

Logo

Abstract

The Complaint Management System is a C-based console application designed to simplify the process of registering, managing, and resolving complaints in a structured way. It provides role-based access for users, staff, and administrators, enabling complaint submission, staff assignment, and status tracking with data stored in text files. The system improves efficiency and accountability by digitizing the workflow, offering a simple yet effective solution that can later be extended into a more advanced platform with database or GUI support. The Complaint Management System (CMS) is a menu-driven program written in C that allows users to register complaints, track progress, and interact with staff and administrators in a structured workflow. The system provides a centralized platform where: Users can register and monitor complaints. Admins can approve complaints, assign staff, and manage users. Staff can handle assigned complaints and update status. This system ensures transparency, accountability, and efficient complaint handling, with persistent data storage through text files.

Variables and Main function Workflow overview

# **Variables in Complaint Management System**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Name** | **Type** | **Scope** | **Description / Purpose** |
| **Struct: User** | | | |
| username | char[30] | Member of User | Stores the username of a registered user |
| password | char[30] | Member of User | Stores the user’s password (plaintext in file) |
| contact | char[20] | Member of User | Stores phone/email contact details |
| **Struct: Complaint** | | | |
| complaintID | int | Member of Complaint | Unique ID for each complaint |
| category | char[30] | Member of Complaint | Complaint type/category (e.g., service, billing) |
| description | char[100] | Member of Complaint | Detailed description of the complaint |
| status | char[20] | Member of Complaint | Current status (Pending, In Progress, Resolved) |
| userID | int | Member of Complaint | ID of the user who submitted the complaint |
| **Struct: StaffEntry** | | | |
| staffID | int | Member of StaffEntry | Unique identifier for each staff |
| name | char[30] | Member of StaffEntry | Staff member’s name |
| assignedCategory | char[30] | Member of StaffEntry | Complaint category assigned to staff |
| **Global Variables** | | | |
| users[100] | User[] | Global | Array storing registered users |
| complaints[200] | Complaint[] | Global | Array storing complaints |
| staff[50] | StaffEntry[] | Global | Array storing staff records |
| userCount | int | Global | Tracks total number of registered users |
| complaintCount | int | Global | Tracks total complaints submitted |
| staffCount | int | Global | Tracks total staff added |
| **Local Variables (common in functions)** | | | |
| choice | int | Local | Stores menu choices |
| tempUsername | char[30] | Local | Temporarily holds username input during login |
| tempPassword | char[30] | Local | Temporarily holds password input during login |
| status | char[20] | Local | Used for updating complaint status |
| found | int | Local | Flag variable to check login/record existence |
| id | int | Local | Used for searching complaint/staff IDs |

### **Functions**

1. **mainDashboard()**
   * Displays the main dashboard menu and lets the user choose between Admin, User, or Staff login/registration.
2. **userDashboard()**
   * Handles all user-related operations such as submitting a complaint, viewing complaint status, and logging out.
3. **adminDashboard()**
   * Provides admin functionalities including approving complaints, assigning staff, managing users and staff records, and monitoring system activity.
4. **staffDashboard()**
   * Manages staff operations like checking assigned complaints, updating complaint statuses, and resolving issues.
5. **registerUser()**
   * Collects user details (username, password, contact info) and stores them in the user records file.
6. **loginUser()**
   * Handles user authentication by checking entered username and password against saved records.
7. **addStaff()**
   * Allows the admin to add new staff members with their name, ID, and assigned category.
8. **submitComplaint()**
   * Lets a logged-in user file a new complaint by entering category and description, then stores it in the complaints file.
9. **viewComplaints()**
   * Displays all submitted complaints with their details and current status.
10. **updateComplaintStatus()**

* Allows staff members to change the complaint’s status (e.g., Pending → In Progress → Resolved).

1. **assignComplaintToStaff()**

* Admin function that assigns a complaint to an available staff member based on category.

1. **saveData()**

* Writes updated users, staff, and complaints information to text files to ensure persistence.

