle.

The Iterative Model relies on the whole product being developed step-by-step (Design/Develop, Test, Implement). Thus this will not only help us identify problems at early stages but also be more cost-effective than the prototype model. It also helps in ensuring that the final product built iteratively, is according to the standards required by the user.



This model gives an opportunity to identify and rectify any major design or planning flaws in the process model because of its cyclic nature. Furthermore, the iterative model is really beneficial as it can accommodate changes in the initially specified requirements for the system.

The Iterative Model allows accessing earlier phases, in which the variations are made respectively. The final output of the project was renewed at the end of the Software Development Life Cycle (SDLC) process.

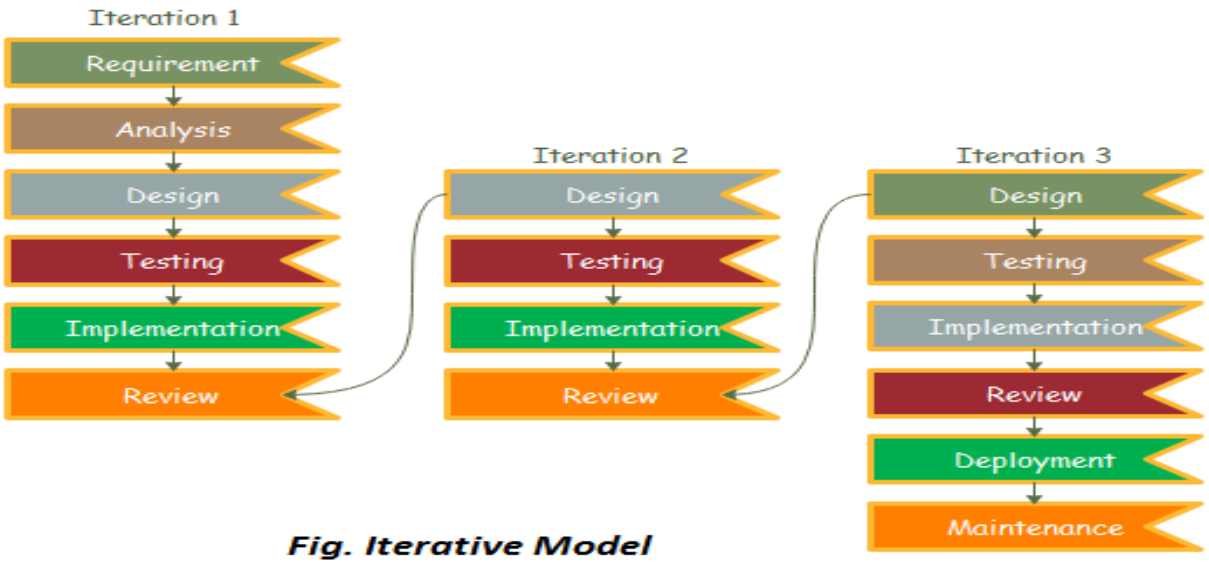


Fig. Iterative Model

1. Requirement gathering & analysis: In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfil or not. Analyst checks that need will be achieved within budget or not. After all of this, the software team skips to the next phase.

2. Design: In the design phase, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

Implementation: In the implementation, requirements are written in the coding language and transformed into computer programmes which are called Software.

4. Testing: After completing the coding phase, software testing starts using different test methods. There are many test methods, but the most common are white box, black box, and grey box test methods.

5. Deployment: After completing all the phases, software is deployed to its work environment.

6. Review: In this phase, after the product deployment, review phase is performed to check the behaviour and validity of the developed product. And if there are any error found then the process starts again from the requirement gathering.

7. Maintenance: In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging and new addition options.

Advantages of Iterative Model

- It is easily adaptable to the ever-changing needs of the project as well as the client.
- It is best suited for agile organisations.
- It is more cost effective to change the scope or requirements in the Iterative model.
- Parallel development can be planned.
- Testing and debugging during smaller iterations is easy.
- Risks are identified and resolved during iteration; and each iteration is easily managed.
- In an iterative model less time is spent on documenting and more time is given for designing.

Disadvantages of Iterative Model

- No fixed budget or deadlines.
- Strong customer involvement in the process
- More resource-intensive than the waterfall model.
- Risk analysis requires highly qualified specialists to check the risks in our system.
- The whole process is difficult to manage.

Why did we use the Iterative Model?

- We have used this model because the requirements of the product are not clearly understood or are unstable.
- The requirements may change quickly after every iteration.
- There can be confusing or difficult functions that can be identified only after reviewing the software again.
- Missing functionalities or removal of any functionality may be required.
- We can look for any flaws in further iterations.

Software and hardware requirements

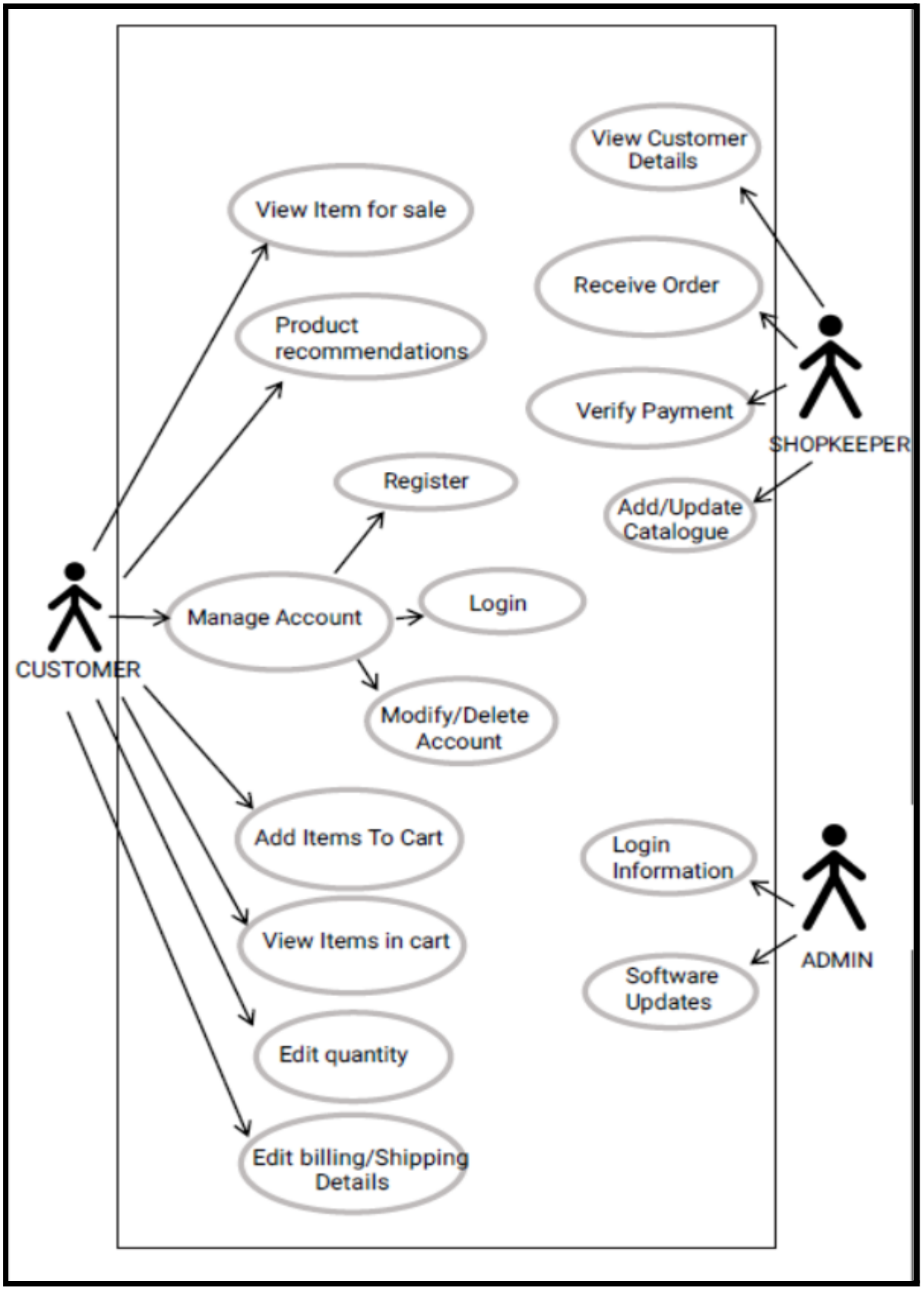
Hardware requirements to run the application:

- A mobile phone with memory space available upto 50 mb

Software used while developing application:

- OS: Windows 10
- Platform: Android Studio
- Programming language: Php
- User Interface: XML

USE CASE DIAGRAM



USE CASE SCENARIO

1. REGISTRATION:

Brief Description: The case describes that the user or any of the actors can register for the system.

