CHAPTER 1
INTRODUCTION

1.1 Problem Definition

In a modern IT-enabled organization, frequent technical issues such as hardware failures, software bugs, network connectivity problems, and system performance concerns are common. These issues, if not addressed promptly, can lead to serious disruptions in daily operations, impacting both productivity and employee satisfaction. Traditionally, such complaints are handled manually or through unorganized communication channels such as emails or verbal reports, which often leads to mismanagement, delays in resolution, lack of accountability, and no tracking mechanism for raised issues.
The absence of a centralized system for handling complaints often results in complaints being overlooked, misprioritized, or resolved inefficiently. Moreover, users do not have a proper way to track the status of their complaints, and administrators face difficulty in managing and filtering complaints by priority or type.
To address these challenges, a structured **Complaint Management System** is required that provides users with a reliable and user-friendly platform to submit complaints while empowering administrators to manage and resolve them efficiently.

1.2 Project Overview/Specification

The **Complaint Management System (CMS)** is a web-based single-page application developed entirely using HTML, CSS, and JavaScript. It is designed to provide a streamlined and user-friendly platform for managing complaints within an IT office or any organizational setting. The primary objective of this system is to replace inefficient manual processes with a modern, digital interface that facilitates both complaint submission by users and effective complaint handling by administrators. Even though the system is frontend-based and does not include a backend or database integration, it efficiently simulates real-world functionality using JavaScript's dynamic capabilities and DOM manipulation.

The application begins with a comprehensive dashboard that offers a real-time visual representation of all complaints, categorized into statuses such as Pending, In Progress, Resolved, and Total Complaints. This helps administrators quickly understand the workload and take necessary actions. Users can submit complaints through a dynamic form that incorporates live character counters, input validation, and tooltips to guide the user while filling in the necessary details. Once a complaint is submitted, the system uses the jsPDF library to automatically generate a formal PDF receipt, which includes all relevant information such as complaint ID, issue type, date, and description. This allows users to keep a personal record of their complaints for future reference or follow-up.

Administrators can securely log in through a simulated login panel to gain access to additional functionalities. After successful authentication, they are able to view the list of complaints, filter them based on status or issue type, and take administrative actions such as updating the status of a complaint or deleting it altogether. The system also allows simulated management of admin users, including adding new credentials. Despite having no backend, the application maintains state efficiently through in-memory data structures and JavaScript arrays.

In terms of user experience, the entire application is designed to be responsive and mobile-friendly, ensuring smooth performance on desktops, tablets, and smartphones alike. This responsiveness is achieved using modern CSS techniques such as flexbox and grid layout, along with media queries for adapting to various screen sizes. Overall, the CMS project not only demonstrates the practical application of frontend web technologies but also reflects a real-world use case where digital transformation can improve organizational processes. It showcases proficiency in frontend development, client-side scripting, UI/UX design, and the ability to create document automation features without server-side support.

The application offers the following major functionalities:

- **Dashboard Overview**: Displays visual statistics for Pending, In-Progress, Resolved, and Total complaints.
- **New Complaint Form**: A form with validation, tooltips, and real-time character count.
- **PDF Export**: Automatic generation of a complaint receipt in PDF format using jsPDF.
- **Admin Login Panel**: Secure login for admin users with the ability to manage complaints and admin users.
- **Filter Options**: Ability to filter complaints based on status and issue type.
- **Responsive Design**: Fully responsive layout suitable for desktops, tablets, and smartphones.

1.3 Hardware Specification

To run the Complaint Management System effectively, the following hardware components are recommended. While the system is lightweight and runs entirely on the client side, an appropriate configuration ensures optimal performance and a smooth user experience.

- **Processor:** A system equipped with an **Intel Core i3 processor or higher**, or an equivalent AMD processor, is recommended. This ensures that the browser can efficiently render the dynamic content and perform real-time JavaScript operations without lag or delay.
- **RAM:** The application requires at least **4 GB of RAM** to function reliably. However, **8 GB or more is recommended** for users who may be multitasking with multiple browser tabs, developer tools, or running other applications in the background.
- **Hard Disk:** The application is lightweight in terms of storage. Around **500 MB of free disk space** is sufficient to accommodate browser cache, locally stored complaint data, and externally fetched assets such as fonts and icons.
- **Display:** A **minimum screen resolution of 1024×768 pixels** is recommended. This ensures that all interface elements are clearly visible and accessible, providing a user-friendly layout across various devices, including desktops and laptops.
- **Input Devices:** A **keyboard and mouse** are essential for interacting with the application, particularly when filling out forms, using the admin panel, or navigating dashboard controls.
- **Internet Connection:** An **internet connection is required only once** for loading external resources such as **Google Fonts** and **Font Awesome icons**. These assets are cached by the browser and enable **offline usage** in subsequent sessions, making the application accessible without a constant internet connection.

1.4 Software Specification

The software specifications are categorized into front-end technologies and external tools/libraries:

1.4.1 Front-End Technologies Used

- **HTML5**: Provides the basic structure and semantic layout for the web pages.
- **CSS3**: Used for styling the components and making the interface visually appealing and responsive.
- **JavaScript (ES6)**: Implements all functionalities including form validation, data storage, filtering, PDF generation, user authentication, and dashboard interactivity.

1.4.2 Tools, Libraries, and APIs

To enhance the functionality and user experience of the Complaint Management System (CMS), several external tools, libraries, and browser APIs have been integrated into the project. These components play a vital role in improving both the visual appeal and technical performance of the application while ensuring that it remains fully frontend-based and does not rely on server-side infrastructure.

One of the key tools used is Font Awesome, a widely adopted icon library that allows developers to integrate scalable vector icons into web applications. In this project, Font Awesome enriches the user interface by providing meaningful and context-specific icons for actions such as submitting a complaint, filtering results, editing entries, deleting records, and navigating the dashboard. The use of icons not only improves the visual hierarchy but also enhances usability and makes the application more intuitive for users.

For typography, the project incorporates Google Fonts, specifically the "Inter" font family, which offers a clean and modern typeface suited for web applications. The consistent use of this font throughout the interface improves readability and contributes to a professional, polished look. By embedding fonts from Google Fonts, the application benefits from better text rendering across different browsers and devices without increasing the project's file size.

Another significant integration is the jsPDF library, a powerful open-source JavaScript tool that enables client-side generation of PDF documents from HTML content. In the context of the CMS, this library is used to automatically generate and allow the download of complaint receipts once a user submits a complaint. The dynamically generated PDF contains all necessary complaint details including ID, date, issue type, and description, providing users with a formal acknowledgment of their submission that can be stored or printed for future reference.

To support data persistence without the need for a backend or database, the system makes use of localStorage, a part of the Web Storage API provided by modern browsers. This API allows the application to store complaint data, admin credentials, and other necessary information directly in the user's browser. Data saved in localStorage remains available across browser sessions, enabling users and administrators to interact with previously stored information even after the page is refreshed or the browser is closed and reopened. This approach ensures data longevity and functionality without requiring server interaction, keeping the application lightweight and fully client-side.

Collectively, these tools and APIs contribute significantly to the CMS's interactivity, appearance, and standalone capabilities, allowing it to perform complex tasks such as PDF generation, data persistence, and UI enhancement with minimal external dependencies.

1.4.3 Environment

The Complaint Management System is entirely browser-based and designed for execution in a client-side environment, meaning it requires no server, database, or internet connection after initial loading. It has been thoroughly tested for compatibility with all modern web browsers, including Google Chrome (recommended for best performance), Microsoft Edge, Mozilla Firefox, and Apple Safari. The system runs efficiently across these platforms, taking advantage of HTML5 and ES6 features fully supported by these browsers. For development and customization, the project is compatible with any modern text editor, with Visual Studio Code being the preferred environment due to its extensive support for web development through extensions, syntax highlighting, and real-time preview capabilities. Since no backend is involved, all code execution and data handling occur locally in the user's browser, making deployment and testing straightforward and hassle-free. This environment setup aligns well with the project's objective of simulating a real-world complaint tracking solution using only frontend technologies.