1. **loadData()**

* Reads data from text files when the program starts, loading users, staff, and complaints into memory.

1. **exitSystem()**

* Cleanly exits the program, usually saving data before closing.

Limitations

Since the Complaint Management System is developed as a basic console-based C program, it naturally comes with several limitations in terms of scalability, usability, and security.

1. **No Database Integration**
   * The system stores data in simple text files. This makes it difficult to handle large data efficiently, and searching/updating can become slow.
2. **Weak Security**
   * User passwords are stored in plain text files, which makes the system vulnerable to data breaches and unauthorized access.
3. **Limited User Interface**
   * Being a console-based program, the system lacks a graphical user interface (GUI), making it less user-friendly compared to modern complaint systems.
4. **Single-Device Dependency**
   * The system works only on the computer where it is installed; it does not support online access or multi-user real-time usage.
5. **No Role Hierarchy Flexibility**
   * Roles are fixed as **Admin, User, and Staff**. The system cannot dynamically add or customize new roles without modifying the source code.
6. **Lack of Notifications**
   * Users are not notified automatically when their complaint status changes; they must manually check by logging in.
7. **Scalability Issues**
   * Since the system is file-based and limited in memory, it may not handle a very large number of users, staff, or complaints efficiently.
8. **Error Handling is Minimal**
   * The program may crash or misbehave if unexpected inputs (like special characters, wrong formats, or very long strings) are given.
9. **No Analytics or Reports**
   * The system cannot generate complaint trends, resolution rates, or performance statistics for admins.
10. **Platform Limitation**

* Since it is written in C and runs in a console environment, it does not directly support mobile or web platforms.

## **Technology Requirements**

**Programming Language**: C (ANSI C standard)

**Compiler**: GCC / Turbo C / MinGW

**IDE Support**: Code::Blocks, Dev-C++, Visual Studio Code

**Operating System**: Windows / Linux

## **Support**

Easy deployment on any system with a C compiler

Simple text file storage makes backup easy

Can be extended to **MySQL/SQLite database**

Code modularity allows further development into GUI/Web/Mobile apps

## **Requirements Elicitation**

### **1. Stakeholder Identification**

* **Admin** – Manages the system, assigns staff, and oversees complaints.
* **Users (Customers)** – Submit complaints and check their status.
* **Staff Members** – Resolve complaints assigned by the admin.
* **System Developers** – Design, implement, and maintain the software.

### **2. Elicitation Method**

The following methods were used to gather requirements:

* **Interviews** with potential users (to understand their needs).
* **Questionnaires/Surveys** to collect feedback about expected features.
* **Observation** of existing manual complaint-handling processes.
* **Document Review** of existing complaint registers and logs.

### **3. Requirements Gathering Topics**

Key areas discussed during requirements gathering included:

* **User Authentication** – Secure login for users, staff, and admin.
* **Complaint Submission** – Easy way for users to register complaints.
* **Complaint Tracking** – Users should be able to check the status of their complaint.
* **Staff Assignment** – Admin should assign complaints to staff.
* **Status Update** – Staff should update progress (Pending → In Progress → Resolved).
* **Data Storage** – All complaints, users, and staff details should be stored persistently.
* **Reporting Needs** – Basic overview of complaints for admin monitoring.

### **4. Analyzing and Prioritizing**

Requirements were analyzed and prioritized based on **importance and feasibility**:

* **High Priority:** User login, complaint submission, complaint tracking, staff assignment, data storage.
* **Medium Priority:** Staff details management, password recovery, complaint categorization.
* **Low Priority:** Graphical interface, automated notifications, performance analytics.