Actors: The following actors take part or interact with the system:

- ADMIN
- SHOPKEEPER
- CUSTOMER

- **Flow of Events:**
  - **Basic Flow:**
    - Any actor registering the system will be asked for username, E-mail ID, password, role and other required data with respect to their specification.
    - The data is then stored into a database.
    - Then it returns a value based on successful registration otherwise an error.
  - **Alternative Flow:** In case of unsuccessful registration, the user is prompted an error and asked to enter relevant and correct information. Since no entry was made the database remains unchanged.

Special Requirements: None

Pre-conditions: None

Post-conditions: Upon successful completion of the use case the actor is registered into the system and the data is stored in the database permanently.

2. LOGIN:

Brief Description: The case describes how any of the registered actors can login to the system.

Actors: The following actors take part or interact with the system:

- ADMIN
- SHOPKEEPER
- CUSTOMER

- **Flow of Events:**
  - **Basic Flow:**
    - The actor is prompted to login with their credentials.
    - Then the data is verified in the database for valid users.
    - If the actor is registered then they are granted access.
  - **Alternative Flow:** In case of unsuccessful login, the user is prompted an error and asked to enter relevant details or is redirected to get registered.

Special Requirements: None

Pre-conditions: The actor must be registered with the system.

Post-conditions: If the user case is executed successfully, the actor is granted access and redirected to a relevant page.

3. MANAGE ACCOUNT:

Brief Description: The case describes that the user or any of the actors can edit or delete their profile from the system.

Actors: The following actors take part or interact with the system:

- ADMIN
- SHOPKEEPER
- CUSTOMER

- **Flow of Events:**
  - **Basic Flow:**
    - The actor can edit the details such as address, phone number, E-mail. In this case data already stored in the database will get replaced by the new data.
    - The actor can also delete its profile. In this case the data stored in the database will be deleted permanently.
  - **Alternative Flow:** In case of unsuccessful edits, the user is prompted an error and asked to enter correct information. Since no entry was made, the database remains unchanged.

Special Requirements: None

Pre-conditions: The actor must be logged into the system.

Post-conditions: Upon successful completion of the use case the actor has edited or deleted their profile into the system and the data is stored in the database permanently.

4. MAINTAIN PRODUCT INFORMATION:

Brief Description: This case allows the actor to maintain product information. This includes adding, changing, deleting, updating an item's information from the system.

Actors: The following actors take part or interact with the system:

- SHOPKEEPER

- **Flow of Events:**
  - **Basic Flow:** This use case starts when the actor wished to add, update or delete an item from the system
    - The system asks the actor to specify the function that they'd like to perform.
    - The actor then fills in the required information.
    - The system then executes one of the sun-flows based on what the actor chose.
      i. If choses "Add item", then adds an item sub-flow is executed.
      ii. If choses "Delete item", then delete an item sub-flow is executed.
      iii. If choses "Update item", the update an item sub-flow is executed.

    a. **Add an item:** The system asks the actor to enter the product information which includes name of the product, brand name, price of product etc. Then the data is stored in the database.

    b. **Update an item:** The system asks the actor to enter item unique code and then item details are retrieved and displayed. Actor

then makes desired changes. Then the system updates item records in the database.

c. **Delete an item:** The system asks the actor to enter item unique code then item details are retrieved and displayed. Then the system asks the actor to confirm deletion. Then the system deletes the record from the database.

- **Alternative Flow:**
  - **Insertion Failed:** If in adding an item in add an item sub-flow, if the insertion failed, the system shows the error message as enter valid details or item already exists, according the values given.
  - **Item not found:** If in the update an item or delete an item sub-flow, an item unique number does not exist, then the system displays an error message. The actor can either cancel the operation which will result in the use case end or they can enter a different item number.
  - **Update Cancelled:** If the actor decides to not update the information in the Update an item sub-flow, the update is cancelled and they are returned to the beginning of the basic flow.
  - **Delete Cancelled:** If the actor decides to not delete the information in the Delete an item sub-flow, the delete is cancelled and they are returned to the beginning of the basic flow.

Special Requirements: None

Pre-conditions: The actor must be logged into the system before the use case begins.

Post-conditions: Upon successful completion of the use case the actor has updated, added or deleted an item from the system. Otherwise the system is unchanged.

5. BUYING PRODUCTS:

Brief Description: The case describes how the actors can buy products from the system.

Actors: The following actors take part or interact with the system:

- CUSTOMER

Flow of Events:

- **Basic Flow:**
  - The actor can search for an item or product by name or unique code and also the actor can also provide the list.
  - A list of items will be retrieved from the database i.e. all products and the different shops that provide it.
  - Then the actor can choose the desired product and the shop also.

- The actor can make its order and wait for the shopkeeper for acknowledgement and after that can go to pick it up.

- **Alternative Flow:** In case of unsuccessful order either due to some problem or by the rejection of the shopkeeper, the user is prompted the issue and asked to order again and returned to the beginning of the basic flow.

Special Requirements: None

Pre-conditions: The actor must be logged into the system.

Post-conditions: Upon successful completion of the use case the actor has placed its order and will be redirected to the main page.

6. ADMINISTRATION:

Brief Description: The case describes how the actor will use the system.

Actors: The following actors take part or interact with the system:

- ADMIN

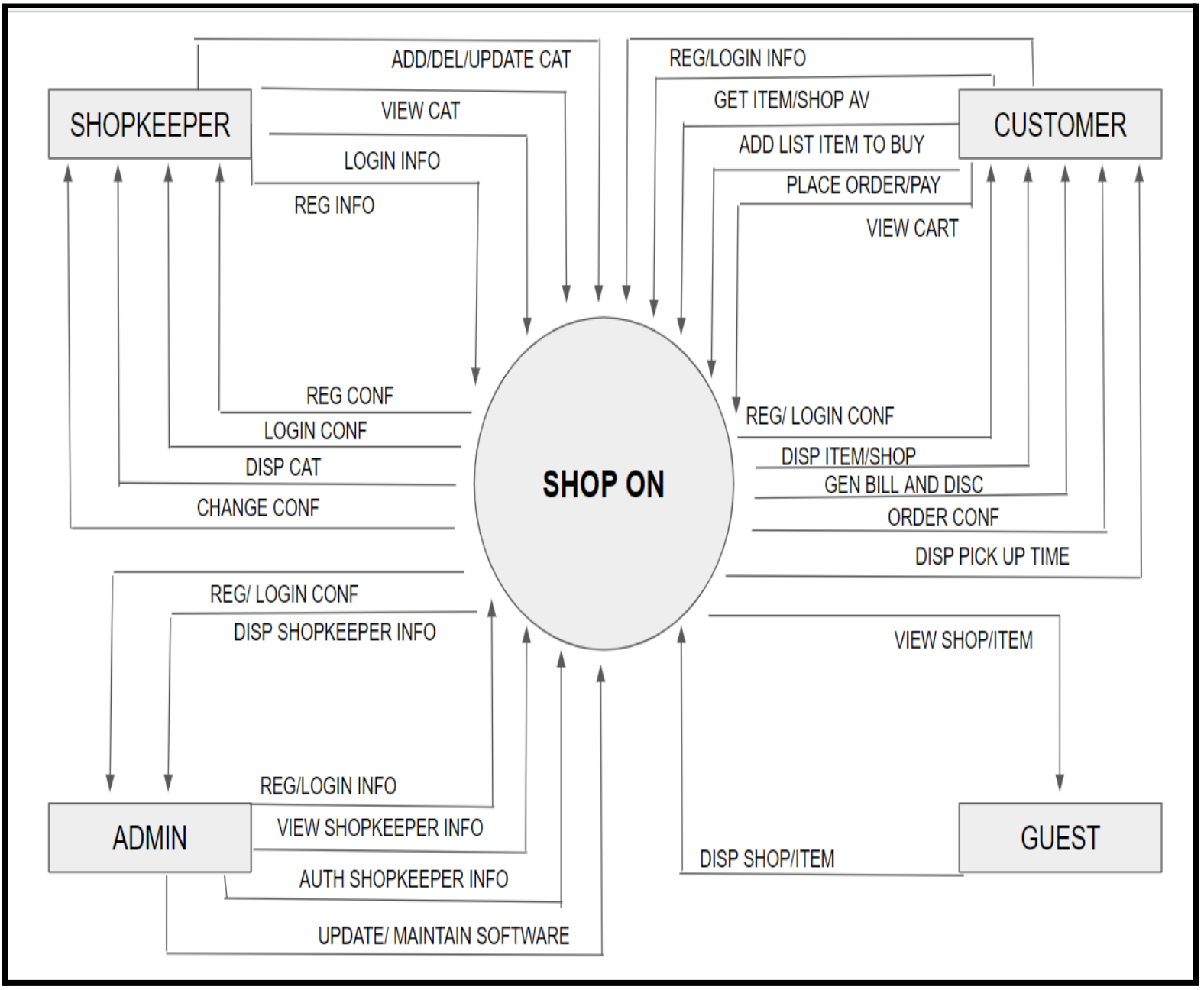
- **Flow of Events:**
  - **Basic Flow:**
    - The actor login into the system same as the user but with special credentials.
    - The actor can be able to view details of the shopkeeper and authenticate them.
    - The actor has the right to update and manage the system.
  - **Alternative Flow:** In case an actor got his credentials incorrect or forgets it then the actor will not be able to access the administration panel. In order to get back credentials, the actor has to approach the officials to generate new credentials.

