

**CONSUMER COMPLAINT HANDLING SYSTEM USING C PROGRAMMING LANGUAGE**

**A PROJECT REPORT**

**Submitted by**

***B.Naga Babu(192212125)***

***T.Sai Mohith(192211229)***

*Under the guidance of*

**Dr R DHANALAKSHMI**

*in partial fulfilment for the completion of course*

**CSA0279 – C PROGRAMMING FOR BRGINNERS**

**TITLE**:

**Consumer Complaint Handling System Using C Programming LanguageS**

**PROBLEM STATEMENT:**

In modern customer-centric industries, efficient handling of consumer complaints is crucial for maintaining satisfaction, trust, and brand loyalty. As businesses grow, they often face challenges in managing and resolving customer issues promptly and effectively. A robust complaint handling system is essential to streamline the process, reduce response times, and ensure that customer grievances are addressed and resolved efficiently.

**TASKS :**

* Accept consumer complaints through a web or mobile interface.
* Automatically categorize complaints based on predefined criteria (e.g., product issues, service delays).
* Assign complaints to the correct department or personnel for resolution.
* Provide real-time tracking and status updates for customers.
* Enable feedback collection after the complaint resolution.

**OUTCOME :**

The outcome of this project is to develop an automated Consumer Complaint Handling System that accepts customer complaints, categorizes them based on the type of issue, assigns them to the appropriate department or personnel, and tracks the resolution progress. The system should also allow for feedback on the resolution process to ensure continuous improvement. By automating the complaint handling process, businesses can enhance customer experience, reduce manual workload, and improve the quality of support services. The system should be designed to handle a large volume of complaints, prioritize critical issues, and maintain an efficient resolution workflow.

**AIM**

The aim of this project is to develop an automated Consumer Complaint Handling System that accepts customer complaints, categorizes them based on the type of issue, assigns them to the appropriate department or personnel, and tracks the resolution progress. The system should also allow for feedback on the resolution process to ensure continuous improvement. By automating the complaint handling process, businesses can enhance customer experience, reduce manual workload, and improve the quality of support services. The system should be designed to handle a large volume of complaints, prioritize critical issues, and maintain an efficient resolution workflow.

**ABSTRACT**

The Consumer Complaint Handling System is a C-based application designed to streamline the process of registering, categorizing, and managing customer complaints. This system addresses the growing need for efficient grievance redressal mechanisms in modern customer-centric industries. Built using structured programming principles, the application provides a simple yet effective platform for handling consumer issues, ensuring improved customer satisfaction and trust.

The system allows users to register complaints, which are then categorized based on predefined criteria such as product or service issues. Each complaint is assigned a unique ID for tracking purposes, and its status can be updated as it progresses through resolution stages. Customers can track the progress of their complaints and provide feedback on the resolution process, promoting transparency and continuous improvement.

The application utilizes C programming features such as file handling to store and retrieve complaints, making it a lightweight yet functional solution. By automating key aspects of complaint management, this system reduces manual effort, enhances operational efficiency, and ensures that customer grievances are addressed promptly.

This project demonstrates the practical implementation of basic complaint handling in C programming, serving as a foundation for more advanced systems integrated with databases and web interfaces.

**INTRODUCTION**

In today's competitive market, effective handling of consumer complaints is essential for maintaining customer satisfaction, trust, and brand loyalty. Businesses, regardless of size, often face challenges in managing customer grievances efficiently, especially as their operations grow. The absence of a structured complaint handling system can lead to delayed resolutions, customer dissatisfaction, and potential damage to the brand's reputation.

The Consumer Complaint Handling System is a software application developed in C programming to address these challenges. This system provides a platform for registering, tracking, and managing customer complaints in an organized manner. Using C’s capabilities such as file handling and structured programming, the system ensures a streamlined process for grievance redressal.

Key functionalities of the system include:

* Registering consumer complaints.
* Categorizing complaints based on predefined criteria.
* Assigning unique IDs to track complaints.
* Updating and monitoring the status of complaints.
* Collecting feedback after complaint resolution.

By automating the complaint handling process, the system minimizes manual effort, reduces response times, and enhances operational efficiency. Though built as a standalone application, it serves as a foundation for scalable and advanced solutions integrated with databases or web-based systems.