### **5. Documentation**

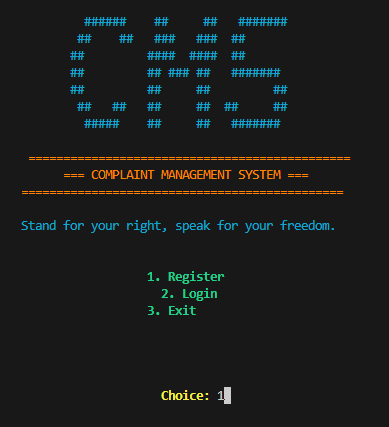
All requirements were documented in:

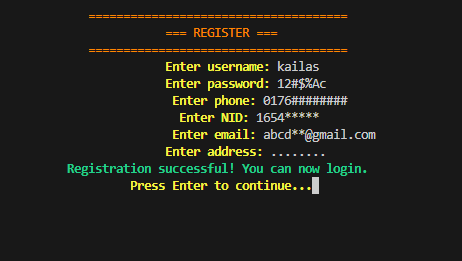
* **Use Case Diagrams & Descriptions** for system interactions.
* **Requirement Specification Document (SRS)** containing functional and non-functional requirements.
* **Data Flow Diagrams (DFD)** showing complaint flow through the system.

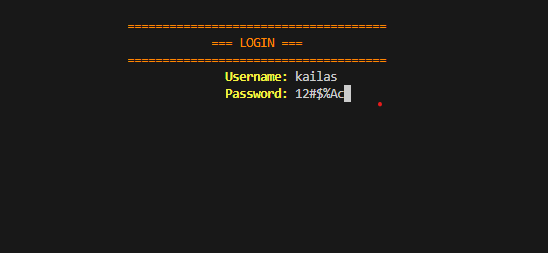
### **6. Review and Validation**

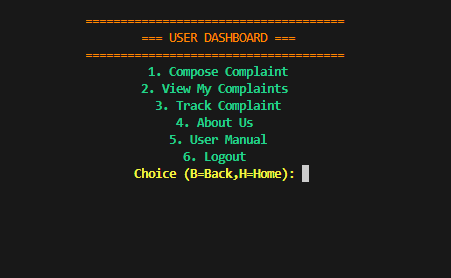
* The documented requirements were reviewed with stakeholders (users, staff, admin).
* Validation ensured that requirements matched stakeholder needs and system goals.
* Feedback was collected, and minor adjustments were made (e.g., simplifying login and making complaint tracking more user-friendly).