Special Requirements: None

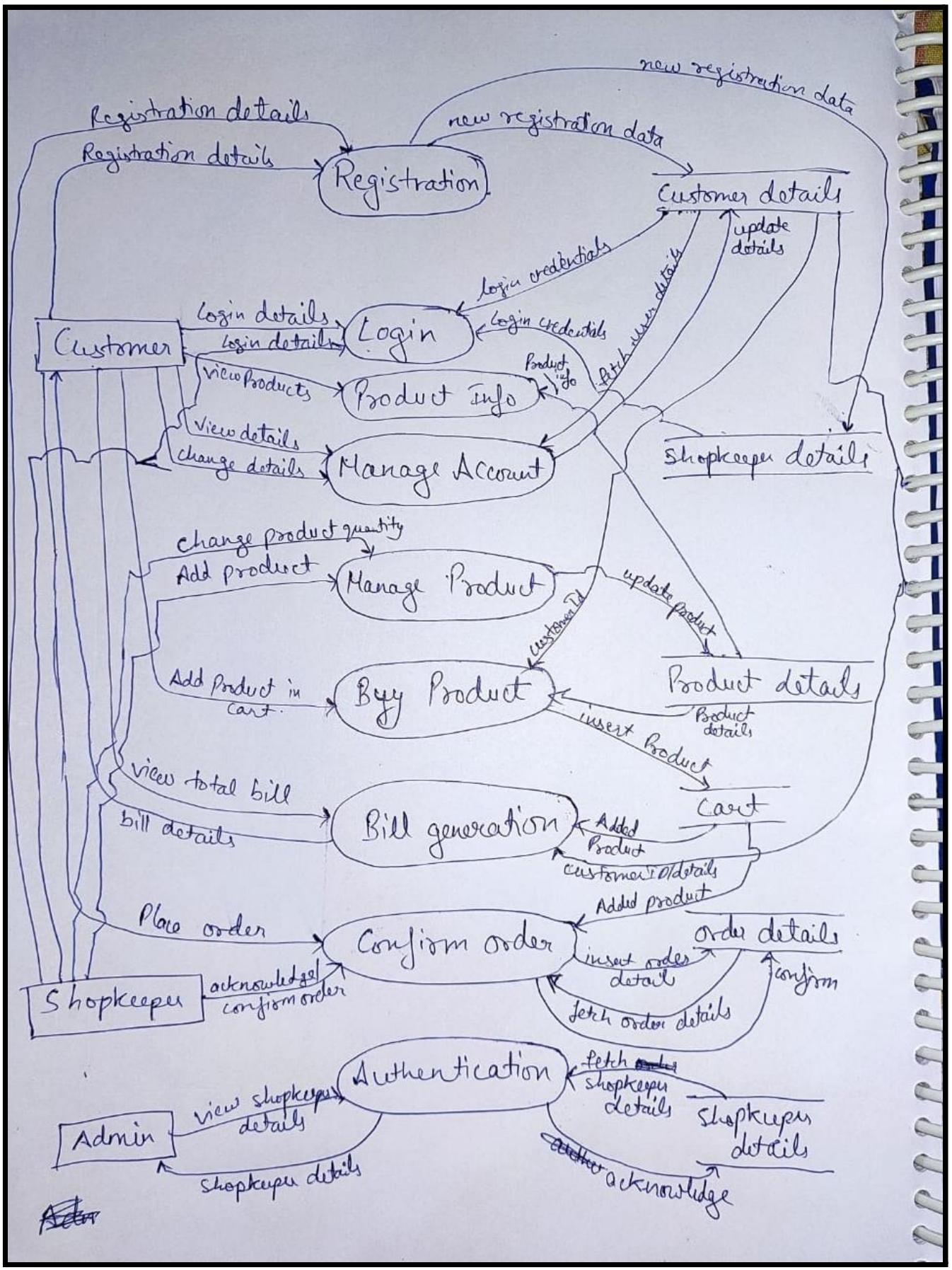
Pre-conditions: The actor must have admin credentials.

Post-conditions: The higher authority officials can approach the software developer team to generate new admin credentials or any other matter.

