POKHARA UNIVERSITY

CRIMSON COLLEGE OF TECHNOLOGY

Butwal-11, Devinagar, Rupandehi

A Project Report On

STOCK MANAGEMENT SYSTEM

[Project Code: PRJ 151]

Submitted in Partial Fulfillment of the Requirement for Degree of Bachelor of Computer Application Awarded by Pokhara University

Under the Supervision of
Suraj Pandey
Department of Science and Technology
Crimson College of Technology

Submitted By:

JEEVAN KAMI

[PU Reg.No: 2024-1-53-0120, Exam Roll No. 24530406]

ANKIT BHUSAL

[PU Reg.No: 2024-1-53-0111, Exam Roll No. 24530397]

PRASHANTA NEUPANE

[PU Reg.No: 2024-1-53-0130, Exam Roll No. 24530415]

ANJIL KHANAL

[PU Reg.No: 2024-1-53-0110, Exam Roll No. 24530396]

Submitted To:

Crimson College of Technology
Department of Science and Technology
Butwal-11, Devinagar, Rupandehi

November, 2025

Crimson College of Technology
(Pokhara University)

Devinagar, Butwal-11, Rupandehi, Lumbini, Nepal
Tel : 00977-71-410380, 410479
Fax : 00977-71-410379
E-mail : admin@cct.edu.np/crimsoncollege24@gmail.com
Website : www.cct.edu.np

Ref.No…….

LETTER OF APPROVAL

This is to certify that this project prepared by **"JEEVAN KAMI :** PU Reg. No: 2024-1-53-0120, Exam Roll No. 24530406, **ANKIT BHUSAL :** PU Reg.No: 2024-1-53-0111, Exam Roll No. 24530397, **PRASHANTA NEUPANE :** PU Reg.No: 2024-1-53-0130, Exam Roll No. 24530415, **ANJIL KHANAL** : PU Reg.No: 2024-1-53-0110, Exam Roll No. 24530396 ", entitled **"STOCK MANAGEMENT SYSTEM"** in partial fulfilment of the requirement for Bachelor's Degree in Computer Application of Pokhara University has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

……..	…..……
Signature	Signature
Mr. Abdul Haq	Mr. Suraj Pandey
Co-ordinator	Supervisor
Department of IT	

CERTIFICATE FROM EXTERNAL EXAMINER

This is to certify that the work carried out by **"Mr.Jeevan Kami, Mr.Ankit Bhusal, Mr.Prashanta Neupane and Mr.Anjil Khanal"** for the completion of the project entitled **"Stock Management System"** in the fulfillment of the requirement of the degree of Bachelor of Computer Application has been accomplished, presented and demonstrated successfully.

During the presentation, I found the student to be enthusiastic, hardworking and authentic and ready with skills to undertake any work related to IT and management.

……..

External Examiner

SUPERVISOR'S RECOMMENDATION

The project documentation in the report carried out by four mentioned second semester undergraduate students of Bachelor of Computer Application (BCA) for the fulfillment of the requirement for the project of second semester under the supervision of **Mr. Suraj Pandey.**

.....

Mr. Suraj Pandey

Supervisor

Department of IT

Crimson College of Technology

STUDENT DECLARATION

We hereby declare that project report entitled **"Stock Management System"** submitted in partial fulfillment of the requirement for Bachelor's Degree of Computer Application of Pokhara University, in our original work and not submitted for the award of any other degree, diploma, fellowship, or any other similar title or prize.

Thanking From:

Jeevan Kami (24530406)

……..

Ankit Bhusal (24530397)

……..

Prashanta Neupane (24530415)

……..

Anjil Khanal (24530396)

……..

ACKNOWLEDGEMENT

In the completion of this project entitled **"Stock Management System,"** all members of the group have contributed equally with dedication and effort. We would like to thank Crimson College of Technology for assigning us this project, as it has provided us with an important opportunity to learn and prepare for our future career development. We are also sincerely thankful to our project supervisor, **Mr. Suraj Pandey,** whose cooperation, guidance, and valuable input played a vital role in bringing this project to this level.

We would further like to express our gratitude to **Mr. Abdul Hak,** IT Coordinator, for his support, interest, and suggestions that helped us improve the quality of this work. Last but not least, we remain thankful to all our teachers, seniors, and friends for their constant encouragement, support, and guidance throughout the completion of this project.

PREFACE

With the changes in time and new developments in computer science, human life is no longer static. The invention of computers has made our lives easier, more comfortable, and reliable.

In this project, we aimed to develop the **Stock Management System,** which provides detailed information and manages tasks efficiently. All group members have put in their best effort to make this project error-free, and any issues that may arise related to the project will be addressed sincerely by us.