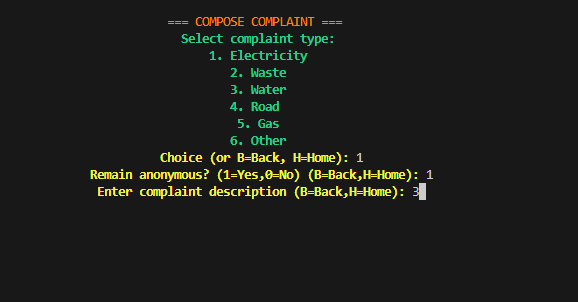
## **User Manual**

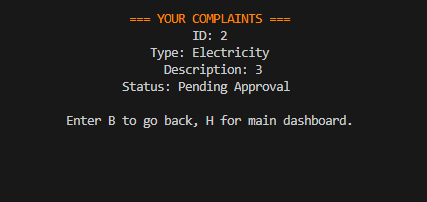


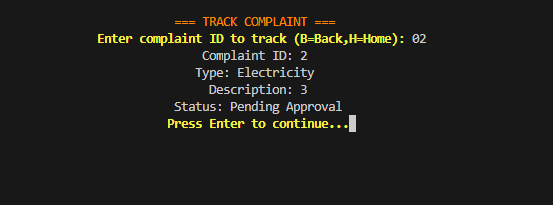


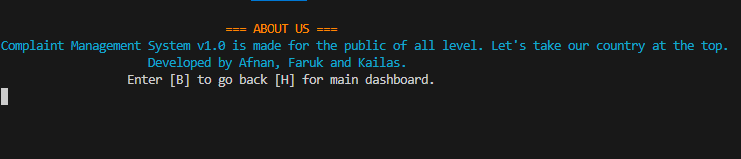


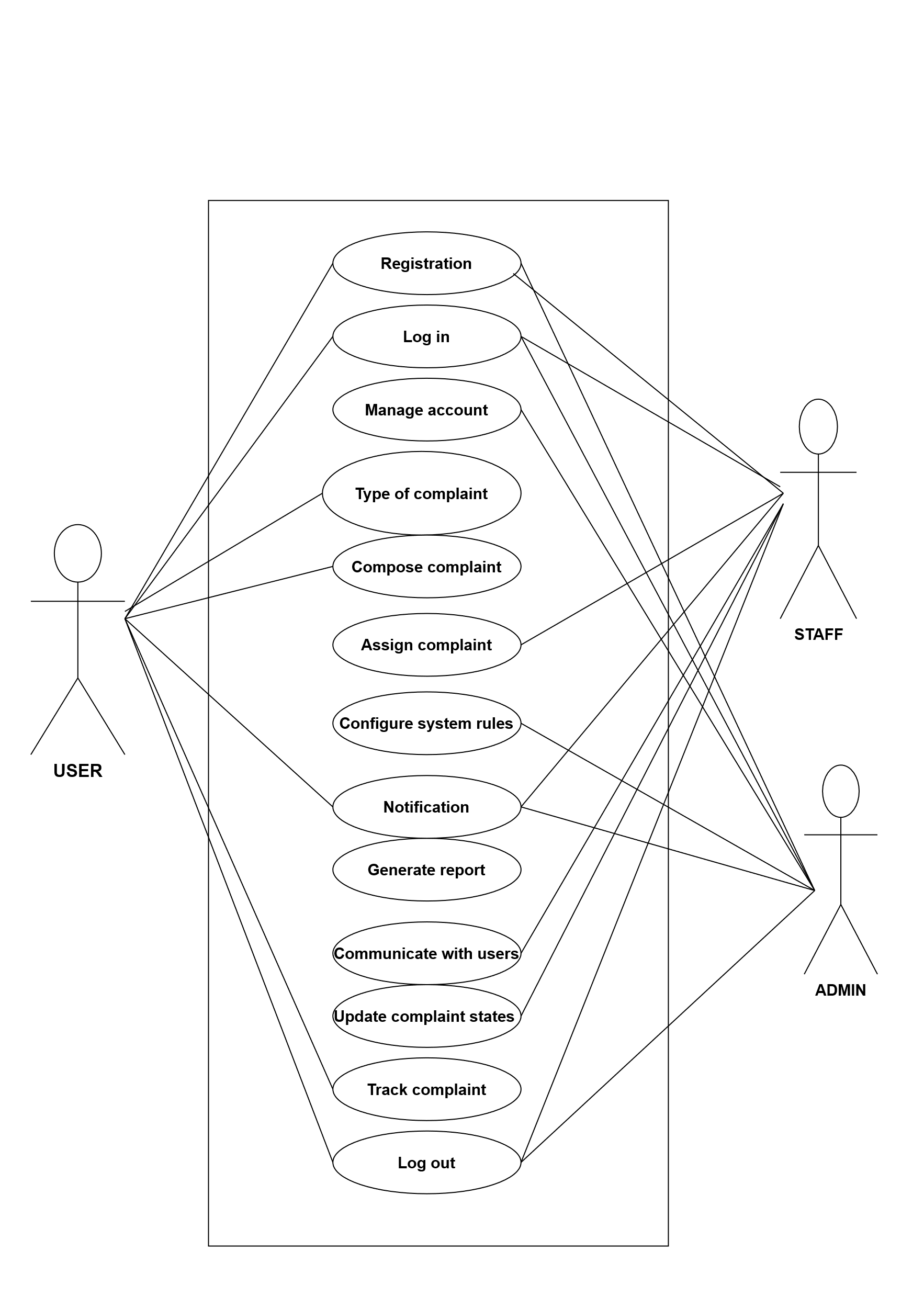










****

## **Use Case Diagram**

**EMPLOYEE**

* Registration

|  |  |  |
| --- | --- | --- |
| Use case name | **Registration** | |
| Goal | Create a new user account in the system with valid credentials. | |
| Precondition | 1. User is not logged in. 2. Registration feature is available. | |
| Success end condition | User account is created; confirmation message is displayed. | |
| Fail end condition | Registration fails; error message is displayed; no account is created. | |
| Primary actor | New User | |
| Secondary actor | System Administrator | |
| Triger | User selects "Register" from the login page. | |
| Description | Step | Action |
| 1.  2.  3.  4.  5.  6. | User navigates to the registration form.   Email, Password, Confirm Password, and Terms of Service checkbox.  User enters valid data in all fields and checks the *Terms of Service* box  User submits the form.  System creates a new account with a unique user ID.  System displays: *"*Account created! Check your email to verify*."* |
| Alternative flow | Step | Action |
| 5a  5b  5c  5d | System rejects submission; highlights the email field with an error.  System enforces rules shows real-time feedback.   System displays error: "Passwords do not match."  System prompts: "This email is already in use. Try logging in or resetting your password*."* |
| Quality requirement | Step | Requirement |
| 6.  7. | Performance: Account creation completes in ≤ 2 seconds.  Reliability: Success message persists even if the user refreshes the page. |

* Log in

|  |  |  |
| --- | --- | --- |
| Use case name | **User Login** | |
| Goal | Authenticate user and grant access to the system. | |
| Precondition | 1. User has a registered account. 2. User is not already logged in. | |
| Success end condition | User is authenticated; redirected to dashboard. | |
| Fail end condition | Authentication fails; access is denied; error message displayed. | |
| Primary actor | Registered User | |
| Secondary actor | Authentication Service | |
| Triger | User clicks "Login" on the homepage or app. | |
| Description | Step | Action |