CONTEXT DIAGRAM



LEVEL 1 DIAGRAM



LEVEL 2 DIAGRAM

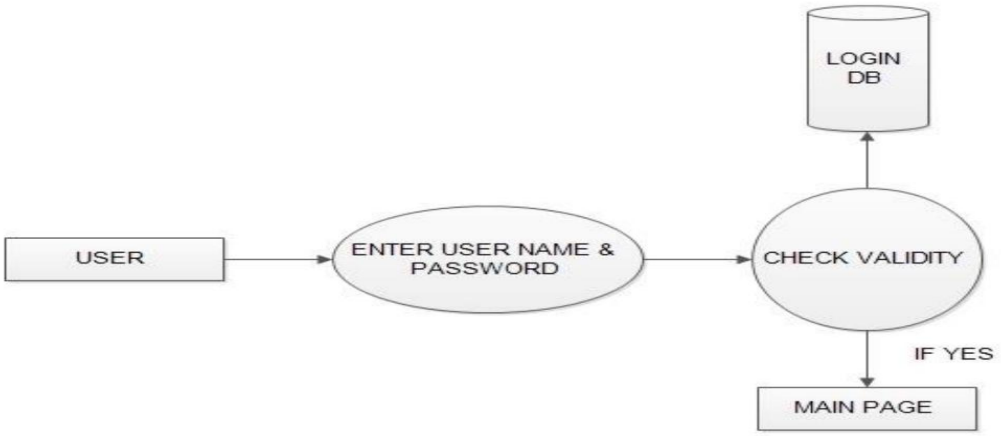


Fig 3.10: Login DFD

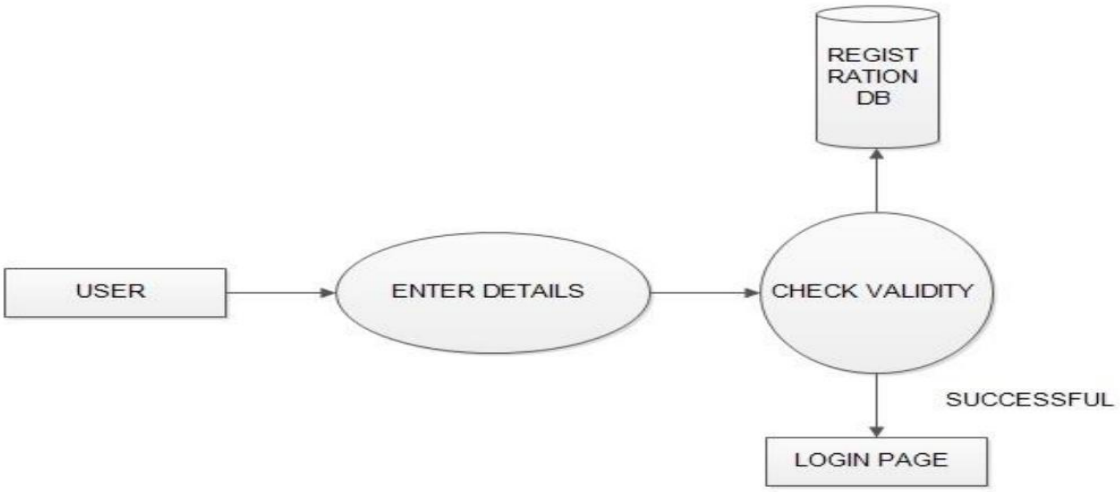


Fig 3.11: Registration DFD

DATA DICTIONARY

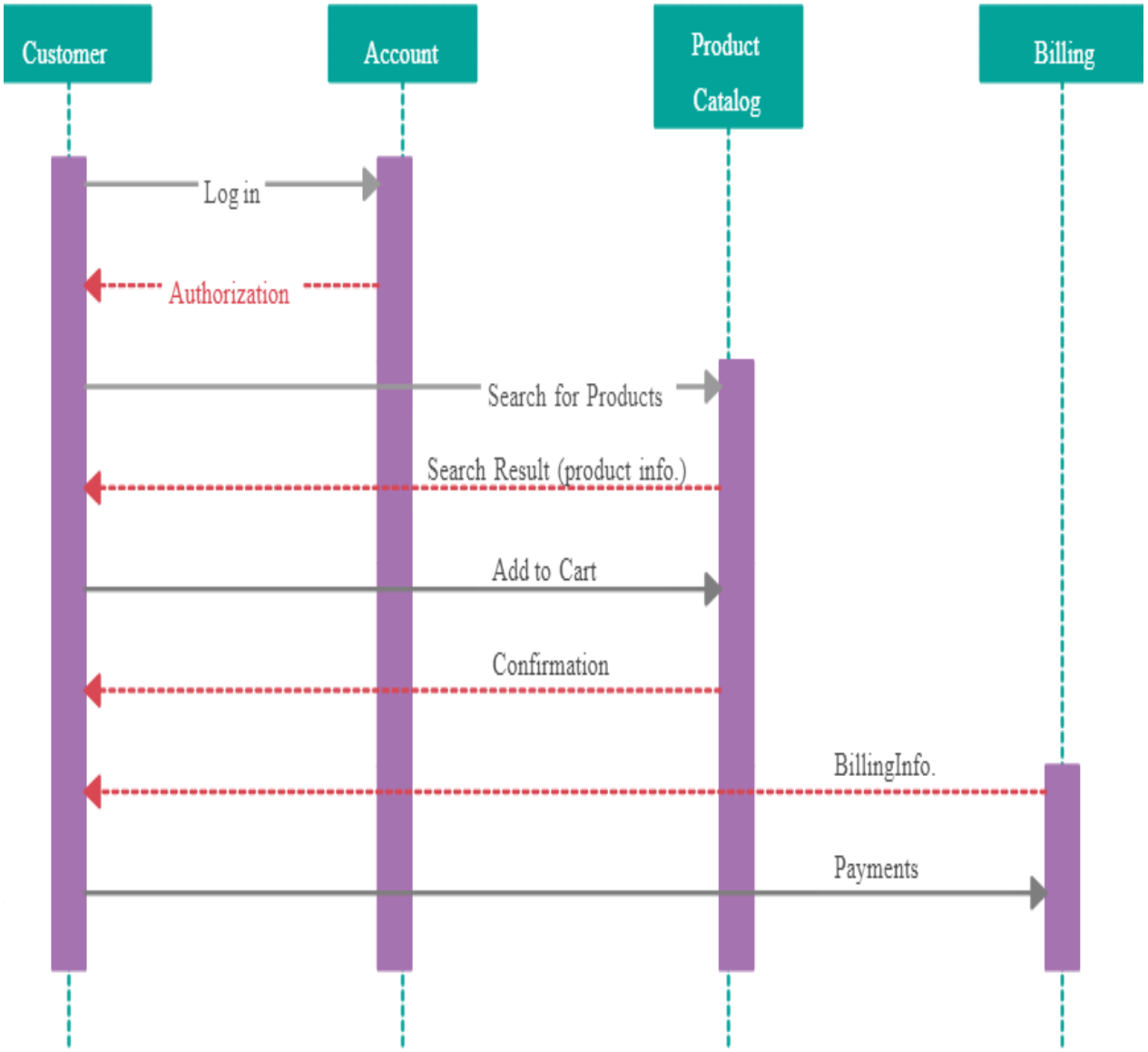
The data dictionary provides an **organised** approach for representing the characteristics of each data object and control item. It has been proposed for describing the content of objects defined during structured analysis. A data dictionary is very **important** in the software development process because:
A data dictionary lists standard terminology for use by an engineer working on a project.
The dictionary provides the analyst with means to determine the definition of different **data structures** in the terms of their component elements.

The Format of Data Dictionary:

- Name the **primary name** of the data or control item, the data store or an external entity.
- Aliases-other names used for first entity.
- **Description** a notion for representing content.
- Type of the data.

DATA	DESCRIPTION
Registration Details	[Email/Mobile Number]
Login Details	Username + Password
User Details	First Name + Last Name + Gender + email id + Mobile Number
Residential Details	Residential Address + [Street/Area/Locality] + [Town/Village] + Post Office + Pin Code
Track Details	Order Status
Enquiry Details	Queries + Suggestions
Location	Latitudes + Longitudes +Time

SEQUENCE DIAGRAM



SOFTWARE REQUIREMENT SPECIFICATION

(SRS) DOCUMENTATION

1. Introduction:

This document aims at defining the overall software requirements for the 'Online Grocery Shopping System'. Efforts are made so that requirements are defined accurately. The final product will have only features mentioned in this document and assumptions for any additional feature should not be made by any of the parties involved in developing/testing/implementing/using this product. In case it is required to have additional features, a formal change request will need to be raised and subsequently a new release of this document and/or the product shall be produced.

1.1. Purpose:

This specification document describes the capabilities that will be provided by the software application "ShopOn". It also states the various required constraints by which the system will abide by. The intended audience for this document is the members of the software development team, testing team and the end users of the product.

1.2. Scope:

ShopOn will be an Online Grocery buying system that will be used by the customers and retailers. The system will allow the individuals to register themselves in the system according to their role. Customers will Receive confirmation after placing the order.

1.3. Definitions, Acronyms, and Abbreviations:

The following abbreviations have been used throughout this document:

OTP: One - Time Password

A - Login Details: Admin Login Details

1.4. References:

- Pressman, R. S., & Maxim, B. R. (2015). Software Engineering: A Practitioner's Approach 8th edition, McGraw-Hill.

- Aggarwal, K. K., & Singh, Y. (2007). Software Engineering. 3rd edition.
- IEEE recommended practice for Software Requirement Specifications. ISO/IEC/IEEE 29148:2011

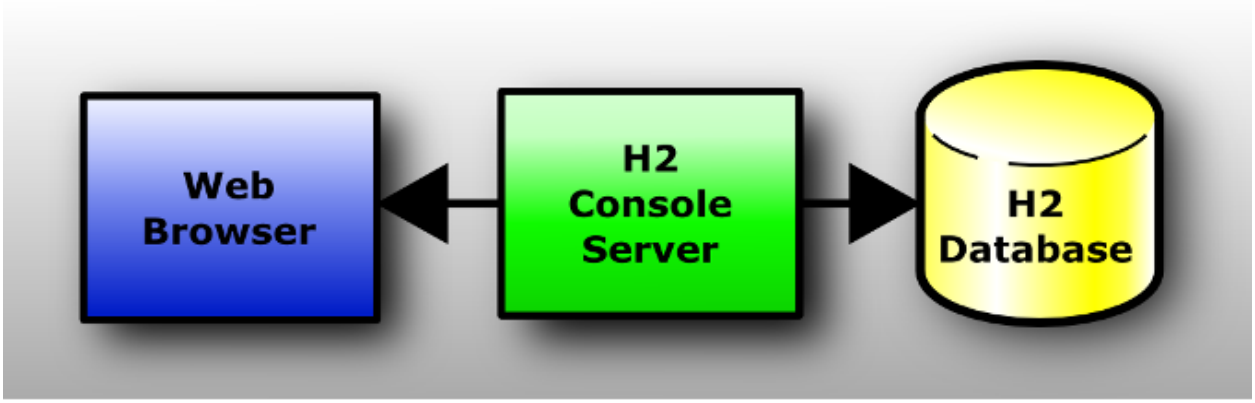
1.5. Overview:

- The rest of this SRS document describes the various system requirements, features, interfaces and functionalities in detail. The second chapter of the SRS deals with and introduces the reader to the system with an overview of the system functionality and system interaction with other systems. The third chapter provides the requirements specification in detailed terms and a description of the different system features.