CHAPTER: 2
DESCRIPTION OF COMPANY

2.1 About the Organisation

Coders Hire Private Limited is a dynamic and modern staffing and recruitment service provider, headquartered in **Guna, Madhya Pradesh, India**. Established in August 2021 but with roots tracing back to 2019 under its original name "Code and Hunt Mafia", Coders Hire has rapidly distinguished itself as a reliable partner for organizations seeking top-tier technical talent on a global scale.

Figure 1: Coders Hire Trademark

Focused primarily on IT recruitment, the company operates across diverse sectors and geographies, including Europe, APAC, EMEA, AMEX, and the Indian domestic market. Coders Hire is recognized not just for their expansive reach but also for their expertise in SAP sourcing, making them a go-to firm for roles demanding specialized SAP and IT skills. Over the past few years, Coders Hire has successfully placed more than 200 candidates while maintaining impressively low candidate backout ratios, a testament to their robust sourcing procedures and candidate engagement strategies.

The organization is privately held, limited by shares, and remains unlisted on stock exchanges. It is currently helmed by Directors **Pulkit Saxena** (CEO) and **Prafulla Saxena** (Senior Executive Human Recruiter). Coders Hire's commitment to quality is evident in their operational philosophy: to provide the best candidates at competitive rates, with quick turnaround times and unmatched reliability. Their client-centric approach is supported by a skilled team of recruiters and specialists, all striving to build lasting partnerships and meet evolving workforce demands.

Beyond recruitment and staffing, Coders Hire offers digital marketing services, SAP application development and support, as well as comprehensive website development solutions. Their holistic approach positions them as more than just a recruitment agency, but as a valued business partner for their clients' long-term growth and digital transformation needs

2.2 Work Culture

At the heart of Coders Hire’s success lies an intentionally cultivated work culture that distinguishes the organization in a highly competitive recruitment industry. The leadership at Coders Hire recognizes that sustained excellence and adaptability depend on a team-oriented environment where contributions from all levels are valued. To this end, the company consciously nurtures an atmosphere defined by inclusivity, support, and continuous learning. Employees are encouraged not only to perform their designated roles but also to voice novel ideas, suggest improvements, and innovate within their domains, fostering a true sense of shared ownership in the company's mission and outcomes.

The culture promotes cross-functional collaboration, with regular team meetings, brainstorming sessions, and knowledge-sharing initiatives that break down silos. This open exchange of insights enables recruiters, developers, and business specialists to stay abreast of market trends, candidate expectations, and technical advancements, translating directly into better client service and results. Management’s accessibility means suggestions and grievances are addressed promptly, while transparent internal communications ensure everyone is aligned with strategic objectives and impending opportunities.

Coders Hire strikes a deliberate balance between autonomy and guidance. Employees are entrusted with significant responsibility within their roles, empowering them to take initiative and make decisions confidently. At the same time, team leads and senior managers offer structured mentorship, helping newer recruits ramp up and seasoned staff develop advanced capabilities. Employee development is prioritized through well-defined training modules, webinars on emerging technologies, and support for relevant certifications, a critical proposition in the ever-evolving IT and SAP recruitment landscape.

Recognition and well-being are cornerstone values. The company celebrates successes and milestones, be it achieving recruitment targets, positive client feedback, or personal employee achievements, through formal appreciation, incentives, and team gatherings. To prevent burnout and promote healthy work-life integration, workloads are distributed judiciously with attention to individual capacity and preferences. Management regularly solicits feedback and adapts workflows to optimize productivity while safeguarding morale. Flexible working arrangements, especially post-pandemic, have been adopted to accommodate employees’ unique needs and circumstances.

The organizational ethos is centered on integrity, accountability, and client obsession. Every employee, regardless of rank, is instilled with the importance of ethical practices, timely delivery, and high-quality output. Team camaraderie extends beyond the workplace as the company encourages social interactions, celebrating cultural events and fostering a sense of community. The synergy between Coders Hire’s professional, empowering, and empathetic culture is evidenced by high retention rates, enduring employee loyalty, and a positive reputation both within the team and among clients and candidates.

In essence, Coders Hire's culture is more than a set of workplace norms; it is a critical enabler of the company's vision, empowering individuals and teams alike to excel, innovate, and adapt as the organization continues to grow on the global stage.

2.3 Management Hierarchy

Coders Hire operates with a streamlined yet effective management structure, fitting for an agile organization involved in recruitment and technology services. The company's core leadership comprises its Directors:

- **Pulkit Saxena** (Director & CEO): Responsible for overall organizational vision, business growth, and operational leadership. Pulkit sets strategic goals, oversees major client engagements, and directs corporate functions.
- **Prafulla Saxena** (Director & Senior Executive Human Recruiter): Handles talent acquisition strategy, key client accounts, and supervises the recruitment team to ensure the highest standards of quality and reliability in candidate placements.

Supporting the Directors, Coders Hire employs several key roles within its talent management and business development divisions:

- **Senior Executive Human Recruiters and Talent Acquisition Specialists**: Professionals like Nainy Bajaj and Ishita Shrivastava play instrumental roles in project staffing, candidate evaluation, client coordination, and ensuring timely, high-quality placements.
- **Business Development Specialists**: Charged with expanding the firm's portfolio, they identify new opportunities, nurture client relationships, and drive business proposition pitches.
- **Operational and Support Staff**: These individuals handle administrative, IT, marketing, and communications functions, ensuring seamless back-end operations.

The management style at Coders Hire is participative, allowing for bottom-up feedback and innovation. Team leads and mid-level managers provide mentorship, ensuring that even the most junior recruits are equipped to contribute meaningfully to projects. This loose but well-defined hierarchy supports the company's mission for both quality and adaptability, keeping decision-making swift and operations agile as the organization continues to grow.

In summary, Coders Hire Private Limited embodies a dynamic fusion of expertise, innovation, and adaptability that distinguishes it within the competitive world of technical recruitment and digital solutions. Driven by a visionary leadership team, the organization has rapidly grown its presence across international markets and established a robust reputation for delivering specialized SAP and IT talent. Their impressive track record in candidate placements, coupled with stringent sourcing standards and client-centric engagement, has enabled the firm to cultivate lasting partnerships while consistently meeting client expectations for quality and turnaround.

Integral to this success is Coders Hire's collaborative and empowering work culture, where learning, idea-sharing, and transparent communication are deeply embedded values. Employees are entrusted with autonomy but benefit from supportive mentorship and a well-defined structure that enables both personal and collective growth. The company's proactive approach to upskilling, recognition, and cross-functional teamwork fosters high retention, sustained motivation, and optimal performance across all departments.

The firm's agile management hierarchy completes its foundation for ongoing success. By promoting participative decision-making and valuing input from team members at all levels, Coders Hire not only sustains operational efficiency but also encourages innovation and flexibility in response to changing client and industry demands. Collectively, these factors position Coders Hire Private Limited as a forward-thinking, client-focused, and resilient organization—well-equipped to thrive and lead in the evolving global landscape of technical staffing and digital transformation services.