| 1  2  3  4  5  6  7 | User navigates to login page.  System displays fields: Email/Username, Password, Remember Me option.  User enters credentials and submits form.  System validates input format (non-empty fields, valid email pattern).  System verifies credentials against database.  System creates authenticated session.  System redirects user to personal dashboard. |
| Alternative flow | Step | Action |
| 4a  5a  5b  6 | Validation fails: System highlights empty/invalid fields.  Invalid credentials: System shows "Incorrect email/password" error.  Account locked: System shows "Account locked - reset password" alert.  Session failure: System logs error; prompts to retry login. |
| Quality requirement | Step | Requirement |
| 4,5  6  7 | Performance: Authentication completes in <1.5 seconds.  Security: Session uses HTTPS-only cookies with 15-min idle timeout.  UX: Error message avoids revealing if email is registered. |

* Type of Complaint

|  |  |  |
| --- | --- | --- |
| Use case name | **Types of Complaint** | |
| Goal | To allow users to submit different types of complaints through the system. | |
| Precondition | The user must be logged into the complaint management system. | |
| Success end condition | The complaint is successfully submitted and categorized. | |
| Fail end condition | The complaint fails to submit due to missing information or system error. | |
| Primary actor | User (Customer) | |
| Secondary actor | Admin (Complaint handler) | |
| Triger | User encounters an issue and wants to submit a complaint. | |
| Description | Step | Action |
| 1  2  3  4  5  6  7  8 | User navigates to the "Submit Complaint" page.  System displays a form with complaint types  User selects the complaint type.   |  | | --- | | User fills out complaint details and submits the form. | | System validates the input.   |  | | --- | | System categorizes the complaint and stores it in the database. |   System notifies the relevant department/admin.  User receives confirmation message. | |
| Alternative flow | Step | Action |
| 5a  5b | If required fields are missing, system displays an error message.  User corrects the input and resubmits |
| Quality requirement | Step | Requirement |
| 1  2 | The complaint form should load within 2 seconds.  System must validate inputs   |  | | --- | |  |  |  | | --- | |  | |