2.Overall Description:

2.1. Product Perspective:

- The application will be windows-based, it is neither self-contained nor independent.



- **System Interface:**

  None

- **User Interface:**

The application has user-friendly and menu-based interface as follows:

Register Screen: The screen displays the columns where users are asked to enter the relevant details based on whether he/she is a customer or a shopkeeper and get themselves register with the system. The screen demands the user's name (full name plus last name), username, password, phone number, email address, age and Aadhaar card common for both. Other than that shopkeeper's details include GST Number. A key will be generated by the admin for a successful registration with the system.

- **Login Screen:** The users shall use their username and password to login the system and access the feature according to their role.

- **Search Screen:** The customer can search the items they want to buy.

- **Bills/Reciepts :** Receipts will be provided to the customers after they are finished with the payment.

- **Payment Screen:** After the successful check-out, through this screen the customer will pay the sum of amount shown on screen.

- **Administration Screen:** The screen is for the admin to work on the system, it will have the several privileges options that the admin has, through which the admin can change/see the customers' and shopkeepers' detail.

- **Hardware Interface:**

The application must run over the internet, all the hardware shall require to connect internet will be hardware interface for the system. E.g., modem, WAN, LAN, Ethernet.

- ❖ Minimum 2GB RAM.
- ❖ 1 TB hard disk for storage.
- ❖ Pentium IV 1GHz.

- **Software Interface:**
  - ❖ Windows operating system windows 7/8/10.
  - ❖ Web browser e.g. Mozilla Firefox, Opera.
  - ❖ WAMP Server.
  - ❖ MySQL Database Management System.
  - ❖ Dreamweaver and PHP Designer.
  - ❖ Google Maps/GeoLocation API.
  - ❖ Communication Interface:

The system shall use the HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol.

- **Memory Constraints:**

At least 64MB RAM and 2GB space on the hard disk will be required for running the application.

2.2. Product Function:

The system will allow only authorized users with specific roles (Administrator, Customer and Retailer). Depending upon the user's role, she/he will be able to
access only specific modules of the system. A summary of the major functions that the software will perform:

- The system will allow users to register themselves first and then login for further process.
- Customers can search for items using names or symptoms.

- Customers can then add items they have selected to the cart.
- After successful check-out by the customer, the shopkeeper can see the items he/she has to prepare for delivery.
- After all the proper check-outs are done, customers can pay the amount.

2.3. User Characteristics:

To use the system user should have:

- ❖ Technical Expertise: Users should be comfortable using general purpose applications on a computer.
- ❖ Language: The user should at least understand the English language to easily function through the system.
- ❖ Knowledge: The user must have basic knowledge regarding common symptoms.

2.4. Constraints:

- Validate for registered users via email id password.
- Be robust enough so that users do not corrupt it in the event of application.
- Be able to handle multiple users at the same time.

2.5. Assumptions and Dependencies:

None

2.6. Apportioning of Requirements:

None

3. Specific Requirements:

This section contains all the software requirements at a level of detail sufficient to enable designers to design or system to satisfy these requirements, and testers to test that the system satisfies those requirements.

3.1. External Interfaces:

- · **User Interfaces:**

The following screens will be provided:

Login Screen: This screen will be the first screen to be displayed. It will allow customers to access different screens based upon the customer's role. Various fields available on this screen will be:

Customer ID: Alphanumeric of length up to 20 characters.

Password: Alphanumeric.

Role: Will have any one of the following values: Shopkeeper, Customer and Admin.

- **Registration:** This screen will be accessible to a visitor wishing to become a user. One will have to fill the fields:

Username

Password

Confirm Password

- · **Search Info:** This screen is visible to Customer where he/she can search different items according to their need.

The fields on this page would be:

Search items by names

Destination

- **Item Info:** This screen will show the available items.
- **Payment Screen:** This is a payment screen, where a customer will successfully pay the fees for the grocery he had bought.
- **Administration:** This screen is only visible to the ADMIN where he/she can make any changes in the database, maintain all other user's accounts and can generate reports whenever required. This screen has various options for the ADMIN to work.

3.2. Performances Requirements:

It must be able to handle multiple users.

High data transfer rate.

24/7 data connection should be available.

3.3. Logical Database Requirements:

The application communicates with the internal database.

The database is used for registration, login, to generate receipts and to check the available items.

3.4. Design Constraints:

None

3.5. Software System Attributes:

- **Availability:**

  The system should have a good internet connection to receive the notifications timely.

  The system should be backed up timely for easy recovery in any dysfunction scenarios.

- **Security:**

  The application is secured by a username and password.

- ❖ · **Maintainability:**

  The application will be designed in a maintainable manner.

  It will be easy to incorporate new requirements in the individual modules (i.e., search item, customer info, payment screen and administration)

- **Portability:**

  The application is portable.

PRODUCT METRICS

Project metrics and the indicators derived from them are used by the project manager and a software team to adapt project work flow and

technical activities. The intent of project metrics is twofold. First, there metrics are used to minimize the development schedule by making the adjustment necessary to avoid delays and mitigate potential problems and risks. Second, project metrics are used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality.

FUNCTION POINT

❖ External Input(EI) : 6

- Shopkeeper/Customer/Admin Registration.
- Add items in Cart.
- Place Order.
- Arrange Catalogue(Shopkeeper)
- Authenticate shop info
- Update/Maintain Software.

❖ External Output(EO): 6

- Shopkeeper/Customer/Admin Registeration confirm/reject.
- Shopkeeper/Customer/Admin Login confirm/reject.
- View cart
- View catalogue
- View shopkeeper info.
- Generate Bill.

❖ External Inquiries(EQ) : 3

- Shopkeeper/Customer/Admin Login
- Get shops available
- Get items available

❖ Internal Logical Files : 5

- Customer details
- Shopkeeper details
- Product details
- Order details
- Cart

❖ External Interface Files(EIF) : 1

- Admin authenticate shops from Google Maps/ Geolocation API.

▪ Computing Function Points

INFORMATION DOMAIN VALUE	COUNT	WEIGHTING FACTOR			
		SIMPLE	AVERAGE	COMPLEX	
External Input (EI)	6	3	4	6	=24
External Output (EO)	6	4	5	7	=30
External Inquiries (EQ)	3	3	4	6	=9
Internal Logical Files (ILF)	5	7	10	15	=50
External Interface Files (EIF)	1	5	7	10	=7
TOTAL					**=120**

COUNT TOTAL = 120

Value Adjustment Factor (VAF)

Sr. No.	Fi	Rating

1.	Does the system require reliable backup and recovery?	3
2.	Are data communication required?	3
3.	Are there distributed processing function?	2
4.	Is performance critical?	3
5.	Will the system need heavily used configuration?	3
6.	Does the system require online data entry?	5
7.	Does online data entry require the input transaction to be built over multiple screens?	3
8.	Are the ILFs updated online?	5
9.	Are the input, output or inquiries complex?	3
10.	Is the internal processing complex?	3
11.	Is the code designed to be reusable?	4
12.	Are conversion & installation include design?	3
13.	Is the system designed for multiple installations in different organizations?	4
14.	Does the application facilitate end user efficiency?	5
TOTAL		**49**

€ To compute function points (FP), the following relationship is used:

$$FP = count\ total \times [0.65 + 0.01 \times \Sigma(F_i)]$$

= 120 X [0.65 + 0.01 X 49]

= 120 X [0.65 + 0.49]

= 120 X 1.14

= 136.8 ≈ 137

€ Assumptions :

The organizational average productivity for a system of this type is 6.5FP/PM and the labour rate of $8000 per month.