Table of Contents

List of Abbreviation 1

List of Figure 2

ABSTRACT 3

CHAPTER 1: INTRODUCTION 4

1.1 Introduction 5

1.2 Problem Statement 5

1.3 Objectives 6

1.4 Scope and Limitations 6-7

1.5 Methodology 8

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW 9

2.1 Background Study 10

2.2 Literature Review 10-11

2.3 Summary 11

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN 12

3.1 System Analysis 13

3.1.1 Requirement Analysis 13

3.1.2 Feasibility Study 14

3.1.3 Gantt Chart 15

3.1.4 Data Flow Diagram 16

3.1.5 Flowchart 17

CHAPTER 4: SYSTEM REQUIREMENTS 18

4.1 Hardware Requirements 19

4.2 Software Requirements 19-20

CHAPTER 5: METHODOLOGY USE FOR TESTING 21

5.1 Methodology use for Testing 22

5.1.1 Testing Methods 22-23

CHAPTER 6: CONCLUSION AND FURTHER ENHANCEMENT 24

6.1 Conclusion 25

6.2 Further Enhancement 25-26

APPENDICS 27-35

8.1 Source Code 27-30

8.2 Screenshots 31-34

8.2 Supervisor log file 35

REFERENCE 36

List of Abbreviation

BCA – Bachelor in Computer Application

CRUD – Create, Read, Update, Delete

DB – Database

GUI – Graphical User Interface

ID – Identification

SMS – Stock Management System

List of Figure

Fig 1: Waterfall Model

Fig 3.1: Gantt Chart

Fig 3.2: 0 level Data Flow Diagram

Fig 3.2: 0 level Data Flow Diagram

Fig 3.4: Flowchart

ABSTRACT

This project report presents the design and development of a **Stock Management System** using the C programming language. Manual and spreadsheet based inventory management often result in inefficiency, duplication, and errors. To address these issues, the proposed system provides a lightweight, console based application capable of handling basic inventory tasks such as adding, updating, deleting, searching, and viewing stock records. The system uses file handling concepts for persistent data storage [3], ensuring that stock details remain available between sessions without requiring an external database or internet connection. It is simple, user friendly, and suitable for small businesses and educational purposes [2]. By automating repetitive inventory tasks, the Stock Management System improves accuracy, reduces human errors, and saves time, making it an effective alternative to traditional manual methods [1][6].

CHAPTER 1: INTRODUCTION

1.1 Introduction

Stock management plays a vital role in businesses that deal with goods or materials. Traditional manual methods such as paper based ledgers or spreadsheet tracking are inefficient and often lead to calculation errors, data duplication, or even loss of records [4].

The Stock Management System (SMS), developed using C programming, aims to simplify inventory control by automating the main operations. Using file handling, it enables the storage and retrieval of data without requiring any external database or internet connection [2]. The menu driven console interface allows users with limited technical knowledge to perform stock operations conveniently. The system is reliable, lightweight, and designed for small enterprises, local shops, and educational purposes.

1.2 Problem Statement

Small and medium sized businesses frequently depend on manual or spreadsheet based systems to track their inventories. These methods are time consuming, error prone, and lack centralized control. Without an automated solution, problems such as stock shortages, over purchasing, and inconsistent data arise.

The proposed Stock Management System aims to overcome these limitations by providing a computerized solution using the C language, emphasizing simplicity, accuracy, and reliability [1][2].

Identified Problems:

- Manual entry errors and missing data.
- Difficulty in searching and updating stock quickly.
- Lack of real time updates and summary reports.
- High time consumption in repetitive manual calculations.
- Absence of secure and organized data storage.

1.3 Objectives

The main objectives of the Stock Management System are as follows:

1. To identify and analyze the limitations of manual and spreadsheet based inventory systems [3].
2. To design a C based stock management application with a user friendly menu.
3. To use file handling for persistent storage of stock data [1].
4. To ensure ease of operation for users with minimal technical knowledge.
5. To reduce human errors and increase data accuracy and reliability [4].

1.4 Scope and Limitations

The Stock Management System (SMS) developed in C is primarily designed to assist small scale businesses, shops, and educational institutions in managing their inventory efficiently. Its scope includes:

- The system allows adding, updating, deleting, searching, and viewing stock items.
- Data is stored using file handling, ensuring records are saved between sessions.
- The menu driven interface makes it easy for users with minimal technical knowledge.
- Suitable for small businesses, local shops, and academic projects.
- Lightweight and can run on basic computer systems without additional software [2].

While the Stock Management System (SMS) addresses many issues associated with manual and spreadsheet based inventory management, it also has certain limitations:

- Designed for a single user; cannot support multiple simultaneous users.
- Console based interface (non GUI).
- Does not generate detailed reports, charts, or analytics.
- Uses text files for storage, limiting scalability for large inventories.

- No online or cloud access for remote inventory management.
- Lacks user authentication or access control, which may affect data security [3][4].

1.5 Methodology

The system development process follows the Waterfall Model, which is one of the earliest structured methodologies in software engineering. According to *Pressman (2014)* [3] and *Sommerville (2016)* [4], the Waterfall Model ensures systematic progress through phases such as requirement analysis, design, implementation, testing, and maintenance. Each stage must be completed before the next begins, minimizing confusion and maintaining project clarity. This model consists of the following phases:

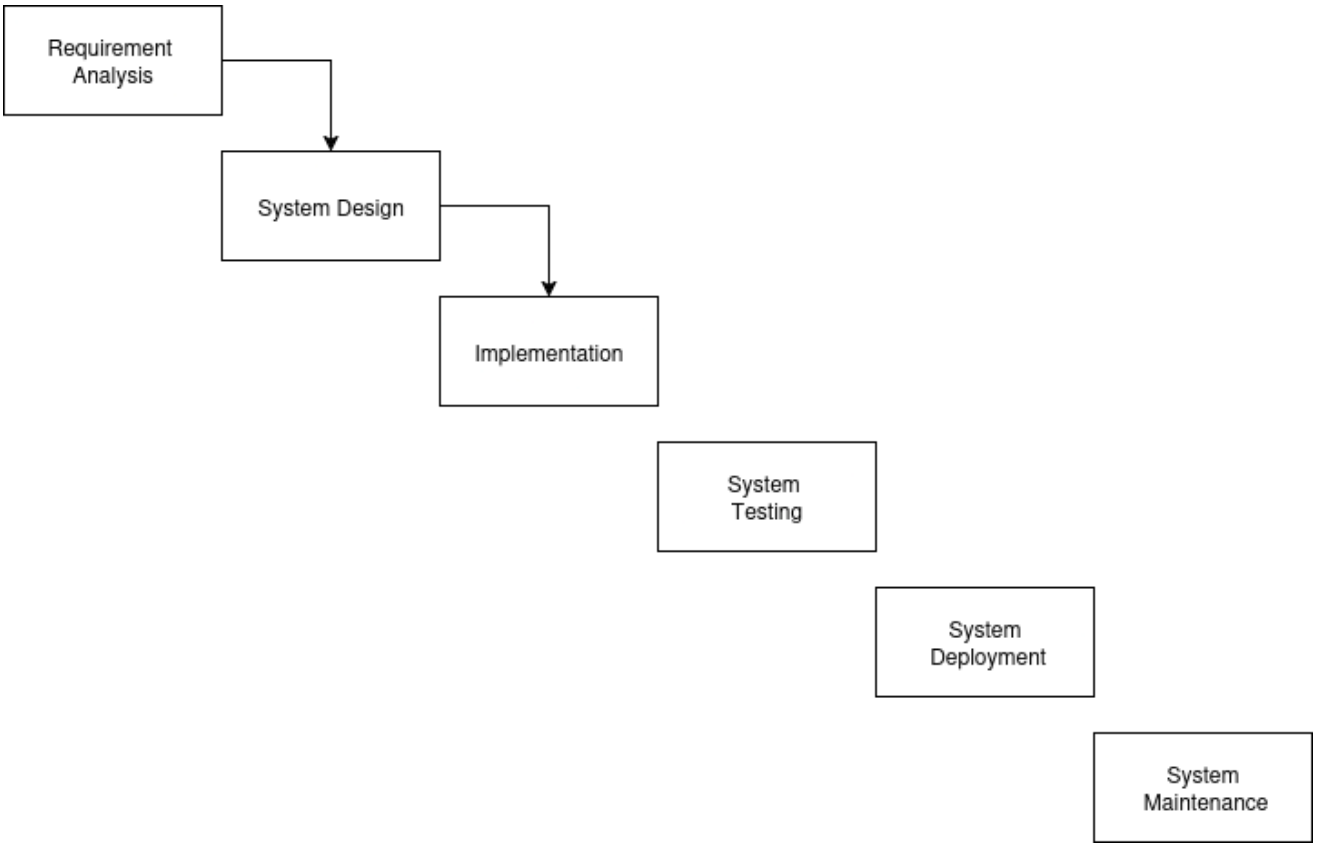


Fig 1: Waterfall Model

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Stock management, also known as inventory management, is a crucial function for businesses that handle physical goods. It involves monitoring the movement of products from suppliers to storage and finally to customers. Effective stock management ensures that the right quantity of products is available at the right time, thereby minimizing overstocking, stock outs, and financial losses [1].

Traditionally, businesses have relied on manual or spreadsheet based inventory systems. Although these methods are simple and inexpensive, they are highly prone to human errors such as duplicate entries, missing records, and calculation mistakes. Manual inventory systems also require significant time and effort and lack real time updates, which makes it difficult for managers to make timely and informed decisions [2][3].

Modern inventory management systems, in contrast, offer automation, higher accuracy, and faster data processing. However, many commercial software solutions are costly, require constant internet connectivity, and demand technical expertise, which makes them unsuitable for small scale enterprises or educational use [4].

This limitation has motivated the development of a lightweight, offline, and console based Stock Management System using the C programming language. Such a system provides essential stock control functions like adding, updating, deleting, searching, and viewing inventory data while remaining efficient, simple, and suitable for small scale inventory management [5].

2.2 Literature Review

According to Balagurusamy (2019) [1] and Kanetkar (2018) [2], the C programming language provides strong control over memory management, data structures, and file handling, making it an ideal choice for developing small scale, efficient, and console based applications. These authors emphasize that C remains one of the most reliable and performance oriented languages for system level programming and educational projects.

Traditional inventory systems that rely on manual or spreadsheet based record keeping often face challenges such as inconsistency, redundancy, and data loss. Pressman (2014) [3] and Sommerville (2016) [4] explain that automation through structured software development reduces such errors and enhances data integrity, reliability, and overall system performance. Their studies further suggest that

following systematic development models such as the Waterfall Model ensures each stage of design and testing is properly validated, resulting in robust and maintainable software.

Additionally, Sebesta (2017) [5] discusses the significance of modular programming, where programs are divided into reusable modules or functions. This approach simplifies maintenance, improves code clarity, and enhances scalability an essential aspect of software like the Stock Management System. Combining file handling capabilities and modular programming allows C based systems to manage persistent data effectively without requiring a complex database or external connectivity.

In summary, the literature demonstrates that C programming is well suited for developing lightweight, educational, and small scale management systems. By integrating structured development methods, file handling, and modular design principles, the proposed Stock Management System can deliver a reliable and efficient inventory solution appropriate for academic and small business contexts.

2.3 Summary

The background study and literature review indicate that small businesses face considerable challenges with manual or spreadsheet based inventory management. Errors, inefficiency, and the lack of real time updates are common issues. Modern software solutions provide accuracy and automation but are often too costly or complex for small scale use.

Developing a C based, console driven Stock Management System addresses these challenges by providing a simple, reliable, and cost effective solution. It allows efficient stock operations adding, updating, deleting, searching, and viewing records while ensuring data persistence through file handling. Modular programming and structured development models enhance maintainability and usability, making the system both practical and educational.

Thus, the proposed system is well suited for small businesses and academic projects, offering a lightweight and user friendly alternative to conventional inventory management approaches [1][2][3][4][5].