This project demonstrates the practical application of C programming in solving real-world problems while highlighting the importance of consumer grievance management for modern businesses.

**CODE IMPLEMENTATION**

**Modules:**

* **Complaint Registration:** Accept customer details and complaint information. This module handles the initial step of the complaint management process. It allows users (customers or administrators) to provide and capture details about a complaint.

1. **Customer Details:** The system collects the customer's name and contact information (e.g., email or phone number).
2. **Complaint Information:** A brief description of the issue is entered, along with any relevant details provided by the customer.
3. **Complaint ID:** The system generates a unique identifier for each complaint, ensuring traceability.
4. **Data Saving:** The collected information is saved to a structured text file (e.g., complaints.txt).

* **Complaint Categorization:** Categorizing complaints ensures that they are handled efficiently and routed to the correct department.

1. **Product:** System collects the information from user for categorization purpose. If user selects the product then system changes the directory to products section.
2. **Service:** System collects the information from user for categorization purpose. If user selects the service then system changes the directory to products section.

* **Assignment:** Assigning complaints to the appropriate personnel or department is crucial for resolving issues effectively.

1. **Routing Rules:** Billing complaints go to the Billing Department; technical issues go to Technical Support.
2. **Personalized Assignment:** If a complaint requires specific expertise, it can be assigned to a named individual.
3. **Notification:** Notify the assigned team or personnel about the new complaint.

* **Status Tracking:** This module tracks the progress of complaints and updates their statuses.

1. **Status Options:**
2. Open: Complaint is registered but not yet assigned.
3. In Progress: The complaint is being handled.
4. Resolved: The issue has been addressed and closed.
5. **Status Updates:** Administrators can update the status as the complaint progresses.
6. **View Status:** Customers or administrators can view the current status using the Complaint ID.

* **Feedback Collection:** Once a complaint is resolved, this module records customer feedback on the resolution process.

1. **Prompt for Feedback:**After marking a complaint as Resolved, prompt the customer to provide feedback (e.g., a rating or comment).
2. **Store Feedback:** Append the feedback to the existing complaint record in the file.
3. **Use Feedback for Improvement:** Analyze feedback to identify areas where the system or personnel can improve.

**Data Storage:**

* Use text files (e.g., complaints.txt) to store complaints.
* Use structured formats for storing data (e.g., CSV or fixed-width records).

**Flow:**

* Main menu for user interaction.
* Sub-menus for various functionalities.

**Code Outline:**

1. **Struct Definition**
2. **Functions**

* Register Complaint
* View Complaints
* Update Complaint Status

1. **Main Menu**

**PROGRAM**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_COMPLAINTS 100

#define MAX\_LENGTH 256

typedef struct {

int id;

char customerName[MAX\_LENGTH];

char complaintType[MAX\_LENGTH];

char description[MAX\_LENGTH];

char status[MAX\_LENGTH];

char feedback[MAX\_LENGTH]; // New field for feedback

} Complaint;

void registerComplaint(Complaint complaints[], int \*count) {

printf("\n--- Register a Complaint ---\n");

if (\*count >= MAX\_COMPLAINTS) {

printf("Error: Complaint limit reached.\n");

return;

}

complaints[\*count].id = \*count + 1;

printf("Enter Customer Name: ");

scanf(" %[^\n]", complaints[\*count].customerName);

printf("Enter Complaint Type (Product/Service): ");

scanf(" %[^\n]", complaints[\*count].complaintType);

printf("Enter Complaint Description: ");

scanf(" %[^\n]", complaints[\*count].description);

strcpy(complaints[\*count].status, "Pending");

complaints[\*count].feedback[0] = '\0'; // Initialize feedback as empty

(\*count)++;

printf("Complaint Registered Successfully!\n");

}

void viewComplaints(Complaint complaints[], int count) {

printf("\n--- View All Complaints ---\n");

if (count == 0) {

printf("No complaints registered.\n");

return;

}

for (int i = 0; i < count; i++) {

printf("Complaint ID: %d\n", complaints[i].id);

printf("Customer Name: %s\n", complaints[i].customerName);

printf("Complaint Type: %s\n", complaints[i].complaintType);

printf("Description: %s\n", complaints[i].description);

printf("Status: %s\n", complaints[i].status);

if (strcmp(complaints[i].status, "Resolved") == 0) {

printf("Feedback: %s\n", complaints[i].feedback);