- Cost per FP = $labour \frac{rate(per\ month)}{value\ of\ FP\ per\ PM}$

  = $\frac{8000}{6.5}$ = $1230.76

  ≈ $1231

- Estimated or Total = Cost of 1 FP X Estimated FP

  = $1231 X 137

  = $168,647

- Efforts = $\frac{Estimated\ FP}{(FP\ per\ PM)}$

  = $\frac{137}{6.5}$

  = 21.07

  ≈21 Person Per Month

EFFORTS USING COCOMO MODEL

COCOMO model, constructive cost model is one of the most widely used software estimation model that addresses the following areas:
Application composition model
Early design model
Post architecture state model

Our project is based on an application-based model as this model is used during early stages of software when prototyping of user interface, consideration of software, system interaction, assessment of performance and evaluation of technology maturity is paramount.

Object Type	Complexity Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Components	-	-	10

Complexity Weight

Object point is indirect software count measure i.e. compiled using counts of number of screens, reports and 3GL components. Each object instance is classified into one of the three complexity levels: simple, medium or complex.
The object count is determined by multiplying the total number of object instances by weighing factor. When component-based development or general software re-used is to be applied, the percent of re-use is estimated and object count is adjusted. In our software project, since we're not re-using any of the components, the **percent re-use here is 0**.

Developers experience & capability	Productivity (PROD)
Very Low	4
Low	7
Nominal	13
High	25
High	50

Productivity Rate

To calculate weight complexity for screen we will follow this table:

No. of views contain	Sources of data tables		
	Total < 4 (< 2 servers < 3 clients)	Total < 8 (2 - 3 servers 3-5 clients)	Total 8 + (> 3 servers > 5 clients)
< 3	Simple	Simple	Medium
3 - 7	Simple	Medium	Difficult
> 8	Medium	Difficult	Difficult

For Screens

To calculate weight complexity for report we will follow this table:

No. of section contain	Sources of data tables		
	Total < 4 (< 2 servers < 3 clients)	Total < 8 (2 - 3 servers 3-5 clients)	Total 8 + (> 3 servers > 5 clients)
0 - 1	Simple	Simple	Medium
2 - 3	Simple	Medium	Difficult
4 +	Medium	Difficult	Difficult

For Reports

€ Number of Screens:

1.Registration: It needs 1 data table Registration details and 2 views of screen is enough.

2.Login: It needs 1 data table Registration details and 3 views are there user, shopkeeper and admin.

3.Cart: It needs 1 data table Cart details and 2 views are there user and admin.

4.Search Items: It needs 1 data table Item details and 3 views are there user, shopkeeper and admin.

5.Edit Inventory: It needs 1 data table Inventory details and 2 views are there shopkeeper and admin.

6. Track status: It needs 1 data table Track details and 1 view is enough.

7.Complaint and suggestion: It needs 1 data table Enquiry and 2 views are there admin and user.

8. General info: It needs 1 data table Issuing section and 2 views are there admin and user.

Number of Reports:

1. Billed Items Report
2. Inventory Report
3. Track Status Report

Measuring Weight Complexity: NAME	OBJECT	COMPLEXITY	WEIGHT
Registration	SCREEN	SIMPLE	1
Login	SCREEN	SIMPLE	1
Cart	SCREEN	SIMPLE	1
Search Items	SCREEN	SIMPLE	1
Edit Inventory	SCREEN	SIMPLE	1
Track Status	SCREEN	SIMPLE	1
Complaint	SCREEN	SIMPLE	1
General Info	SCREEN	SIMPLE	1
Billed Items Report	REPORT	MEDIUM	5
Track Status Report	REPORT	MEDIUM	5
Inventory Report	REPORT	MEDIUM	5
View Report	REPORT	MEDIUM	5
Registration	3GL	DIFFICULT	10
Login	3GL	DIFFICULT	10
Verification Of User Details After Login	3GL	DIFFICULT	10
Forgot Password In Login	3GL	DIFFICULT	10
Search Items	3GL	DIFFICULT	10
Item Details	3GL	DIFFICULT	10
Bill	3GL	DIFFICULT	10
TOTAL			**98**

Object Points

Add all the weighted object instances to get one number and this is known as object point count.

Object Point = ∑ (number of object instances) * (Complexity weight of each object instance)

= (8*1)+(4*5)+(7*10)
= 98

NOP

NOP are the object point that will need to be developed and differ from the object point count because there may be reuse of some object instance in the project.

NOP =(object points) x [(100-reuse%)/100]
NOP =98 (as reuse% is 0)

Calculating Productivity Rate (PROD):

Here, the Developer's Experience is Very Low. Therefore, PROD = 4.

Calculating estimated effort

Estimated Effort = NOP/PROD
= 98/4
= 24.5
= 25 person-month

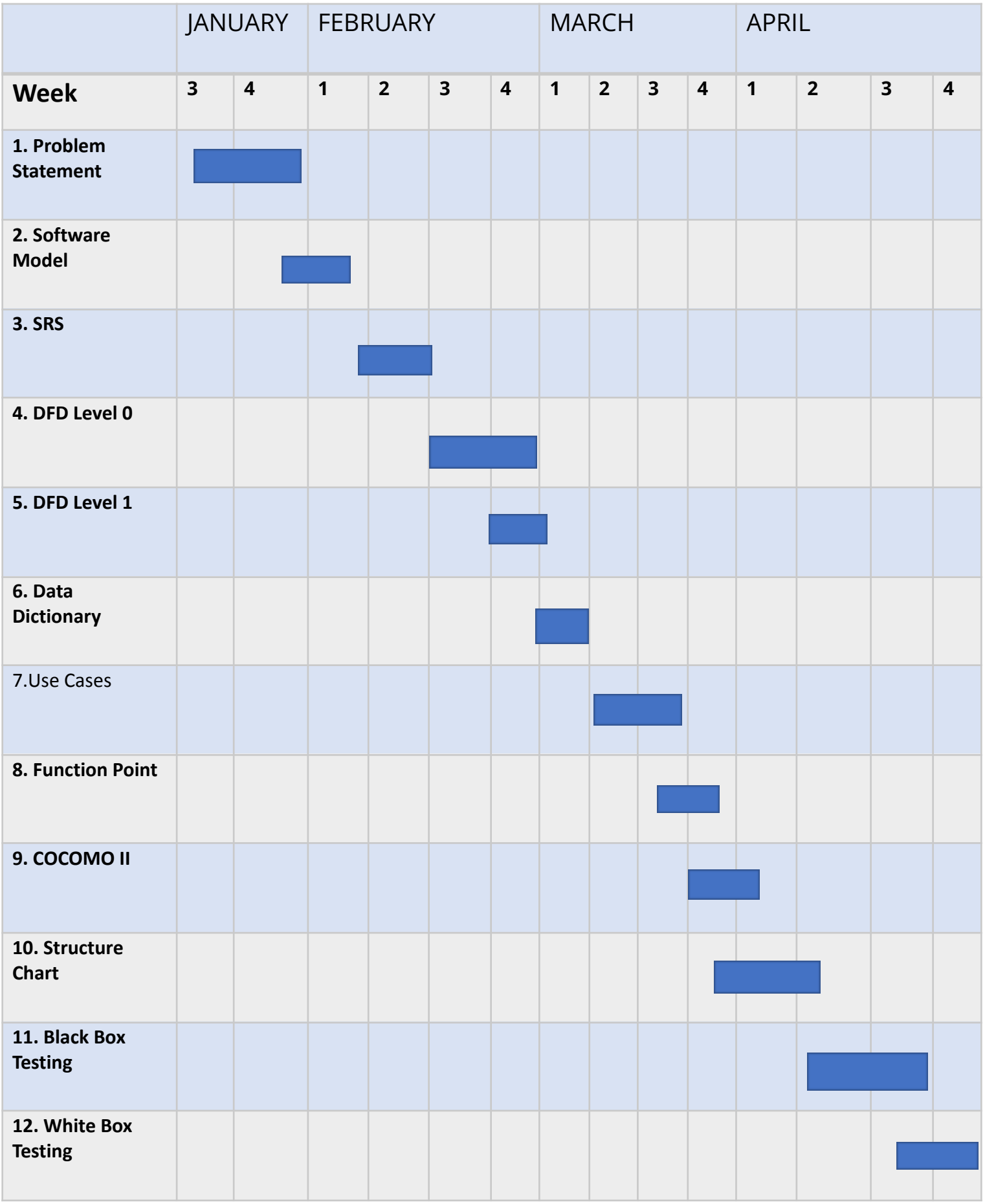
Calculating Estimated Cost:

Now, given a burdened labour rate of $8000 per month.