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

System analysis is the process of understanding the existing system, identifying problems, and defining requirements for the proposed solution. The goal is to design a system that efficiently manages inventory, reduces errors, and simplifies stock operations for small businesses using C programming and file handling.

3.1.1 Requirement Analysis

Requirement analysis defines what the system must do (functional requirements) and how it must perform (non functional requirements).

a) Functional Requirements:

- Add, update, delete, search, and view stock items.
- Store and retrieve stock data persistently using file handling.
- Display confirmation messages for each operation.
- Menu driven interface for ease of navigation.

b) Non Functional Requirements:

- Lightweight and operable on basic computers.
- User friendly with minimal technical knowledge required.
- Fast and reliable performance.
- Data integrity and minimal risk of loss.

3.1.2 Feasibility Study

Based on the initial investigation, the project was expanded to include a detailed feasibility study. A feasibility study evaluates a proposed system in terms of its workability, organizational impact, ability to meet user requirements, and effective use of available resources. It addresses several critical questions:

1. What are the user's demonstrable needs, and how does the proposed system fulfill them?
2. What resources are available to develop and implement the system?
3. What are the likely organizational impacts of the proposed system?
4. Is it worthwhile to address the identified problems through this system?

In the context of this project, the feasibility analysis focused on identifying current issues in stock management and exploring ideas for a new, efficient solution. The study followed a structured sequence of steps to ensure thorough evaluation:

- Form a project team and appoint a project leader; prepare system flowcharts to visualize existing and proposed processes; enumerate potential system solutions.
- Define and identify the characteristics and functionalities of each proposed system.
- Evaluate the performance, cost effectiveness, and practical suitability of each system alternative.
- Assign weights to system performance and cost data to aid in decision making.
- Select the most suitable proposed system based on combined analysis.
- Prepare and submit a final project directive to the management for approval.

This structured approach ensures that the Stock Management System (SMS) is practical, cost effective, and capable of meeting user requirements. It minimizes risks while maximizing the system's overall benefits and organizational impact.

3.1.3 Gantt Chart

	August September				
Task	Aug 01 Aug 05	Aug 06 Aug 12	Aug 13 Aug 21	Aug 22 Aug 28	Aug 28 Sep 01
Problem Identification					
Requirement Analysis					
System Design					
Debugging & Testing					
Implementation					
Documentation					