* Compose Complaint

|  |  |  |
| --- | --- | --- |
| Use case name | **Compose Complaint** | |
| Goal | To enable users to compose and submit a complaint based on different categories such as service, behavior, maintenance, or billing. | |
| Precondition | User is logged into the complaint system. | |
| Success end condition | Complaint is composed, categorized, and saved/submitted successfully. | |
| Fail end condition | Complaint composition fails due to missing input, system error, or invalid category. | |
| Primary actor | User (Customer or Resident) | |
| Secondary actor | Admin or Support Staff | |
| Triger | User clicks on “Compose Complaint” to report an issue. | |
| Description | Step | Action |
| 1  2  3  4  5  6  7  8  9  10 | User logs into the complaint portal  User clicks on the “Compose Complaint” option.  System displays a form with complaint type options  User selects a complaint type.  User enters complaint subject and detailed message.  User attaches optional files or images  User clicks “Submit”.  System validates the input.  System saves and categorizes the complaint  User receives confirmation of successful submission. |
| Alternative flow | Step | Action |
| 8a  8b  9a | If required fields are empty, system shows a validation error.  User corrects the errors and resubmits the complaint.  If the system cannot save the complaint due to a server error, it notifies the user and logs the error. |
| Quality requirement | Step | Requirement |
| 3  4  7 | Complaint types should load dynamically from a central database.  Selection must allow only one valid complaint type.  Submit action must process within 5 seconds.. |

* Manage User Account

|  |  |  |
| --- | --- | --- |
| Use case name | **Manage User Account** | |
| Goal | To allow users to create, update, and delete their accounts. | |
| Precondition | User must access the system interface. | |
| Success end condition | User account is successfully created, updated, or deleted. | |
| Fail end condition | Action fails due to validation or system error. | |
| Primary actor  Secondary actor | User | |
| System Administrator | |
| Triger | User initiates account management actions via interface. | |
| Description | Step | Action |
| 1  2  3  4  5 | User selects "Manage Account" from the menu.  System displays account management options.  User selects to create, edit, or delete account.  System processes request and performs validation.  If valid, system saves changes and confirms action. |
| Alternative flow | Step | Action |
| 4a  4d  5a | If validation fails, system displays error message to user.  User corrects the input and resubmits the form.  If system error occurs, user is informed and advised to retry |
| Quality requirement | Step | Requirement |
| 1  2 | Response time must be under 2 seconds for account actions  Data should be encrypted and stored securely. |

|  |  |  |
| --- | --- | --- |
| Use case name | **Generate Report** | |
| Goal | To allow the admin to generate and download complaint-related reports. | |
| Precondition | Admin is logged into the system. | |
| Success end condition | Report is generated successfully and ready for download. | |
| Fail end condition | Report generation fails due to system error or missing data. | |
| Primary actor  Secondary actor | Admin | |
| Staff | |
| Triger | Admin selects "Generate Report" from the dashboard. | |
| Description | Step | Action |
| 1  2  3  4  5  6 | Admin logs into the system and navigates to the report section.  Admin selects the type of report (e.g., daily, weekly, monthly).  Admin applies filters (date range, status, category, etc.).  System processes the request and fetches relevant data from the database. .  System generates the report and displays a download or preview option.  Admin downloads or views the report. |
| Alternative flow | Step | Action |
| 4a  5a  6a | If data for selected filters is unavailable, system notifies "No records found."  If report generation fails, system displays an error message.  Admin can retry with different filters or cancel the action. |
| Quality requirement | Step | Requirement |
| 1  2 | Report must be generated within 3 seconds for datasets under 1000 entries.  Reports should be downloadable in PDF and Excel formats. |