Or, Cost = $(25 * 8000) = $168000.

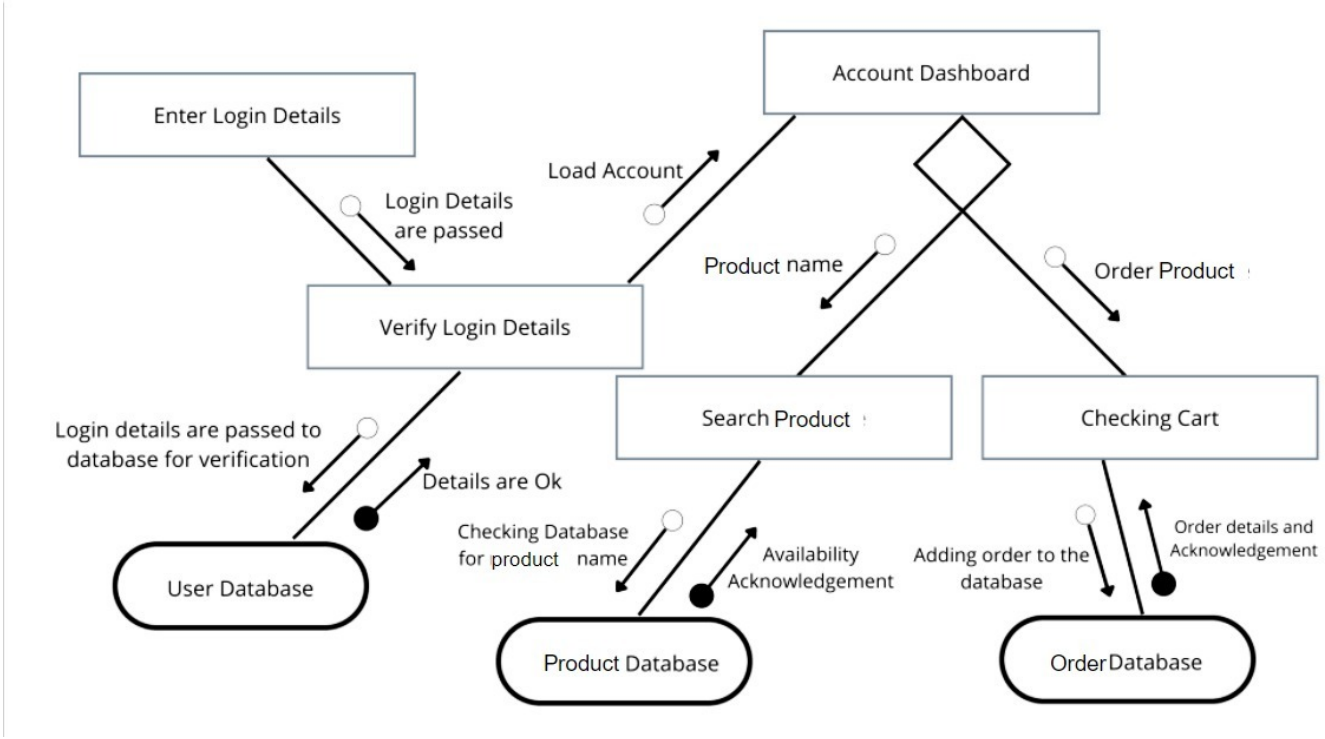
Therefore,the cost to do the project is $2,00,000.

GANTT CHART

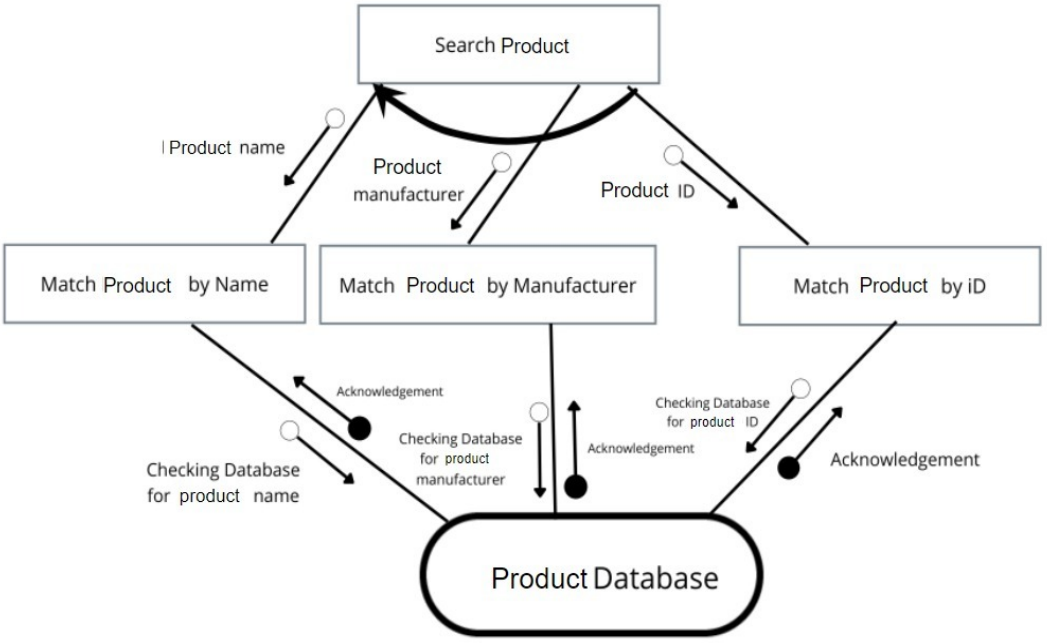


STRUCTURE CHART

1. LOGIN



SEARCH PRODUCT



BLACK BOX TESTING

Test cases for Boundary value analysis:

Experience shows that test cases that are close to boundary conditions have a higher chance of detecting an error. Here, boundary conditions means an input value may be on the boundary, just below the boundary (upper side) or just above the boundary (lower side).
We, here, perform Black Box Testing on the LOGIN module.
The range for username is 6-18 and password is 8-10 characters. Both can be alphanumeric.
Password should have a minimum one special character and minimum one upper case letter.
The combination of username and password must match from the values in the database.

Best Case:

As we know, with the single fault
assumption theory, 4n + 1 test cases can be designed and which are here in this case equal to 9. The boundary value test cases are:

Test Cases	Username	Password	Expected Output
1	Neelam Roy	123456@Abc	Invalid
2	Rohit Sharma	123456@Abc	Invalid
3	Jai Singh	456789@Xyz	Valid
4	Ishika Suneja	456789@Xyz	Invalid
5	Nehal Chawla	123456@Xyz	Invalid
6	Sunita Rawat	Oyz$7321	Invalid
7	Sunita Rawat	abC123@3	Invalid
8	Sunita Rawat	jmS?67	Valid
9	Sunita Rawat	65@?#1Z	Invalid

Worst Case:

If we reject the “single fault” assumption theory of reliability and may like to see what happens when more than one variable has an extreme value.

Worst case testing for a function of n variables generates test cases. As our login module takes 2 input variables

Thus, total test cases = = 25

Therefore, the no. of test cases for the login module are 25.

Test Cases	Username	Password	Expected Output
1	Neelam Roy	123456@Abc	**Valid**
2	Neelam Roy	pqR@113355	**Invalid**
3	Neelam Roy	xyZ@1122	**Invalid**
4	Neelam Roy	srT@98	**Invalid**
5	Neelam Roy	456@1As	**Invalid**
6	Rohit Sharma	123456@Abc	**Invalid**
7	Rohit Sharma	pqR@113355	**Valid**
8	Rohit Sharma	xyZ@1122	**Invalid**
9	Rohit Sharma	srT@98	**Invalid**
10	Rohit Sharma	456@1As	**Invalid**
11	Jai Singh	123456@Abc	**Invalid**
12	Jai Singh	pqR@113355	**Invalid**
13	Jai Singh	xyZ@1122	**Valid**
14	Jai Singh	srT@98	**Invalid**
15	Jai Singh	456@1As	**Invalid**
16	Ishika Suneja	123456@Abc	**Invalid**
17	Ishika Suneja	pqR@113355	**Invalid**
18	Ishika Suneja	xyZ@1122	**Invalid**
19	Ishika Suneja	srT@98	**Valid**

20	Ishika Suneja	456@1As	**Invalid**
21	Nehal Chawla	123456@Abc	**Invalid**
22	Nehal Chawla	pqR@113355	**Invalid**
23	Sunita Rawat	xyZ@1122	**Invalid**
24	Sunita Rawat	srT@98	**Invalid**
25	Sunita Rawat	456@1As	**Valid**

Robustness Testing:

It is nothing but the extension to boundary value analysis. In robustness testing, the software is tested by giving invalid values as inputs. Robustness testing is usually done to test exception handling.