Fig 3.1 : Gantt Chart

3.1.4 Data Flow Diagram

Fig 3.2: 0 level Data Flow Diagram

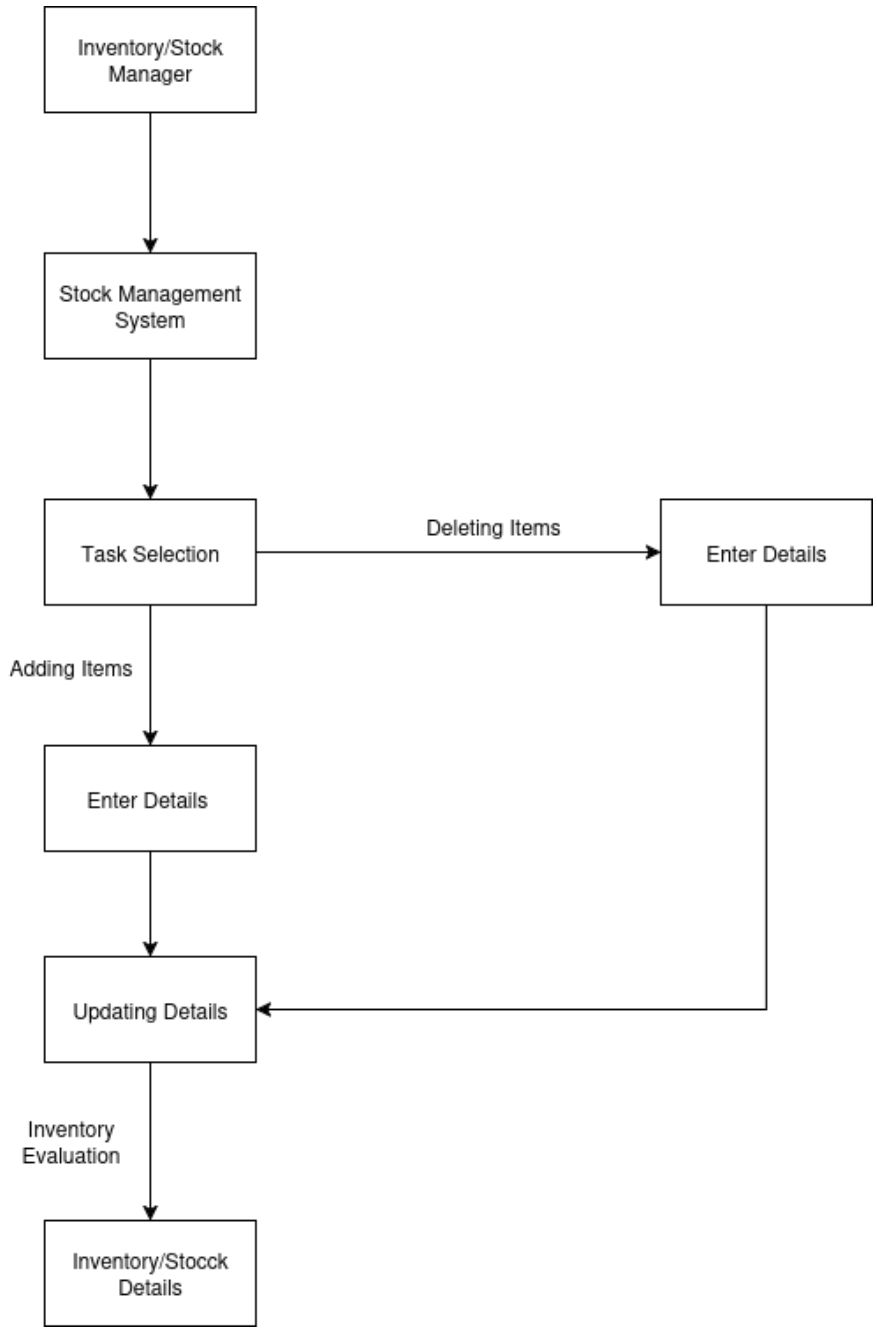


Fig 3.3: 1 level Data Flow Diagram

3.1.5 Flowchart

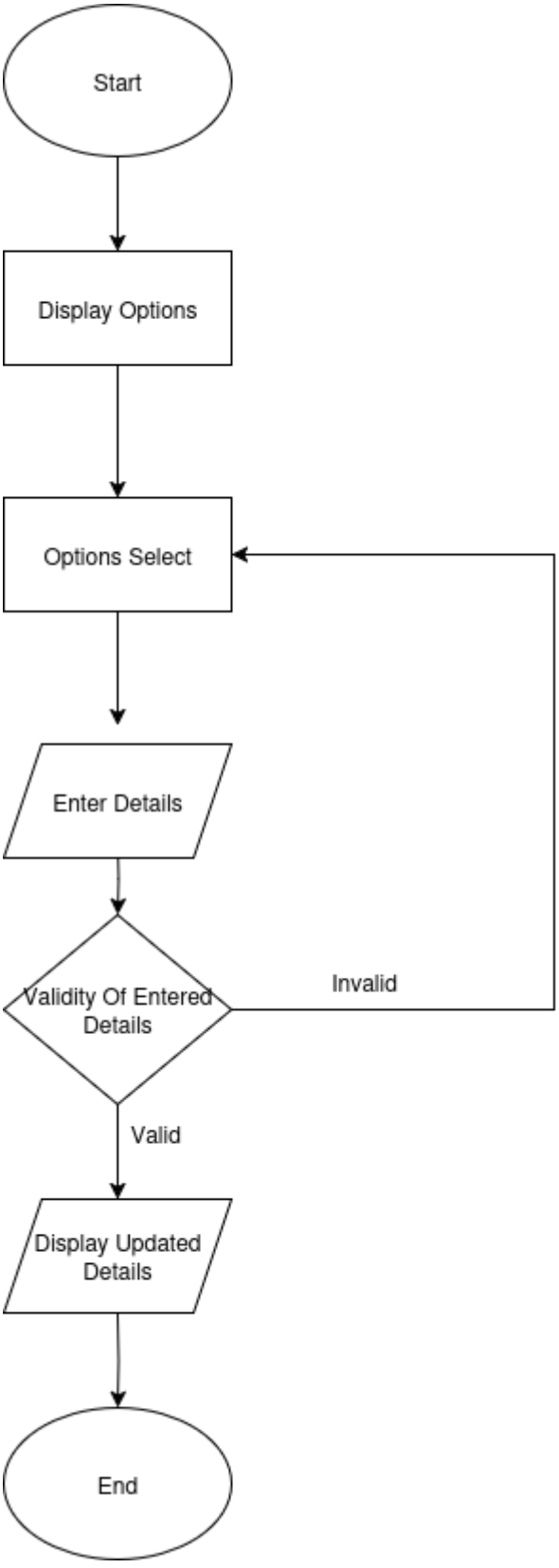


Fig 3.4 : Flowchart

CHAPTER 4: SYSTEM REQUIREMENTS

4.1 System Requirements

System requirements define the essential hardware and software components necessary to implement and run the Stock Management System (SMS) efficiently. These requirements ensure that the system operates reliably, meets user expectations, and supports all stock management operations. According to Pressman (2014) [3] and Sommerville (2016) [4], identifying and defining system requirements is critical for smooth system performance and usability, as well as for minimizing errors during deployment.

4.1.1 Hardware Requirements

The Stock Management System is designed to be lightweight and capable of running on standard computer hardware. The minimum hardware requirements include:

- Processor: Intel Core i3 or equivalent
- RAM: 4 GB or higher
- Storage: 100 MB of free disk space
- Display: Minimum resolution of 1024×768 pixels
- Input Devices: Keyboard and mouse
- Operating System**:** Windows 7, 8, or 10; alternatively, Linux

These specifications ensure that the system can run a console based C application efficiently with file handling for persistent storage. The design allows small businesses, shops, and educational institutions to use the system without the need for advanced hardware, aligning with the principle of cost effective and accessible software design emphasized by Balagurusamy (2019) [1] and Kanetkar (2018) [2].

4.1.2 Software Requirements

The software requirements define the tools and environments necessary to develop, compile, and execute the Stock Management System efficiently:

- Operating System: Windows (7/8/10) or Linux
- Compiler: Any standard C compiler such as Turbo C, Dev C++, GCC, or Code: Blocks IDE
- Text Editor / IDE: Notepad++, Code: Blocks, or Dev C++ for coding, debugging, and file management

- File Handling Support: The system must allow reading from and writing to text files to ensure persistent data storage
- Optional Tools: PDF reader for viewing reports and documentation

Using standard and lightweight components ensures the system is user friendly, maintainable, and cost effective, in line with good software engineering practices [3][4][5]. These requirements allow the Stock Management System to function smoothly across different platforms while remaining accessible for educational purposes and small scale business operations.