CHAPTER 3
DESCRIPTION OF WORK

3.1 Existing System

In many traditional IT-enabled organizations, internal complaints or service requests are generally handled using informal, unstructured methods. These include physical registers, spreadsheets, verbal communication, emails, or basic messaging platforms. While these channels may seem convenient at first, they are prone to several operational limitations:

- **Lack of Centralization**: Complaints are scattered across different mediums, making it difficult to monitor or analyze.
- **No Tracking Mechanism**: Employees do not have the ability to check the status or progress of their complaints.
- **Unstructured Data**: Unformatted emails and verbal communication lack consistency and clarity.
- **Manual Prioritization**: IT support teams have to go through emails or messages manually to prioritize tasks.
- **No Escalation Workflow**: There is no built-in logic to escalate complaints based on urgency or pending status.
- **No Reporting or Analytics**: Managers cannot generate statistics or reports to assess performance or recurring issues.
- **Time-Consuming Resolution**: Due to the unorganized nature, time is wasted in identifying, assigning, and resolving issues.

These limitations often result in **delayed responses**, **user dissatisfaction**, and **increased workload** on IT personnel without clear accountability.

3.2 Proposed System

The proposed **Complaint Management System** aims to address the above shortcomings by introducing an interactive, responsive, and efficient web-based platform. It allows users to submit complaints and administrators to manage them effectively — all from within a browser, without requiring backend or server setup.

Features and Modules:

1. **Complaint Submission Form:**

   - A well-structured, user-friendly form capturing full name, email, department, issue type, priority, date, and detailed description.
   - Field validation, real-time error messages, tooltips, and character count to enhance usability.

2. **Dashboard Module:**
   - Displays key metrics: Pending Complaints, In Progress, Resolved, and Total.
   - Shows recently submitted complaints.
   - Visual status indicator for system health (based on complaint volume).

3. **Admin Login & Authentication:**
   - Login panel for system administrators using default credentials.
   - Admins can be added dynamically through the system.

4. **Admin Panel:**
   - Admins can view all complaints in tabular form.
   - Ability to filter by complaint status and type.
   - Features to update complaint status (Pending, In Progress, Resolved), delete complaints, and view full details.

5. **PDF Generation:**
   - On successful complaint submission, a downloadable complaint receipt is generated using **jsPDF**.
   - Contains all complaint details with a unique complaint ID.

6. **Data Storage:**
   - Uses browser's localStorage for persistence.
   - Data remains saved even after refreshing or closing the browser window.

7. **Mobile Responsiveness:**
   - Optimized for both desktop and mobile devices.
   - Sidebar toggle functionality and adaptive layout ensure usability on all screen sizes.

Benefits Over Existing System:

• Structured and Formalized Complaint Handling:

The CMS introduces a standardized process for complaint intake and resolution, replacing informal, inconsistent, or paper-based systems. Every complaint is logged, categorized, and assigned a unique ID, ensuring a systematic and trackable workflow.

• Transparent Workflow with Accountability:

With clear visibility into complaint statuses and administrative actions, the system fosters transparency and accountability. Users can track their complaint progress, while admins have access to logs of all submitted issues, reducing miscommunication and oversight.

• Improved Admin Response Time:

Real-time dashboard statistics and filtering tools help administrators prioritize and respond to complaints faster. The structured panel layout and visual indicators reduce the time needed to locate, interpret, and act on each complaint.

• Easy Status Tracking by Users:

The PDF receipt issued upon submission and the consistent tracking system enable users to stay informed about their complaint's progress. This transparency reduces the need for repeated follow-ups or manual inquiries.

• Digital Records Eliminate the Need for Paperwork:

By digitizing the entire complaint handling process, the CMS removes the dependency on manual records and printed forms. All data is stored electronically in the browser, making the system eco-friendly, efficient, and easy to maintain.

3.3 Feasibility Study

Before implementing any new system, it's essential to evaluate its practicality and long-term impact. The feasibility study for the Complaint Management System has been conducted in the following areas:

a. Technical Feasibility

- The system is entirely built with **client-side technologies**: HTML, CSS, and JavaScript.
- Compatible with all modern browsers (Chrome, Firefox, Edge, Safari).
- No server, hosting, or database setup is required, reducing complexity.
- Lightweight and efficient in performance, even on low-end devices.
- jsPDF integration for offline PDF export is proven and tested.

b. Operational Feasibility

- The system is simple and intuitive — no prior training required.
- Complaints can be submitted in less than 2 minutes.
- Admins can manage records with minimal effort.
- Can be scaled up with minimal changes (e.g., adding backend later if needed).
- Can be installed and used within intranet settings.

c. Economic Feasibility

- No software licenses or hosting costs — all libraries used are open-source.
- Development cost is minimal (suitable for student/intern/fresher projects).
- Saves future costs by reducing manual labor, communication delays, and IT overhead.

d. Legal and Security Feasibility

- No sensitive data is stored on external servers — everything remains within the browser.
- No personal information is shared with third-party services.
- Since the system is local, it complies with internal data privacy norms.
- PDF receipts are automatically generated and downloadable without cloud processing.

e. Environmental Feasibility

- Promotes **paperless workflow** by digitally storing complaint records and exporting them as PDFs.
- Reduces the use of printed registers, manual forms, and physical storage, contributing to sustainability goals.

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 Requirement Specification

The Complaint Management System was developed after careful analysis of the existing problems in manual complaint handling processes. To address the challenges effectively, both **functional** and **non-functional requirements** were defined.

Functional Requirements

The system must provide the following essential features to both users and administrators:

- Users must be able to submit complaints using a structured form.
- The system should validate all form fields before submission.
- Each complaint must be assigned a unique complaint ID.
- Users should get a confirmation along with a downloadable PDF.
- Admin users should log in with secure credentials.
- Admin should be able to:
    - View all complaints in a tabular format.
    - Filter complaints by type and status.
    - Edit the complaint status.
    - Delete complaints as needed.
    - Add new admin users for shared access.

Non-Functional Requirements

Apart from basic operations, the system should maintain performance, usability, and reliability:

- The system must be mobile-responsive and browser-compatible.
- Data must persist using browser localStorage even after closing the tab.
- Interface should be intuitive and user-friendly.
- The application should perform efficiently on low-end systems.
- It must not require a backend server or internet after initial loading.

4.2 Flowcharts / DFDs / ERDs

To understand system behavior and data movement, flowcharts and diagrams were used to model logic and structure.

Flowchart: Complaint Submission Process

This flowchart visualizes the process from user input to complaint storage and confirmation.