There are **four** additional test cases which are outside the legitimate input domains. Hence, total test cases in robustness testing is **6n + 1**, where n is the number of input variables.

As n=2(no. of input variables)

Thus, total test cases = (6*2+1) = 13

Therefore, the no. of test cases for the login module are 13.

Test Cases	Username	Password	Expected output
1	Neelam Roy	123456@Abc	Invalid
2	Rohit Sharma	123456@Abc	Invalid
3	Jai Singh	456789@Xyz	Valid
4	Ishika Suneja	456789@Xyz	Invalid
5	Nehal Chawla	123456@Xyz	Invalid
6	Sunita Rawat	Oyz$7321	Invalid
7	Sunita Rawat	abC123@3	Invalid

8	Sunita Rawat	jmS?67	Valid
9	Sunita Rawat	65@?#1Z	Invalid
10	Sunita Rawat	jkL@7890	Invalid
11	Sunita Rawat	pqR@113355	Invalid
12	Neelam Roy	123456@Abc	Invalid
13	Shashank Dev	123456@Abc	Invalid

WHITE BOX TESTING

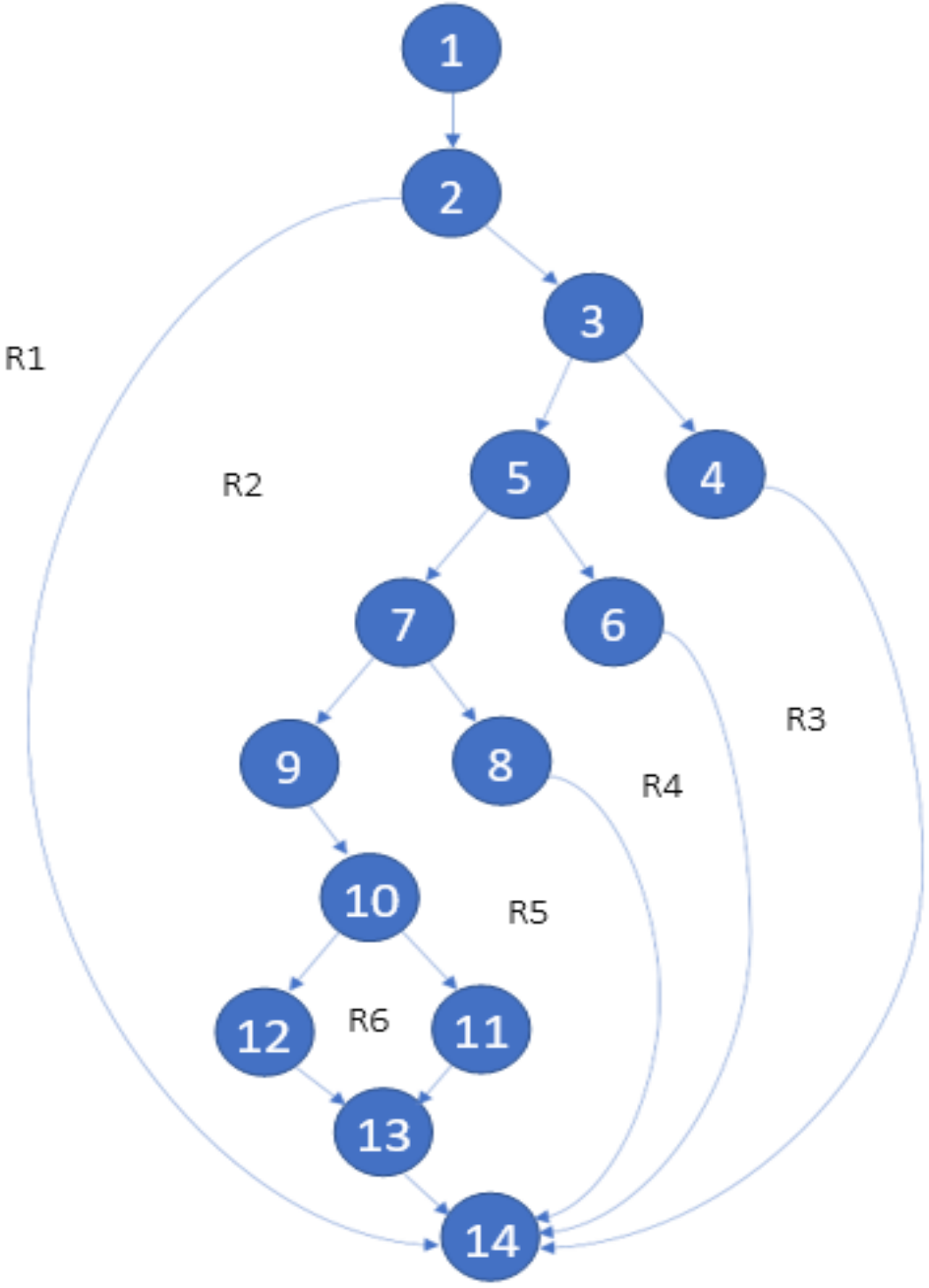
❖We are performing this white box testing for register screen:

CODE:

```
<?php
1. require 'db.php';
2. if(isset($_POST["submit"]))
{
$user = $_POST["txt3"];
$pass = $_POST["txt4"];
$cpass = $_POST["txt5"];
$gen = $_POST["buttons"];
$mail = $_POST["txt6"];
3.          if (empty($user) || empty($mail) || empty($pass) ||
empty($cpass))
4. {       echo "FILL IN ALL THE FIELDS";
}
5. else if(!filter_var($mail, FILTER_VALIDATE_EMAIL))
{
6. echo "wrong email";
}
7. else if($pass!=$cpass)
{
8. echo "password mismatch";
}
```

```
9.  Else
{
$sql = " Insert Into users(username, email, passwd, gender)
values('$user', '$mail', '$pass', '$gen')";
10. if(mysqli_query($conn, $sql))
{
11. echo "Registered successfully"
}
12. else echo "Connection error";
13. }
14. }
?>
```

FLOW GRAPH :



CYCLOMATIC COMPLEXITY: Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program. The nodes in the graph indicate the smallest group of commands of a program, and a directed edge in it connects the two nodes i.e. if second command might immediately follow the first command.

There are three techniques to calculate the cyclomatic complexity which are as follows:

1) **V(G)= E-N+2*P**
   where, E= number of edges in the flowgraph
   N= number of nodes in the flowgraph
   P= number of connected components

   So, according to our flowgraph:
   E= 18, N=14, P=1
   Thus, V(G)=E-N+2*P
   =18-14+2
   =6
2) **V(G)= P+1**
   where, P= number of predicate nodes
   So, according to our code:
   P=5
   Thus, V(G)=P+1
   =5+1
   =6
3) **V(G)= Total number of regions**
   Hence, according to our flowgraph:
   No. of regions= 6 i.e., R1,R2,R3,R4,R5,R6
   Thus, V(G)= 6

SO, THE CYCLOMATIC COMPLEXITY, V(G)= 6

Independent Paths

An independent path is any path through the program that introduces at least one new set of processing statements or a new condition. When stated in terms of a flow graph, an independent path must move along at least one edge that has not been traversed before the path is defined.

From the above **Flow Graph of Code,** the **INDEPENDENT PATHS** are:

Path 1: 1 → 2 → 14

Path 2: 1 → 2 → 3 → 4 → 14

Path 3: 1 → 2 → 3 → 5 → 6 → 14

Path 4: 1 → 2 → 3 → 5 → 7 → 8 → 14

Path 5: 1 → 2 → 3 → 5 → 7 → 9 → 10 → 11 → 13 → 14

Path 6: 1 → 2 → 3 → 5 → 7 → 9 → 10 → 12 →
13 → 14 Six Regions are: R1, R2, R3, R4, R5, R6

SCREENSHOT OF LOGIN MODULE AND DATABASE