}

printf("-------------------------\n");

}

}

void updateStatus(Complaint complaints[], int count) {

int id;

printf("\n--- Update Complaint Status ---\n");

printf("Enter Complaint ID: ");

scanf("%d", &id);

for (int i = 0; i < count; i++) {

if (complaints[i].id == id) {

printf("Enter New Status (Resolved/In Progress): ");

scanf(" %[^\n]", complaints[i].status);

if (strcmp(complaints[i].status, "Resolved") == 0) {

printf("Enter Feedback for the Resolved Complaint: ");

scanf(" %[^\n]", complaints[i].feedback);

}

printf("Status Updated Successfully!\n");

return;

}

}

printf("Error: Complaint ID not found.\n");

}

int main() {

Complaint complaints[MAX\_COMPLAINTS];

int count = 0;

int choice;

while (1) {

printf("\n--- Consumer Complaint Handling System ---\n");

printf("1. Register Complaint\n");

printf("2. View Complaints\n");

printf("3. Update Complaint Status\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

registerComplaint(complaints, &count);

break;

case 2:

viewComplaints(complaints, count);

break;

case 3:

updateStatus(complaints, count);

break;

case 4:

printf("Exiting the system.\n");

exit(0);

default:

printf("Invalid choice. Please try again.\n");

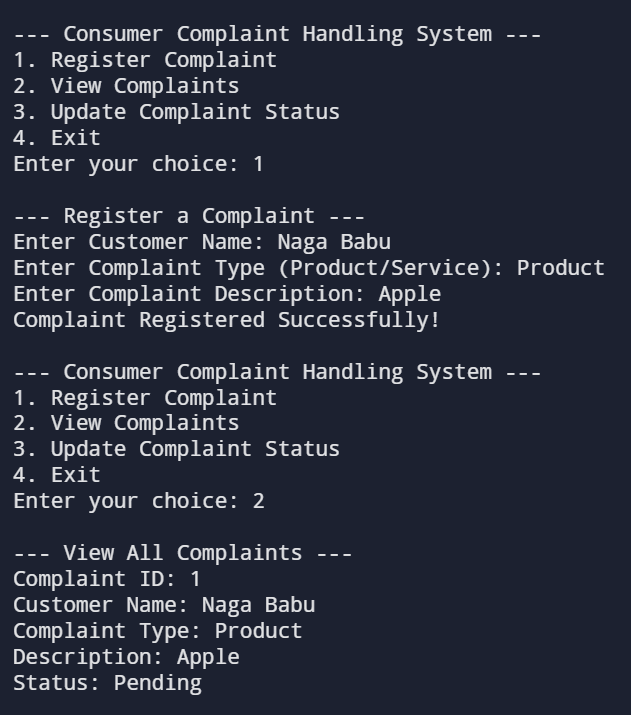
}

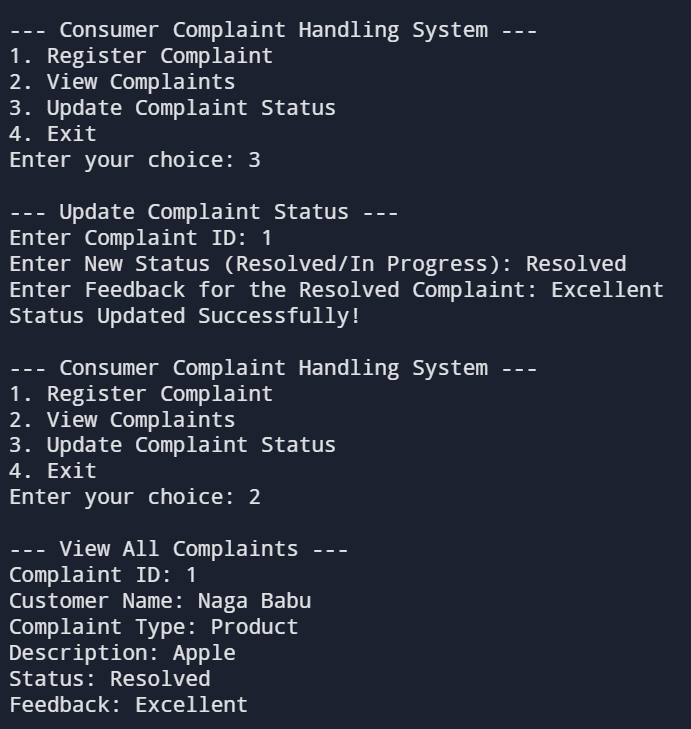
}

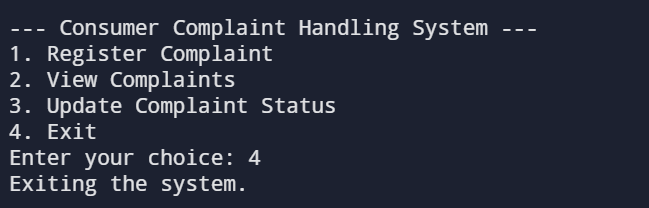
return 0;

}

**RESULT**







**ENGINEERING STANDARDS**

1. C Language Standards(ISO/IEC 9899)
2. Software Development Standards
3. Data Security and Privacy Standards
4. Usability and Accessibility Standards
5. System Performance Standards
6. Interoperability and Integration Standards
7. Ethical and Social Responsibility Standards
8. Documentation and Reporting Standards
9. Scalability and Future Readiness

Out of which, I’m interested to take Software Development Standards and C Language Standards as they provide a systematic framework to ensure the successful development of the Consumer Complaint Handling System. These standards guide the entire lifecycle of software creation, from planning and design to implementation, testing, and maintenance. They ensure that the software is reliable, functional, and user-friendly while meeting industry benchmarks for quality.

Key standards include ISO/IEC 12207, which outlines a structured approach to software development, ensuring every phase is well-organized and documented. IEEE 829 focuses on testing documentation to validate the software’s performance and identify any issues before deployment. Additionally, ISO/IEC 25010 emphasizes software quality, ensuring functionality, reliability, usability, and security.