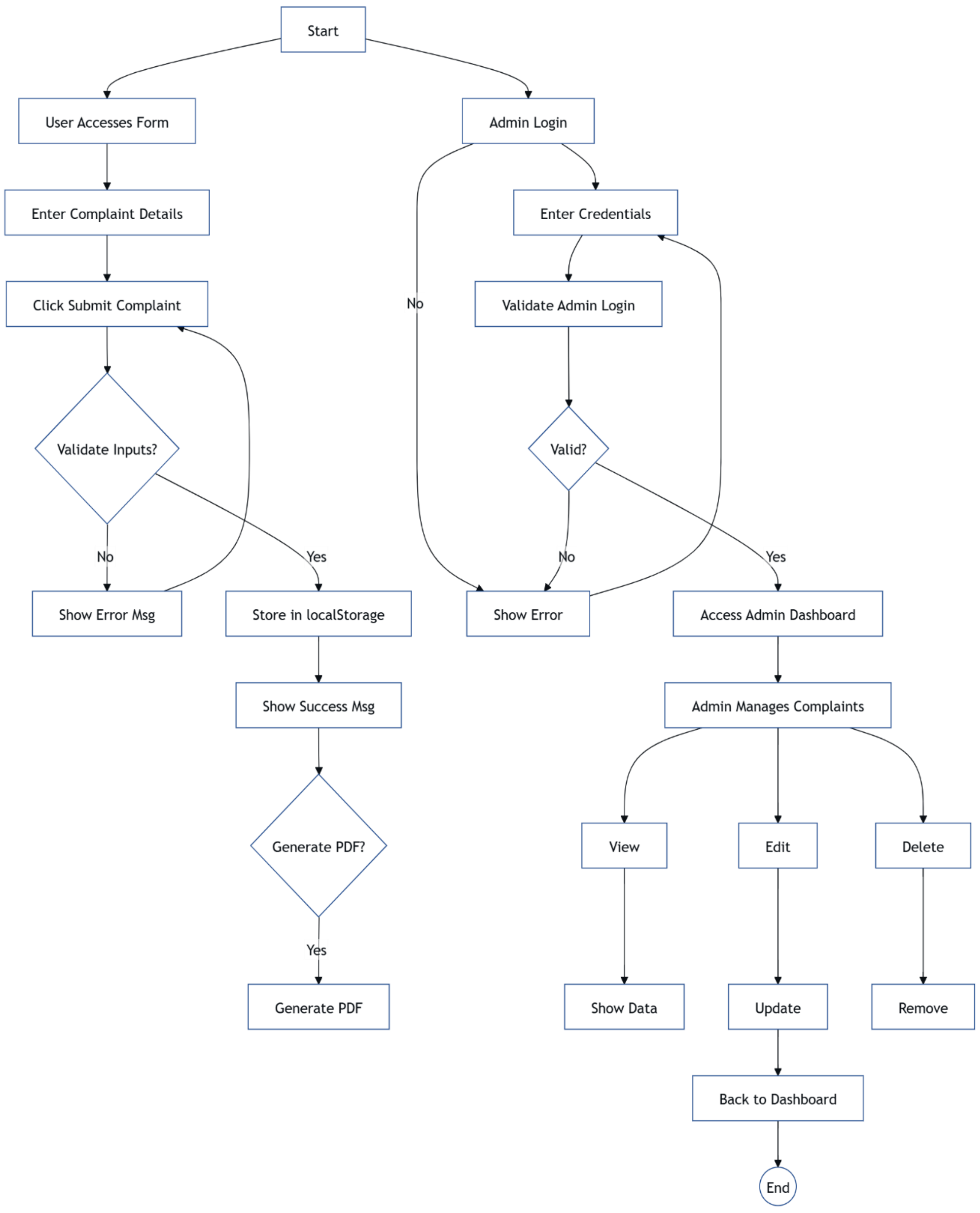


Figure 2: Flowchart

Data Flow Diagram (DFD) – Level 0

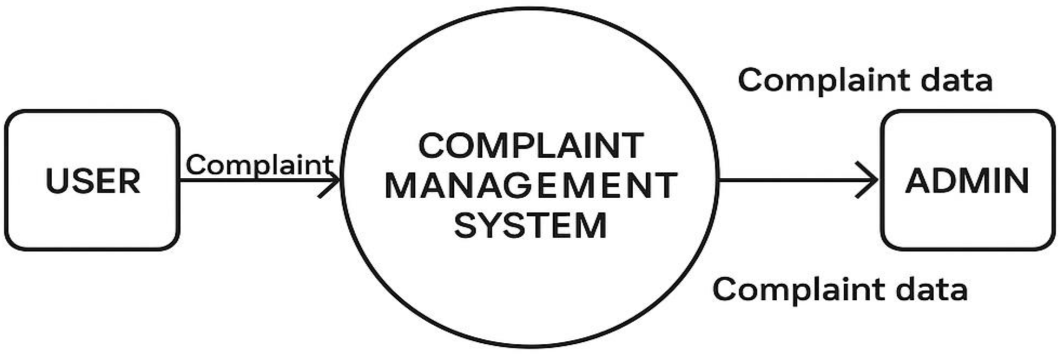


Figure 3: Data Flow Diagram – Level 0

DFD – Level 1

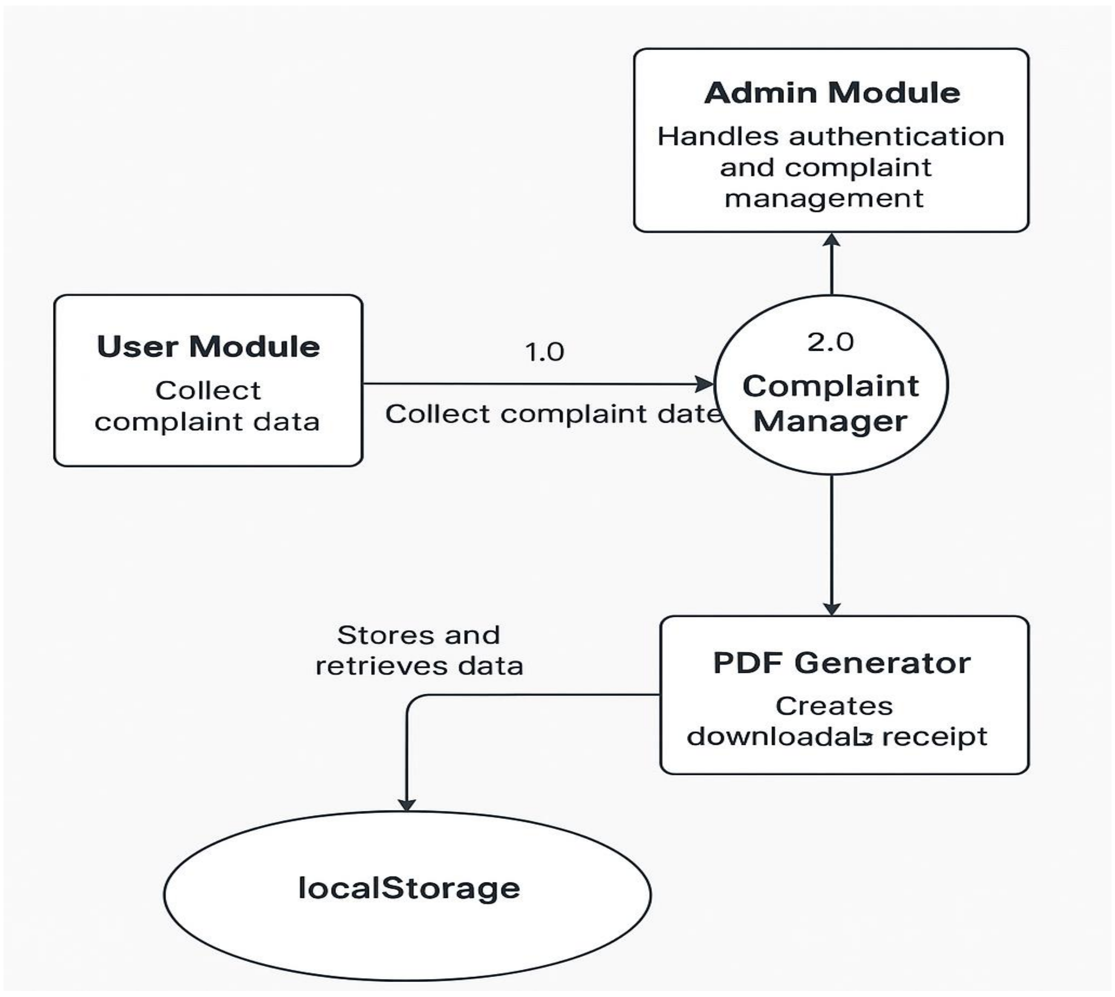


Figure 4: Data Flow Diagram – Level 1

Entity Relationship Diagram (ERD)