CHAPTER 5: METHODOLOGY USE FOR TESTING

5.1 Methodology use for Testing

Testing is a critical phase in software development, ensuring that the system performs as expected and meets defined functional and non functional requirements. For the Stock Management System (SMS), testing ensures that all core operations adding, updating, deleting, searching, and viewing stock records function correctly, and that data is accurately stored and retrieved using file handling mechanisms. Proper testing reduces the risk of errors, enhances reliability, and improves user confidence in the system [3][4].

Testing also validates the overall design and implementation, confirming that the software meets user needs, complies with specifications, and is robust under real world conditions. For this project, a combination of Black Box, White Box, Grey Box, and Acceptance Testing methods was used to comprehensively evaluate the system.

5.1.1 Testing Methods

Testing is an essential phase to ensure that the Stock Management System functions correctly, reliably, and efficiently. For this project, the following testing methods were used:

a. **Black Box Testing:**

Black Box Testing was the primary testing method employed. It focuses on verifying system functionality without examining internal code structures. Inputs were provided to the system, and outputs were observed to ensure correctness. All operations, including adding, updating, deleting, searching, and viewing stock items, were tested to ensure they produced the expected results. This method validates the system from the **end user perspective** and ensures functional requirements are met [3][4].

b. **White Box Testing:**

Limited White Box Testing was performed on critical functions, especially those handling **file operations** and core modules such as addStock(), updateStock(), and deleteStock(). This method examines the internal logic of the program to verify that each function behaves correctly and that data stored in files maintains integrity. White Box Testing helps identify hidden errors within the code that could affect system reliability [5].

c. **Grey Box Testing:**

Grey Box Testing combines aspects of Black Box and White Box methods. In this project, it was applied to verify how data flows between modules and to ensure the

consistency and accuracy of stored stock records. This testing method allows limited knowledge of internal code while primarily focusing on functional verification, helping to detect issues that may not be evident from purely external testing [4][5].

d. **Acceptance Testing**

Acceptance Testing ensures that the system meets all user requirements and functions correctly under real world conditions. For the Stock Management System, acceptance testing focused on validating all core functionalities and user interactions. Successful completion of this testing confirms that the system is ready for deployment, reliable, and usable by small business operators or educational users [3][4].

CHAPTER 6: CONCLUSION AND FURTHER ENHANCEMENT

6.1 Conclusion

The Stock Management System (SMS) developed using C programming with file handling provides a practical and efficient solution for small scale inventory management. The system automates essential operations such as adding, updating, deleting, searching, and viewing stock items, which significantly reduces the risk of human errors and improves overall accuracy compared to manual or spreadsheet based methods [1][2].

By leveraging file handling, the system ensures data persistence, allowing stock records to remain available across sessions without requiring an external database or internet connectivity. The menu driven interface provides a simple and user friendly environment, making the system accessible to users with minimal technical knowledge.

Comprehensive testing, including Black Box, White Box, Grey Box, and Acceptance Testing**,** confirmed that the system is reliable, accurate, and efficient [3][4]. As a result, the Stock Management System is cost effective**,** improves workflow efficiency, and is well suited for small businesses, shops, and educational purposes [1][2].

6.1 Further Enhancement

While the current system effectively meets its objectives, several opportunities exist for future improvement and scalability:

1. GUI Implementation: Upgrade from a console based interface to a graphical user interface (GUI) to improve usability and visual appeal for end users.
2. Database Integration: Incorporate a database such as MySQL or SQLite to support larger datasets and enable multi user access.
3. Reporting and Analytics: Add functionalities to generate stock reports, summaries, and analytics, supporting informed decision making.
4. User Authentication: Implement login and role based access control to enhance data security and control user privileges.
5. Cloud Synchronization: Enable cloud based storage to allow data access from multiple devices and remote locations.
6. Multi user Support: Support simultaneous access for multiple users, making the system suitable for medium to large scale organizations.

These enhancements will make the system more robust, scalable, and adaptable for evolving business needs, while maintaining the simplicity and cost effectiveness that make it suitable for small scale users [3][5].

APPENDICS

SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Stock {
    int id;
    char name[50];
    int quantity;
    float price;
};

void addStock();
void viewStock();
void searchStock();
void updateStock();
void deleteStock();

FILE *file;

int main() {
    int choice;
    while (1) {
        printf("\n          Stock Management System          \n");
        printf("1. Add Stock\n");
        printf("2. Update Stock\n");
        printf("3. Delete Stock\n");
        printf("4. Search Stock\n");
        printf("5. View All Stock\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: addStock(); break;
            case 2: updateStock(); break;
            case 3: deleteStock(); break;
            case 4: searchStock(); break;
            case 5: viewStock(); break;
            case 6: exit(0);
            default: printf("Invalid choice! Try again.\n");
        }
    }
    return 0;
}
```

```
void addStock() {
    struct Stock s;
    file = fopen("stock.txt", "a");
    if (file == NULL) {
        printf("Error opening file!\n");
        return;
    }
    printf("Enter Item ID: ");
    scanf("%d", &s.id);
    printf("Enter Item Name: ");
    scanf(" %[^\n]%*c", s.name);
    printf("Enter Quantity: ");
    scanf("%d", &s.quantity);
    printf("Enter Price: ");
    scanf("%f", &s.price);

    fprintf(file, "%d %s %d \t\t%.2f\n", s.id, s.name, s.quantity, s.price);
    fclose(file);
    printf("Record Added Successfully!\n");
}

void viewStock() {
    struct Stock s;
    file = fopen("stock.txt", "r");
    if (file == NULL) {
        printf("No records found!\n");
        return;
    }
    printf("\nID\tName\tQuantity\tPrice\n");
    while (fscanf(file, "%d %s %d %f", &s.id, s.name, &s.quantity, &s.price) != EOF) {
        printf("%d\t%s\t%d\t%.2f\n", s.id, s.name, s.quantity, s.price);
    }
    fclose(file);
}

void searchStock() {
    struct Stock s;
    int id, found = 0;
    printf("Enter Item ID to search: ");
    scanf("%d", &id);
    file = fopen("stock.txt", "r");
    if (file == NULL) {
        printf("No records found!\n");
        return;
    }
    while (fscanf(file, "%d %s %d %f", &s.id, s.name, &s.quantity, &s.price) != EOF) {
        if (s.id == id) {
```