* Generate Report

|  |  |  |
| --- | --- | --- |
| Use case name | **Submit Complaint** | |
| Goal | To allow a user to submit a complaint to the system. | |
| Precondition | User must be logged into the system. | |
| Success end condition | Complaint is successfully recorded in the system. | |
| Fail end condition | Complaint submission fails due to validation or system error. | |
| Primary actor | User | |
| Secondary actor | System | |
| Triger | User selects “Submit Complaint” from the menu/interface. | |
| Description | Step | Action |
| 1  2  3  4  5  6 | User logs into the system.  ser navigates to the "Submit Complaint" page.  User enters complaint details (category, subject, description, attachments).  User clicks the “Submit” button.  System validates the input data.  If valid, system stores the complaint in the database. |
| Alternative flow | Step | Action |
| 5a  6a | If validation fails (e.g., empty subject/description), system highlights errors.  If database connection fails, system shows "Unable to submit complaint. Try later." |
| Quality requirement | Step | Requirement |
| 5  6  7 | |  | | --- | | If the complaint is already resolved or closed, the system disables the withdrawal option. | | If user cancels at confirmation step, system returns to complaint view without changes. | | If database or system error occurs during withdrawal, system shows an error message. | |

* Submit Complaint

|  |  |  |
| --- | --- | --- |
| Use case name | **Withdraw Complaint** | |
| Goal | To allow a user to cancel or withdraw a previously submitted complaint. | |
| Precondition | User is logged into the system and has at least one active complaint. | |
| Success end condition | Complaint is marked as withdrawn and no longer processed. | |
| Fail end condition | Withdrawal fails due to system error or restriction (e.g., already resolved). | |
| Primary actor | User | |
| Secondary actor | System | |
| Trigger | User selects an active complaint and chooses the "Withdraw" option. | |
| Description | Step | Action |
| 1  2  3  4  5  6  7  8  9 | |  | | --- | | User logs into the system. | | User navigates to "My Complaints" section. | | User selects a complaint with status “Pending” or “In Progress.” | | User clicks on “Withdraw Complaint” option. | | System prompts for confirmation. | | User confirms the withdrawal. | | System updates the complaint status to “Withdrawn” and logs the action. | | System sends a confirmation message to the user. |   User attaches optional files or images |
| Alternative flow | Step | Action |
| 8a  8b  9a | If required fields are empty, system shows a validation error.  User corrects the errors and resubmits the complaint.  If the system cannot save the complaint due to a server error, it notifies the user and logs the error. |
| Quality requirement | Step | Requirement |
| 3  4  7  10 | Complaint types should load dynamically from a central database.  Selection must allow only one valid complaint type.  Submit action must process within 5 seconds.  User should get a unique complaint ID in the confirmation. |

* Withdraw Complaint
* Assign Complaint

|  |  |  |
| --- | --- | --- |
| Use case name | **Assign Complaint** | |
| Goal | To allow the admin to assign a complaint to a staff member for resolution. | |
| Precondition | Admin is logged in and there are unassigned complaints in the system. | |
| Success end condition | Complaint is assigned to a staff member successfully and notification is sent. | |
| Fail end condition | assignment fails due to no available staff or system error. | |
| Primary actor  Secondary actor | User | |
| System Administrator | |
| Triger | Admin selects an unassigned complaint from the complaint list | |
| Description | Step | Action |
| 1  2  3  4  5  6 | Admin logs into the system and navigates to the complaint management section.  Admin views the list of unassigned complaints.  Admin selects a complaint and chooses "Assign Complaint."  System displays a list of available staff members.  Admin selects a staff member and confirms the assignment.  System saves the assignment and sends a notification to the selected staff. |
| Alternative flow | Step | Action |
| 4a  5a | If no staff members are available, the system displays a message: "No staff available for assignment."  If system encounters an error during saving, show "Assignment failed. Please try again." |
| Quality requirement | Step | Requirement |
| 5  6  6 | The assignment action must be completed within 2 seconds.  Notification (email/SMS/in-app) must be sent to the assigned staff member immediately.  All assignment actions must be logged with a timestamp and admin ID for audit tracking. |