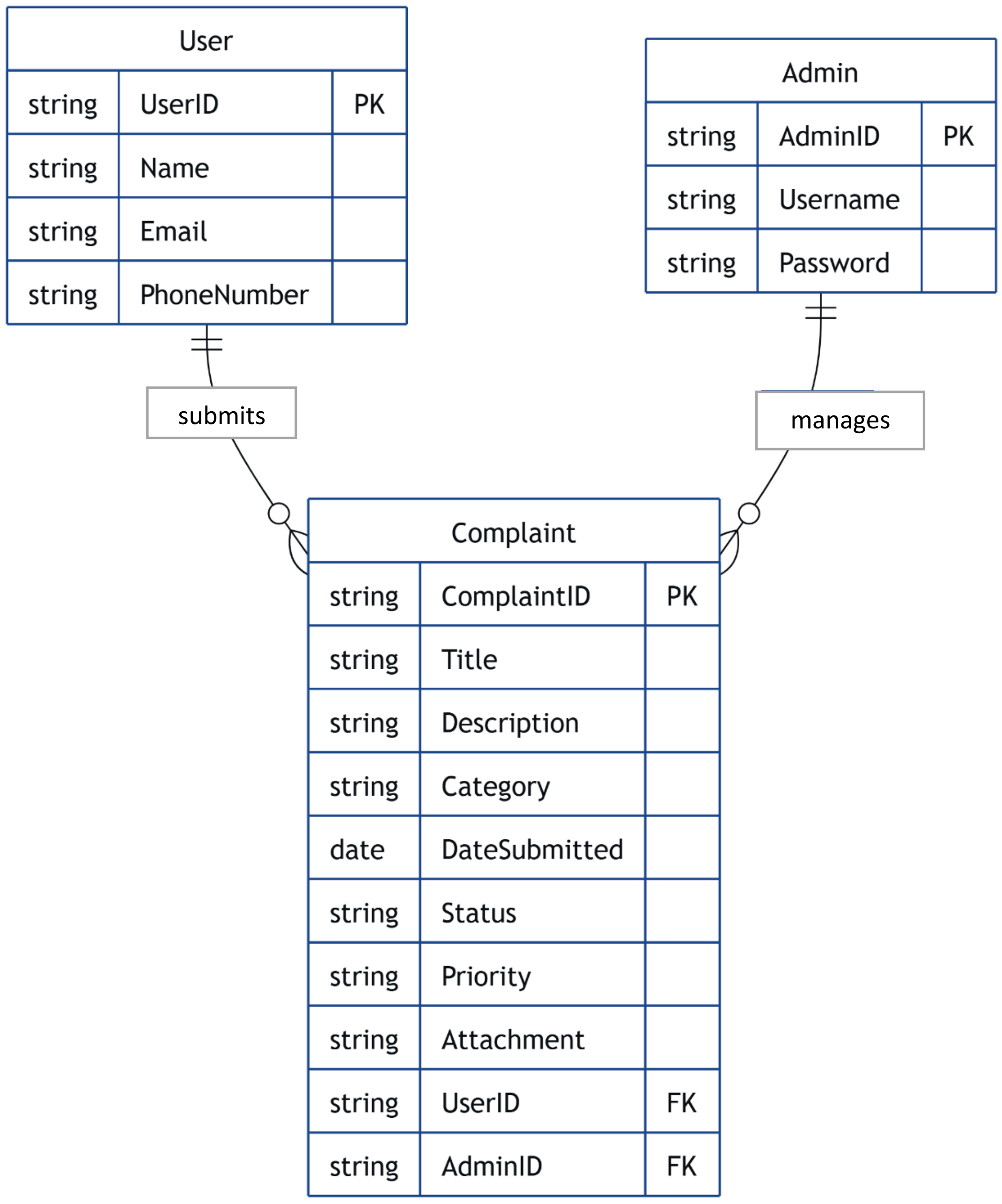


Figure 5: Entity Relationship Diagram

4.3 Design and Test Steps / Criteria

The system was designed with modularity, reusability, and user experience in mind. Each component was tested to ensure smooth functionality.

Design Approach

- Separation of concerns using HTML (structure), CSS (style), and JS (logic).

- Encapsulated functionality in a single class ComplaintManager.
- Tooltips and real-time character counter added for user convenience.
- Sidebar layout with toggle button for mobile responsiveness.

Testing Criteria

To ensure the Complaint Management System (CMS) performs as expected and delivers a reliable user experience, the application was subjected to rigorous testing under various functional and usability conditions. Each module and feature was validated to confirm adherence to predefined requirements and error-free operation across standard browsers and devices.

1. Field Validation Before Form Submission:

One of the primary areas tested was the user complaint form. Each input field — including name, email, department, issue type, priority, date, and detailed description — was thoroughly tested to confirm that it accepts only valid data. The system successfully prevents submission when any mandatory field is left empty or contains invalid input. Edge cases such as invalid email formats, overly short or long descriptions, or missing priority levels were tested and appropriately handled with real-time error messages, ensuring data integrity at the entry point.

2. Prevention of Form Submission with Invalid Inputs:

To maintain the accuracy and usefulness of stored data, it was verified that the form does not allow submission if any required field is left blank or fails validation. For example, attempts to bypass validations using browser developer tools or by disabling JavaScript were identified and mitigated. This ensures that no incomplete or incorrect data is processed by the system, maintaining the integrity of complaint records.

3. Admin Login and Access Restriction:

The admin login functionality was tested to ensure that access to the administrative panel is properly restricted. Attempts to access admin controls without valid credentials resulted in appropriate denial of access. Default credentials were validated, and simulated dynamic credential handling was tested to ensure only authorized users could log in and perform administrative tasks like viewing, editing, updating, or deleting complaints

4. PDF Generation Accuracy:

The jsPDF-based complaint receipt generation was tested to confirm that each receipt accurately reflects the information entered by the user at the time of submission. The test cases included checking for correct formatting, presence of all relevant fields, alignment, and inclusion of the unique complaint ID. It was also ensured that the PDF downloads correctly on various browsers without breaking layout or missing data.

5. Performance of Admin Actions:

Core administrative functionalities — such as viewing complaint details, filtering by status, updating complaint statuses, and deleting complaints — were all tested for real-time response and reliability. The application successfully handled these operations without lag, delay, or data loss. These actions dynamically updated the localStorage and DOM, ensuring that the UI remained responsive and reflected all changes instantly.

6. Seamless Navigation Between Sections:

The user experience was further tested by navigating between various parts of the application, including the complaint form, admin login, dashboard, and admin panel. Transitions were smooth and intuitive, with no visual glitches or broken links. On both desktop and mobile views, the layout adapted correctly, and elements retained their intended positioning and styling, confirming the system's responsive design and navigational robustness.

4.4 Algorithms and Pseudo Code

To implement complex operations, certain algorithms and logic were applied and structured as pseudo code for easy understanding.

4.4.1 Complaint ID Generation

Each complaint is given a unique ID using a simple counter stored in localStorage.

Pseudo Code:

```
Start
If 'currentComplaintId' not in localStorage:
   Set complaintId = 1
Else:
   Fetch complaintId from localStorage
On new complaint submission:
   Assign current complaintId
   Increment complaintId
   Store updated value back in localStorage
End
```

4.4.2 Form Validation

Each field is validated when user interacts with the form.

Pseudo Code:

```
For each form field:
    If field is empty:
        Mark as invalid
        Show error message
    Else if field format is incorrect:
        Show format-specific error
    Else:
        Mark as valid
        Allow form submission
```

4.4.3 PDF Generation

After submission, complaint details are compiled into a PDF using jsPDF.

Pseudo Code:

```
On complaint success:
    Fetch complaint details
    Create jsPDF instance
    Add complaint fields (ID, Name, Issue, etc.)
    Format description with word-wrap
    Add submission timestamp
    Save file as 'complaint-ID.pdf'
```

4.5 Testing Process

Testing is a critical phase to ensure software reliability and correctness. The following types of testing were conducted:

1. Unit Testing

- Each JavaScript function (e.g., validateField, generatePDF) tested individually.
- Verified state changes in localStorage after actions.