```
            printf("Record Found:\nID: %d\nName: %s\nQuantity: %d\nPrice: %.2f\n", s.id, s.name,
s.quantity, s.price);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Record not found!\n");
    }
    fclose(file);
}

void updateStock() {
    struct Stock s;
    int id, found = 0;
    FILE *temp;
    printf("Enter Item ID to update: ");
    scanf("%d", &id);

    file = fopen("stock.txt", "r");
    temp = fopen("temp.txt", "w");
    if (file == NULL || temp == NULL) {
        printf("Error opening file!\n");
        return;
    }

    while (fscanf(file, "%d %s %d %f", &s.id, s.name, &s.quantity, &s.price) != EOF) {
        if (s.id == id) {
            printf("Enter New Name: ");
            scanf(" %[^\n]%*c", s.name);
            printf("Enter New Quantity: ");
            scanf("%d", &s.quantity);
            printf("Enter New Price: ");
            scanf("%f", &s.price);
            found = 1;
        }
        fprintf(temp, "%d %s %d \t%.2f\n", s.id, s.name, s.quantity, s.price);
    }

    fclose(file);
    fclose(temp);

    remove("stock.txt");
    rename("temp.txt", "stock.txt");

    if (found) {
        printf("Record Updated Successfully!\n");
    } else {
```

```
        printf("Record not found!\n");
    }
}

void deleteStock() {
    struct Stock s;
    int id, found = 0;
    FILE *temp;
    printf("Enter Item ID to delete: ");
    scanf("%d", &id);

    file = fopen("stock.txt", "r");
    temp = fopen("temp.txt", "w");
    if (file == NULL || temp == NULL) {
        printf("Error opening file!\n");
        return;
    }

    while (fscanf(file, "%d %s %d %f", &s.id, s.name, &s.quantity, &s.price) != EOF) {
        if (s.id == id) {
            found = 1;
            continue; // Skip writing this record
        }
        fprintf(temp, "%d %s %d \t%.2f\n", s.id, s.name, s.quantity, s.price);
    }

    fclose(file);
    fclose(temp);

    remove("stock.txt");
    rename("temp.txt", "stock.txt");

    if (found) {
        printf("Record Deleted Successfully!\n");
    } else {
        printf("Record not found!\n");
    }
}
```

SCREENSHOTS

a) Main Menu

b) Add Stock

c) Update Stock

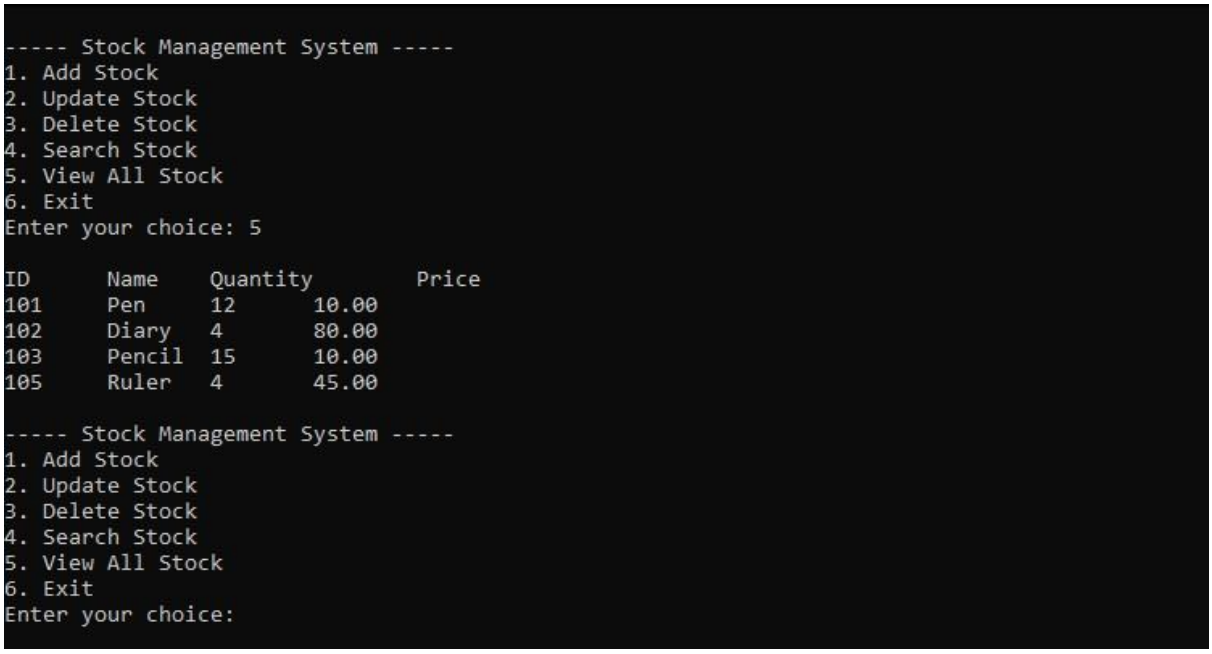
```
----- Stock Management System -----
1. Add Stock
2. Update Stock
3. Delete Stock
4. Search Stock
5. View All Stock
6. Exit
Enter your choice: 5