By adhering to these standards, I aim to develop a system that not only meets its intended purpose of handling consumer complaints effectively but also upholds high-quality benchmarks, ensuring customer trust and satisfaction.

1. **C Language Standard**
2. **ISO/IEC 9899:1990 (C90 or ANSI C)**

* First standardization of the C language.
* Introduced features such as:
* Function prototypes.
* Standard library functions (e.g., stdio.h, stdlib.h).
* Improved type checking.
* Known as ANSI C when adopted by the American National Standards Institute (ANSI).

1. **ISO/IEC 9899:1995 (C95)**

* A minor revision of C90.
* Included:
* Normative corrections.
* Wide character support (wchar\_t, <wchar.h>, <wctype.h>).
* New library functions (e.g., wprintf, mbtowc).

1. **ISO/IEC 9899:1999 (C99)**

* Significant update to the language.
* New features:
* Inline functions (inline keyword).
* Variable-length arrays (VLAs).
* restrict keyword for pointer optimization.
* Improved support for floating-point operations.
* New standard headers: <stdbool.h>, <stdint.h>, <complex.h>.
* Enhanced support for single-line comments (//).
* Initial declarations inside for loops.

1. **ISO/IEC 9899:2011 (C11)**

* Focused on improving concurrency and safety.
* Added:
* Multi-threading support (e.g., <threads.h>, <atomic.h>).
* Anonymous structures and unions.
* \_Generic keyword for generic programming.
* Improved bounds-checking functions (e.g., <stdnoreturn.h>).
* Static assertions (\_Static\_assert).
* Made VLAs optional for compiler implementation.

1. **ISO/IEC 9899:2018 (C17 or C18)**

* Minor revision of C11.
* Addressed defect reports (technical corrections).
* No new language features or libraries introduced.

1. **ISO/IEC 9899:2023 (C23)**

* Latest revision focused on usability and modernization.
* Introduced:
* New keywords like typeof.
* Improved support for embedded systems.
* Added Unicode character literals and string prefixes (u8, u, U).
* Enhanced diagnostic capabilities.
* New library functions and macros for better standard compliance.

1. **Software Development Standards**
2. **ISO/IEC 12207:** Software Development Life Cycle (SDLC) Processes

* Defines a framework for software development processes, including planning, design, implementation, testing, deployment, and maintenance.
* Ensures that each stage of the project is systematically approached and thoroughly documented.