2. UI/UX Testing

- Checked responsiveness on mobile, tablet, and desktop screens.
- Ensured

- all icons, buttons, and labels are properly aligned.

3. Functional Testing

- Submitted multiple complaints and validated dashboard updates.
- Checked correct rendering and filtering in Admin Panel.

4. Edge Case Testing

- Entered long and short strings, special characters, and blank submissions.
- Tested complaints with maximum character limits (e.g., 1000 characters).

5. Browser Compatibility

- Verified system works correctly on Chrome, Edge, and Firefox.

All tests passed successfully and the system was found to be stable, fast, and user-friendly.

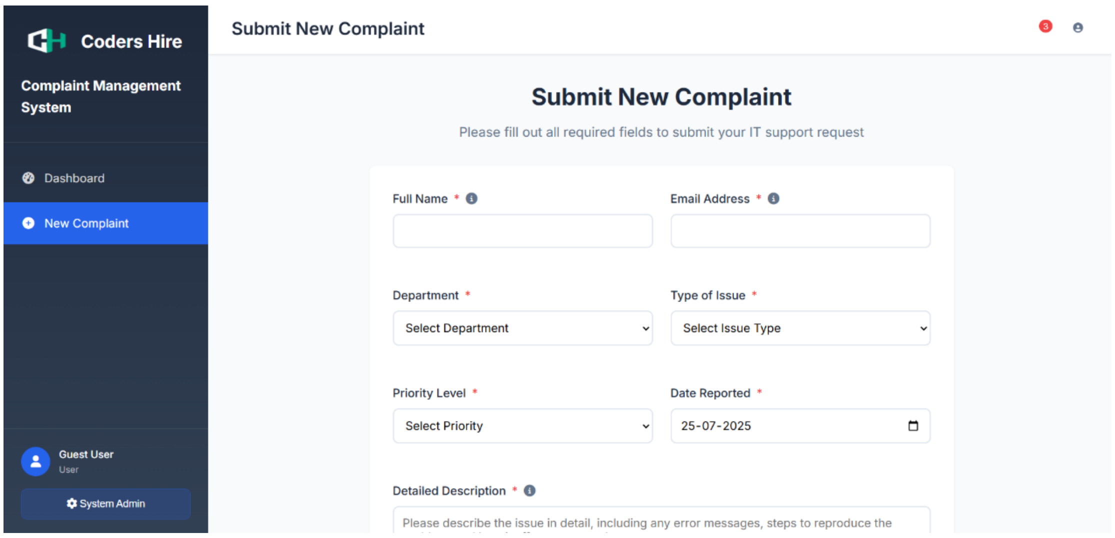
CHAPTER 5

RESULTS / OUTPUTS

The successful implementation of the **Complaint Management System** led to a functional, interactive, and responsive web application. Testing was done in various user environments, and the system met all defined objectives.

5.1 User-Side Outputs:

- **Complaint Form Interface**
  A responsive form with real-time validation, tooltips, and character counter was presented to the user. Input errors were highlighted immediately, enhancing user experience.



Submit New Complaint

Submit New Complaint

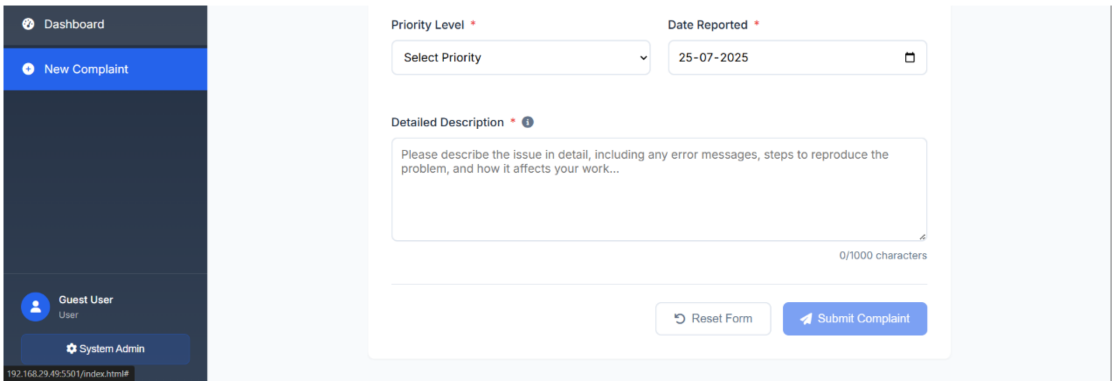
Please fill out all required fields to submit your IT support request

Full Name *	Email Address *
Department *	Type of Issue *
Select Department	Select Issue Type
Priority Level *	Date Reported *
Select Priority	25-07-2025

Detailed Description *

Please describe the issue in detail, including any error messages, steps to reproduce the

Figure 6: Complaint Form Interface Part 1



Priority Level *	Date Reported *
Select Priority	25-07-2025

Detailed Description *

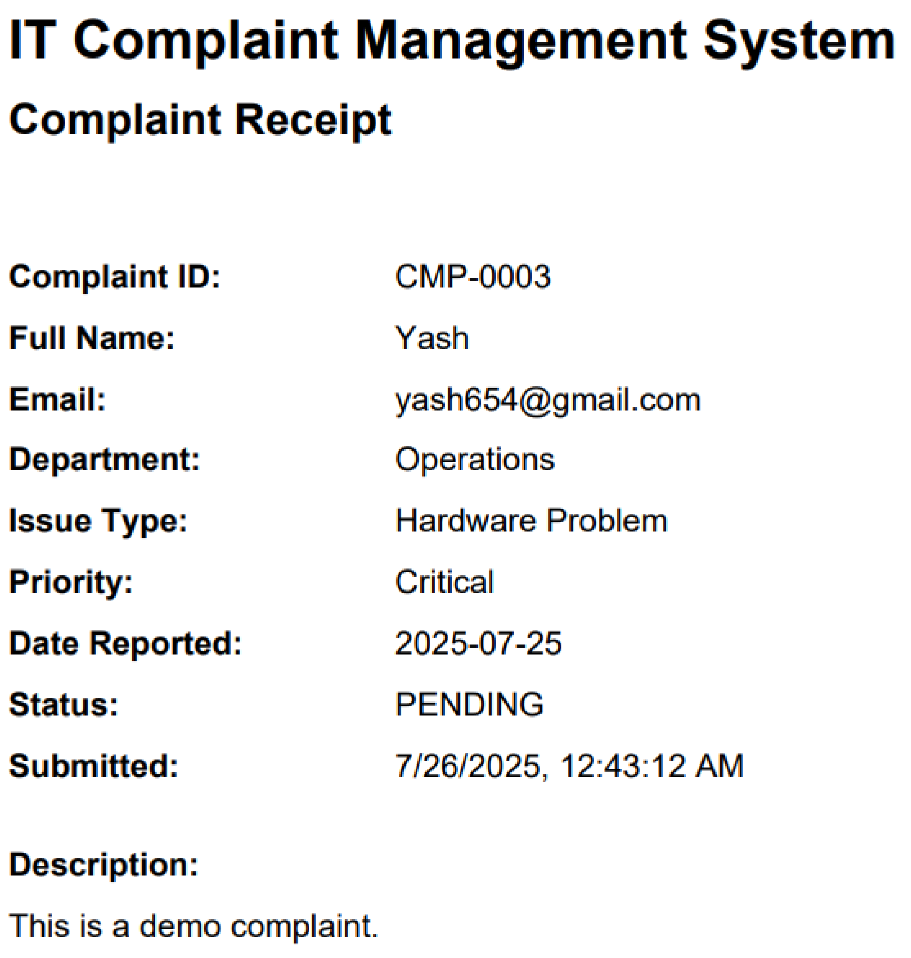
Please describe the issue in detail, including any error messages, steps to reproduce the problem, and how it affects your work...