ID      Name    Quantity        Price
101     Pen     12      10.00
102     Notebook        24      100.00
103     Pencil  15      10.00
104     Marker  5       50.00
105     Ruler   4       45.00

----- Stock Management System -----
1. Add Stock
2. Update Stock
3. Delete Stock
4. Search Stock
5. View All Stock
6. Exit
Enter your choice: 2
Enter Item ID to update: 102
Enter New Name: Diary
Enter New Quantity: 4
Enter New Price: 80
Record Updated Successfully!
```

d) Delete Stock

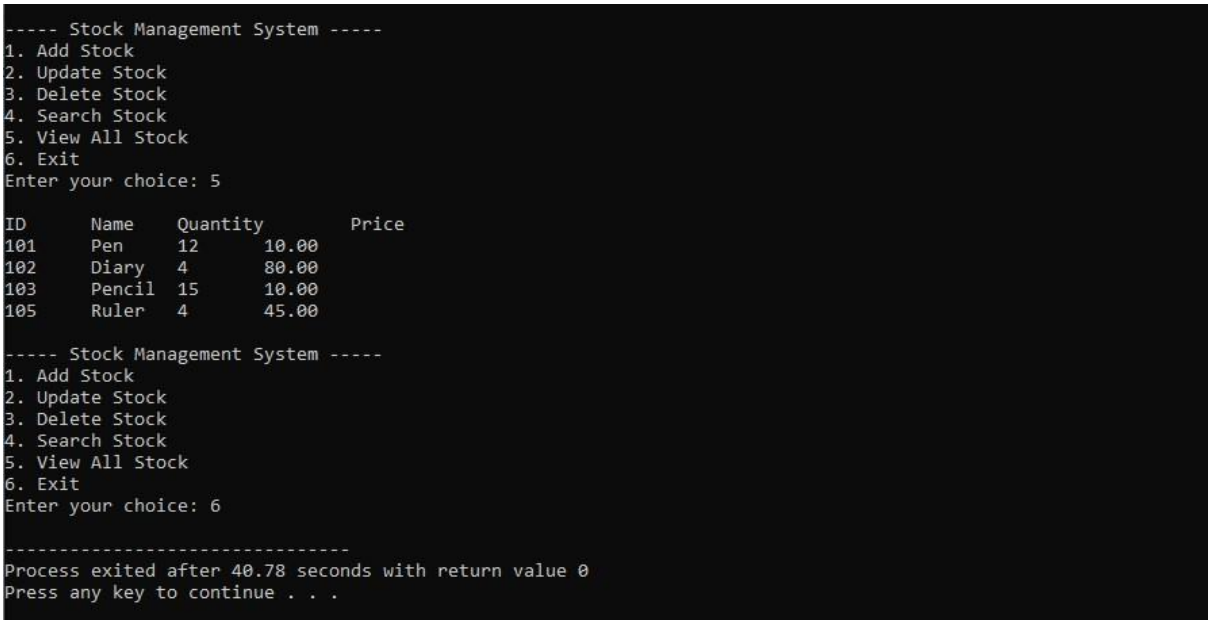
```
----- Stock Management System -----
1. Add Stock
2. Update Stock
3. Delete Stock
4. Search Stock
5. View All Stock
6. Exit
Enter your choice: 5

ID      Name    Quantity        Price
101     Pen     12      10.00
102     Diary   4       80.00
103     Pencil  15      10.00
104     Marker  5       50.00
105     Ruler   4       45.00

----- Stock Management System -----
1. Add Stock
2. Update Stock
3. Delete Stock
4. Search Stock
5. View All Stock
6. Exit
Enter your choice: 3
Enter Item ID to delete:
104
Record Deleted Successfully!
```

e) Search Stock

f) View Stock



g) To Exit


----- Stock Management System -----
1. Add Stock
2. Update Stock
3. Delete Stock
4. Search Stock
5. View All Stock
6. Exit
Enter your choice: 5
ID Name Quantity Price
101 Pen 12 10.00
102 Diary 4 80.00
103 Pencil 15 10.00
105 Ruler 4 45.00
----- Stock Management System -----
1. Add Stock
2. Update Stock
3. Delete Stock
4. Search Stock
5. View All Stock
6. Exit
Enter your choice: 6
Process exited after 40.78 seconds with return value 0
Press any key to continue . . .

SUPERVISOR LOG FILE

Name of Supervisor: Mr. Suraj Pandey Name of Project: Stock Management System

Date	Length of Session	Method of Supervision	Feedback from Supervisor	Signature of Supervisor
2082/04/16	25	Verbal Exchange	About the mid defense report and presentation	
2082/04/21	30	Direct Observation	Project progress observed, focus on requirements.	
2082/04/28	30	Direct Observation	Data Collecting for algorithm	
2082/05/06	20	Verbal Exchange	Implementation of algorithm	
2082/05/12	30	Direct Observation	Suggestions to make better user interface	
2082/05/16	20	Verbal exchange	About the data correction of redundancy while listing records.	

…… ……

Jeevan Kami **Ankit Bhusal**

(24530406) **(24530397)**

…… ……

Prashanta Neupane **Anjil Khanal**

(24530415) **(24530396)**

REFERENCE

1. Balagurusamy, E. (2019). *Programming in ANSI C.* McGraw Hill Education.
2. Kanetkar, Y. (2018). *Let Us C.* BPB Publications.
3. Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach (8th ed.).* McGraw Hill Education.
4. Sommerville, I. (2016). *Software Engineering (10th ed.).* Pearson Education.
5. Sebesta, R. W. (2017). *Programming in C (5th ed.).* Pearson Education.