1. **IEEE 829:** Software Testing Documentation

* Establishes standards for creating test-related documents like test plans, test cases, and test reports.
* Ensures comprehensive testing to identify and address system errors or vulnerabilities.

1. **ISO/IEC 25010**: Software Quality Model

* Functionality: Ensures the system meets customer requirements, such as registering and tracking complaints.
* Reliability: The system should consistently handle a large volume of complaints without failures.
* Usability: Design an intuitive interface for ease of use by both customers and administrators.
* Performance Efficiency: Optimize the system for quick response times and low resource usage.
* Security: Protect sensitive consumer data from unauthorized access.
* Maintainability: Ensure the system can be easily updated or modified.
* Portability: Enable the system to run on various platforms.

1. **ISO/IEC 26514:** User Documentation

* Provides guidelines for creating user-friendly documentation for both system operators and consumers.
* Ensures that users can easily navigate and operate the system with minimal training.

1. **IEEE 730:** Software Quality Assurance

* Outlines the process for ensuring that the software meets its quality objectives.
* Includes reviews, audits, and evaluations to maintain high-quality standards throughout development.

**FUTURE SCOPE**

The Consumer Complaint Handling System developed in C programming lays the groundwork for a basic grievance management solution. However, its capabilities can be significantly enhanced to meet the demands of modern businesses and consumer expectations. Below are potential future advancements for the system:

1. **Database Integration**

* Replace file-based storage with robust database systems like MySQL or PostgreSQL to manage large volumes of complaints more efficiently.
* Enable faster data retrieval, better organization, and enhanced scalability.

1. **Web and Mobile Interfaces**

* Develop web-based and mobile-friendly interfaces to allow consumers to register complaints and track statuses conveniently.
* Use frameworks like PHP, Node.js, or React for web development and Android/iOS SDKs for mobile applications.

1. **Advanced Categorization**

* Implement machine learning algorithms for complaint classification to handle complex data and identify patterns in grievances.
* Incorporate NLP (Natural Language Processing) techniques for categorizing complaints based on textual inputs.

1. **Real-Time Tracking and Notifications**

* Introduce a notification system to keep customers informed about complaint progress through SMS, email, or push notifications.
* Allow real-time tracking of complaint status via a customer portal.

1. **Analytics and Reporting**

* Add analytics dashboards to generate insights into complaint trends, resolution times, and customer satisfaction.
* Use these insights to identify recurring issues and improve products or services.

1. **Multi-Language Support**

* Include multi-language support to cater to diverse customer bases and improve accessibility.

1. **Integration with Customer Relationship Management (CRM) Tools**

* Connect the complaint system to CRM tools like Salesforce or Zoho to centralize customer interactions and enhance service quality.

1. **AI-Driven Assistance**

* Integrate AI chatbots to handle frequently asked questions, guide customers through complaint registration, and provide instant resolutions for common issues.

1. **Feedback Analytics**

* Utilize sentiment analysis on customer feedback to measure satisfaction and identify areas of improvement.

1. **Data Security and Compliance**

* Implement advanced encryption techniques to ensure data security.
* Ensure compliance with data privacy laws like GDPR or CCPA for businesses operating in regions with stringent regulations.

**CONCLUSION**

The **Consumer Complaint Handling System** developed using C programming serves as a foundational application for managing customer grievances in an organized and efficient manner. By enabling the registration, categorization, tracking, and status updating of complaints, the system addresses key challenges in complaint handling, such as delayed responses and lack of transparency.

This project highlights the practical use of C programming for solving real-world problems, leveraging its file handling and structured programming capabilities. While the current system provides basic functionalities, it demonstrates the importance of automation in reducing manual effort, improving operational efficiency, and enhancing customer satisfaction.

The system's potential for future enhancements, including integration with databases, web interfaces, and advanced technologies like machine learning, ensures its scalability and adaptability to modern business needs. With further development, the application can evolve into a comprehensive platform that significantly contributes to building trust and loyalty among consumers.

In conclusion, this project not only showcases the feasibility of implementing a complaint management system in C but also underscores the critical role such systems play in maintaining a positive customer experience in today’s customer-centric industries.