0/1000 characters

Figure 7: Complaint Form Interface Part 2

Complaint ID and Timestamp Generation
Every submitted complaint was automatically assigned a unique ID and timestamp, ensuring traceability and proper record-keeping.

- **PDF Generation**
After form submission, users received a downloadable PDF receipt of their complaint, formatted professionally using the jsPDF library.



IT Complaint Management System

Complaint Receipt

Complaint ID:	CMP-0003
Full Name:	Yash
Email:	yash654@gmail.com
Department:	Operations
Issue Type:	Hardware Problem
Priority:	Critical
Date Reported:	2025-07-25
Status:	PENDING
Submitted:	7/26/2025, 12:43:12 AM

Description:

This is a demo complaint.

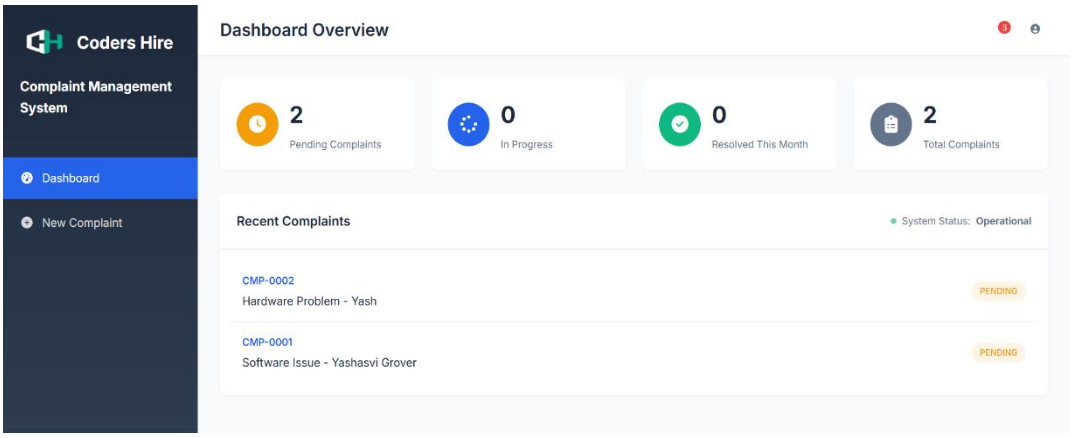
Figure 8: Complaint Receipt

- **Complaint Confirmation Modal**
A modal popup displayed a success message, complaint ID, and download link upon successful submission.

- **Data Persistence**
Complaints remained saved even after page refresh or browser closure, proving the effective use of localStorage.

5.2 Admin-Side Outputs:

- **Admin Login Panel**
  Secure login interface that restricted access to admin functionalities.
- **Dashboard Overview**
  Live stats of complaint categories: Pending, In Progress, Resolved, and Total were displayed visually on the dashboard.



Dashboard Overview

2 Pending Complaints
0 In Progress
0 Resolved This Month
2 Total Complaints

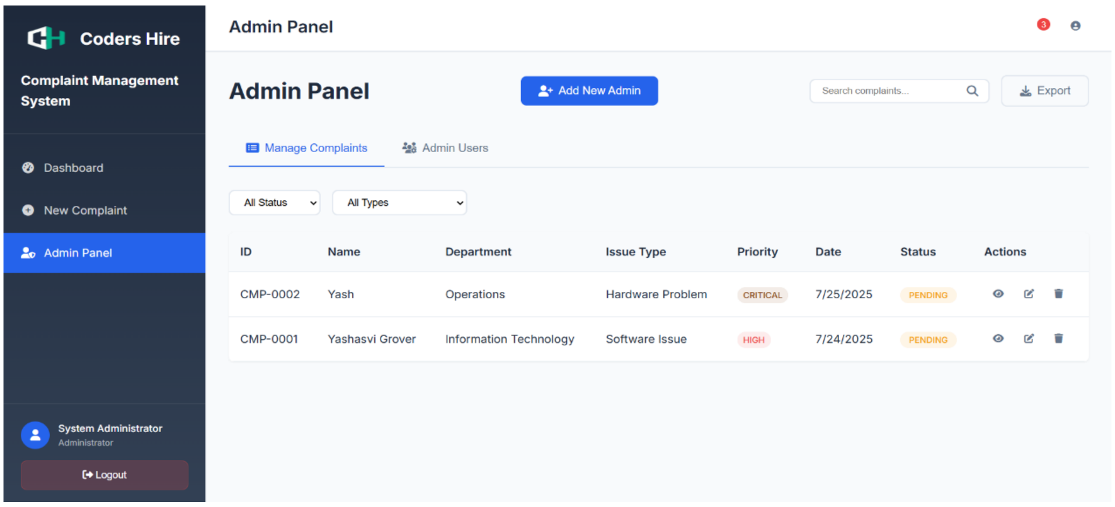
Recent Complaints — System Status: Operational

CMP-0002
Hardware Problem - Yash — PENDING

CMP-0001
Software Issue - Yashasvi Grover — PENDING

Figure 9: Dashboard Overview

- **Complaint Management Table**
  Admin could:
  - View all complaints in a sortable table.
  - Filter complaints by issue type and status.
  - Update status from dropdown menus.
  - Delete complaints with a single click.
- **Admin Management**
  Option to add new admin users was implemented and tested successfully.



Admin Panel

ID	Name	Department	Issue Type	Priority	Date	Status	Actions
CMP-0002	Yash	Operations	Hardware Problem	CRITICAL	7/25/2025	PENDING	
CMP-0001	Yashasvi Grover	Information Technology	Software Issue	HIGH	7/24/2025	PENDING	

Figure 10: Admin Dashboard Overview

- **Mobile Responsiveness**
  The admin panel adjusted layout correctly on mobile devices, maintaining usability.

CHAPTER 6
CONCLUSION

6.1 Conclusion

The developed Complaint Management System stands as a robust and efficient digital solution tailored for streamlining the complaint-handling process within an organization. By replacing outdated and inefficient manual methods, the system introduces an intuitive, interactive, and user-friendly web-based interface that enables users to file complaints effortlessly. The platform ensures a seamless flow from complaint submission to resolution, offering essential features such as form validation, complaint tracking, status updates, and automated PDF generation — all developed using core HTML, CSS, and JavaScript without reliance on server-side scripting or database connectivity.

This front-end-only architecture was intentionally chosen to reduce complexity, increase accessibility, and ensure lightweight deployment across various devices and screen sizes. The system's fully responsive design ensures compatibility with mobile and desktop platforms, making it ideal for internal use in IT departments, HR divisions, maintenance teams, or any organizational unit that requires issue tracking and resolution workflows.

Additionally, the project fulfills all functional requirements such as real-time form processing, printable complaint summaries, and structured UI navigation, as well as non-functional requirements like ease of use, portability, performance efficiency, and aesthetic design. The absence of backend dependencies not only simplifies hosting and usage but also reduces security risks and maintenance overhead, making it a practical and scalable prototype for organizations aiming to digitize their internal support systems.

Overall, this Complaint Management System provides real-world utility, especially for small to mid-sized enterprises, startups, and academic institutions looking for an immediate, accessible, and cost-effective solution to manage grievances and complaints systematically.