* Configure System Rules

|  |  |  |
| --- | --- | --- |
| Use case name | **Configure System Rules** | |
| Goal | To allow the admin to configure system-level rules, thresholds, and settings | |
| Precondition | Admin is logged into the system with configuration privileges. | |
| Success end condition | System rules are successfully saved and applied. | |
| Fail end condition | Configuration fails due to validation error or system failure. | |
| Primary actor  Secondary actor | Admin | |
| Staff, User. | |
| Triger | Admin selects "System Configuration" from the admin panel. | |
| Description | Step | Action |
| 1  2  3  4  5 | Admin logs in and navigates to the "System Configuration" section.  Admin selects the type of rule to configure (e.g., auto-escalation time, SLA).  Admin inputs or updates rule values.  System validates the input values.  System saves the configuration and confirms update success. |
| Alternative flow | Step | Action |
| 4a  5a  5b | If input is invalid (e.g., negative time values), system shows validation error.  If saving fails, system shows a technical error message and logs the failure.  Admin may retry or cancel the operation. |
| Quality requirement | Step | Requirement |
| 4  5  5 | Input validation must catch incorrect formats or unsupported values.  System must apply new rules immediately without requiring a restart.  Configuration changes must be audit-logged for traceability. |

* Update Complaint status

|  |  |  |
| --- | --- | --- |
| Use case name | **Update Complaint Status** | |
| Goal | To allow authorized personnel to update the status of a complaint in the system | |
| Precondition | * The complaint exists in the system. * The user is logged in and has permission to manage complaints. | |
| Success end condition | The complaint status is updated and stored successfully in the system. | |
| Fail end condition | The complaint status is not updated due to system error or invalid input. | |
| Primary actor | Admin / Officer | |
| Secondary actor | System | |
| Triger | Admin selects a complaint and initiates a status update | |
| Description | Step | Action |
| 1  2  3  4  5  6 | Staff navigates to the complaint management section.  Staff selects a specific complaint from the list.  Staff selects a new status  Staff confirms the update.  System validates and updates the complaint status in the database.  System sends a notification to the complainant about the status change. |
| Alternative flow | Step | Action |
| 5a  5b | If the update fails due to a validation error or server issue, system displays an error message.  Staff corrects the issue and retries the update. |
| Quality requirement | Step | Requirement |
| 1  2  3 | Complaint status update must reflect in the system immediately (within 2 sec)  Only authorized users can update the complaint status.  All updates must be logged with timestamp and user ID. |

* Log out

|  |  |  |
| --- | --- | --- |
| Use case name | **Logout** | |
| Goal | To allow the user to securely exit the system and end the current session. | |
| Precondition | The user is logged into the system. | |
| Success end condition | The user is logged out and redirected to the login or home page. | |
| Fail end condition | The logout process fails and the session remains active. | |
| Primary actor | User | |
| Secondary actor | System | |
| Triger | The user clicks the "Logout" button or selects "Logout" from the menu. | |
| Description | Step | Action |
| 1  2  3  4 | User initiates logout by clicking the "Logout" button.  System processes the logout request.  System clears user session data from server and client side.  System redirects user to the login or home screen. |
| Alternative flow | Step | Action |
| 2a  3a  4a | Session timeout or network error occurs during logout.  System displays an error message and retries logout.  If retry fails, system forces session termination on next login. |
| Quality requirement | Step | Requirement |
| 1  2  3 | Logout action must be completed within 2 seconds.  All session tokens and sensitive data must be cleared securely.  User should receive a confirmation that they are logged out. |

The Complaint Management System provides an organized and efficient way to handle complaints, ensuring transparency and accountability for users, staff, and administrators. By centralizing complaint submission, tracking, and resolution, it reduces manual errors, saves time, and improves communication between stakeholders. The system demonstrates how technology can streamline administrative processes, enhance user satisfaction, and support better decision-making, making it a valuable tool for any organization managing multiple complaints.

Conclusion