6.2 Recommendations

While the current implementation of the Complaint Management System (CMS) meets its intended functional requirements and demonstrates effective frontend-only complaint handling, there are still opportunities for optimization, risk mitigation, and future readiness. The following recommendations are proposed to enhance data reliability, improve administrative efficiency, and prepare the system for production-level deployment.

1. Periodic Backup of LocalStorage Data:

As the CMS relies solely on the browser's localStorage for data persistence, all complaint records and admin credentials are stored client-side. However, this storage is prone to accidental clearing (e.g., by clearing browser cache or resetting settings), and may vary across browsers. To mitigate the risk of permanent data loss, a backup feature should be implemented that allows administrators to export all stored data periodically to a secure file format (JSON or CSV). This data can then be saved externally and re-imported in the event of corruption or loss, ensuring business continuity.

2. Role-Based Access Control (RBAC):

Currently, all administrative privileges are tied to a single "admin" login system. Introducing granular access control roles—such as Admin, Super Admin, and Viewer—would help regulate permissions within the system. For example, a Viewer may only be allowed to view complaints without editing them, while a Super Admin can add or remove other admin users and perform sensitive actions like complaint deletion. This hierarchy improves system security and reduces the risk of unauthorized or accidental data manipulation.

3. Enhanced Data Export Options (CSV/Excel):

To support reporting, data analysis, and auditing, the CMS should provide a data export functionality where complaints can be downloaded in structured formats like CSV (Comma-Separated Values) or Microsoft Excel files (.xlsx). This would allow administrators to review complaint trends, conduct bulk analysis in spreadsheet tools, and even share complaint data with stakeholders without relying solely on the application interface.

4. Advanced Search and Sorting Features:

With the growing volume of complaints, it becomes increasingly important for admins to quickly locate relevant records. Adding features such as search by keyword, filter by complaint status or issue type, and date range selection would significantly enhance navigation. These capabilities would allow for real-time sorting, searching, and categorization of complaints based on dynamic criteria, improving efficiency and reducing manual effort during high-volume periods.

6.3 Future Scope

The current version of the Complaint Management System functions effectively as a standalone frontend application for internal use or prototyping. However, to extend its use to real-world deployment at scale, several future enhancements can be introduced.

These would elevate the application into a production-grade solution capable of supporting enterprise environments.

1. Backend Integration (Node.js, PHP, Firebase, etc.):

Integrating the application with a backend server is the most important step toward scalability. Backend technologies such as Node.js, Express, PHP, or Firebase can be used to store complaint data securely in databases like MongoDB, MySQL, or Firebase Realtime Database. This shift will allow for multi-user access from different devices, support user login sessions, and ensure that data is centrally stored and protected on the cloud. Backend integration would also enable API-based complaint submission, history retrieval, and admin authentication.

2. Email and SMS Notifications:

To improve communication and engagement, the system can be extended to send automated email or SMS alerts. Users can receive notifications whenever the status of their complaint is updated, and admins can be alerted instantly when new complaints are registered. Email services like SendGrid, Mailgun, or SMTP APIs, and SMS gateways like Twilio or Fast2SMS, can be integrated to automate these communications.

3. Graphical Analytics and Reporting Module:

To provide insights into organizational issues and improve decision-making, a visual analytics dashboard can be added. This module would display data-driven reports using pie charts, bar graphs, and trend lines, showing key metrics such as:

- Number of complaints over time
- Most frequently reported issues
- Average resolution time
- Department-wise complaint distribution

Libraries like Chart.js or Google Charts can be used to create dynamic, interactive visualizations that make complaint data more actionable.

4. Print and Export Options for Records:

In addition to individual complaint receipts, the system can offer a mass export or print feature. This would allow users or admins to download entire complaint logs in PDF, CSV, or Excel format for record-keeping, compliance, or offline reference. Batch generation of printable reports could be useful for monthly audits or administrative reviews.

5. User Accounts and Complaint Tracking System:

To streamline the complaint-follow-up process and reduce admin load, user accounts or token-based complaint tracking can be introduced. Users could log in securely using email/password or OTP, or receive a unique complaint token ID upon submission. Using this, they can track the real-time status of their complaint without requiring admin intervention. It also allows the history of previous complaints to be accessed for returning users.

6. Deployment on Hosting Platforms:

To make the CMS publicly accessible or accessible across a wider organizational network, it can be deployed on hosting platforms such as GitHub Pages, Netlify, or Firebase Hosting. These services provide free and reliable hosting for frontend projects, support custom domain mapping, and offer built-in features like CI/CD pipelines, HTTPS security, and deployment previews. Making the application live would enable remote access and open the door for wider adoption and feedback.

REFERENCES

- **MDN Web Docs (developer.mozilla.org)**
  The Mozilla Developer Network (MDN) provided detailed documentation and examples for HTML, CSS, and JavaScript, which were instrumental in understanding syntax, best practices, and implementing responsive web elements.

- **W3Schools (www.w3schools.com)**
  Used for quick reference on HTML tags, CSS properties, and JavaScript functions. The tutorials helped in rapid prototyping and implementation of UI components.

- **Stack Overflow (stackoverflow.com)**
  Referenced for resolving specific coding issues and understanding how to handle events, DOM manipulation, and debugging techniques during the development process.

- **CSS-Tricks (css-tricks.com)**
  Utilized for understanding complex CSS layouts, animations, and styling techniques, especially Flexbox and Grid for responsive design.

- **JavaScript Info (javascript.info)**
  This site helped in learning and applying modern JavaScript features like fetch, async/await, event handling, and modular code structure.

- **Google Fonts (fonts.google.com)**
  Used to enhance the visual appeal of the project by integrating custom web fonts.

- **Font Awesome (fontawesome.com)**
  Icons from Font Awesome were used to improve the UI/UX by adding meaningful visuals to buttons, navigation, and forms.

- **GitHub (github.com)**
  For code versioning, referencing open-source examples, and publishing the project repository.

- **CodePen / JSFiddle**
  Online editors like CodePen and JSFiddle were used for experimenting with UI elements before final integration.

- **Netlify / Vercel / GitHub Pages**
  For hosting the final web project online and testing deployment issues.

- **YouTube Tutorials (e.g., CodeWithHarry, Web Dev Simplified)**
  Video tutorials that guided the practical implementation of concepts.

APPENDICES

Appendix A: Details of Software / Simulator

- **Frontend Languages Used:**
  - HTML5
  - CSS3
  - JavaScript (ES6)
- **Libraries and Tools:**
  - jsPDF – For PDF generation
  - Font Awesome – For UI icons
  - Google Fonts – Typography
  - Web Storage API (localStorage) – For storing data locally
- **Editor Used:**
  - Visual Studio Code
- **Execution Environment:**
  - Runs on any modern web browser (Chrome, Firefox, Edge)
  - No server or backend required
  - Offline functionality after initial load

Appendix B: Steps to Execute / Run / Implement the Project

1. **Download or Clone the Project Files:**
    - Files include: index.html, style.css, script.js.
2. **Open in Code Editor (Recommended: VS Code):**
    - Make changes if necessary or preview the code.
3. **Run the Application:**
    - Double-click index.html or open it via browser (Right-click → Open with → Chrome/Edge).
4. **File a Complaint:**
    - Fill out the form with appropriate fields.
    - Click submit → a PDF receipt is generated.
    - Complaint is saved in browser memory (localStorage).
5. **Login as Admin:**
    - Click "System Admin" and enter default credentials (e.g., admin / admin123).
    - Admin can view, update, delete, and filter complaints.
6. **Test Responsiveness:**
    - Open in mobile browser or resize the screen to see responsive behavior.
7. **Add New Admins:**
    - Use the Admin Panel's "Add Admin" option to register more admins.
8. **Export as PDF:**
    - Use built-in PDF functionality or print page to PDF for reporting.