##### “DIGITAL COMPLAINT BOX”

**Realtime Research Project/Societal Related Project Submitted To**

##### JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

In partial fulfilment of the academic requirement for the award of the degree

Of

##### BACHELOR OF TECHNOLOGY

**IN**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

****

Submitted by

**Ghouse Pasha 23081A0503**

**Mohd Naveed 23081A0509**

**Asfakur Rahman 23081A0536 DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**

##### SHADAN COLLEGE OF ENGINEERING & TECHNOLOGY

**(AN AUTONOMOUS INSTITUTION)**

**(NAAC -A+ & NBA Accredited and ISO 9001:2015 certified institution)**

##### PEERANCHERU, HYDERABAD-500086

**(Affiliated to JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD)**

##### HYDERABAD-500087 (2024-2025)

Department of Computer Science and Engineering, SCET 1

##### “DIGITAL COMPLAINT BOX”

**Realtime Research Project/Societal Related Project Submitted To**

## JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

In partial fulfilment of the academic requirement for the award of the degree OF

## BACHELOR OF TECHNOLOGY

**IN**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

****

**UNDER THE GUIDANCE OF**

**Dr. G SRIDHAR**

Submitted by

**Ghouse Pasha 23081A0503**

**Mohd Naveed 23081A0509**

**Asfakur Rahman 23081A0536**

**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

## SHADAN COLLEGE OF ENGINEERING & TECHNOLOGY

##### (AN AUTONOMOUS INSTITUTION)

**(NAAC -A+ & NBA Accredited and ISO 9001:2015 certified institution)**

## PEERANCHERU, HYDERABAD-500086

**(Affiliated to JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY**

## HYDERABAD) HYDERABAD-500087

###### (2024-2025)

Department of Computer Science and Engineering, SCET 2

## DECLARATION

I hereby declare that the project report entitled “DIGITAL COMPLAINT BOX” submitted by us to SHADAN COLLEGE OF ENGINEERING AND TECHNOLOGY HYDERABAD” in partial

fulfilment for the award of B.TECH in ARTIFICIAL INTELLIGENCE AND DATA SCIENCE” is a record of Bonafide project work carried out by our team under the guidance of “DR.G SRIDHAR” submitted that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university

## DATE SIGNATURE OF STUDENT

|  |  |  |
| --- | --- | --- |
| **Ghouse Pasha** | **23081A0503 (** | **)** |
| **Mohd Naveed** | **23081A0509 (** | **)** |
| **Asfakur Rahman** | **23081A0536 (** | **)** |

Department of Computer Science and Engineering, SCET 3

##### CERTIFICATE

This is to certify that the dissertation certified “DIGITAL COMPLAINT BOX” is a Bonafide work done and

submitted by

|  |  |  |
| --- | --- | --- |
| **Ghouse Pasha** | **23081A0503 (** | **)** |
| **Mohd Naveed** | **23081A0509 (** | **)** |
| **Asfakur Rahman** | **23081A0536 (** | **)** |

In partial fulfilment of requirement for the award of degree of bachelor of technology inCOMPUTER SCIENCE ENGINEERING. Shadan College of Engineering & Technology. Affiliated to Jawaharlal Nehru Technological University. Hyderabad

is a record of Bonafide work carried out by them under our guidance and supervision.

The results presented in this discussion have been verified and have been found to be satisfactory. The results embodied in this dissertation have not been

Submitted to any other university for the award of any other degree or diploma.

**INTERNAL EXAMINER HEAD OF THE DEPARTMENT**

#### EXTERNAL EXAMINER

Department of Computer Science and Engineering, SCET 4

##### ACKNOWLEDGEMENT

* I thank GOD almighty for guiding us throughout the project.
* I would like to thank our parents without whose blessing, we would not have been able to accomplish our goal.
* I would like to thank our internal guide **DR.G.SRIDHAR,** for all the help and support he extended to us.
* I am extremely grateful to **DR.G.SRIDHAR, HEAD OF THE DEPARTMENT OF CSE,** for providing us with the best facilities and atmosphere for the creative work, guidance and encouragement.
* I would like to thank **DR. ATEEQ UR RAHMAN, PRINCIPAL** and all staff members of our college and friends for extending their cooperation during our project.

##### SUBMITTED BY

|  |  |  |
| --- | --- | --- |
| **Ghouse Pasha** | **23081A0503 (** | **)** |
| **Mohd Naveed** | **23081A0509 (** | **)** |
| **Asfakur Rahman** | **23081A0536 (** | **)** |

Department of Computer Science and Engineering, SCET 5

“DIGITAL COMPLAINT BOX”

Department of Computer Science and Engineering, SCET 6

|  |  |  |
| --- | --- | --- |
| TITLE | TABLE OF CONTENTS | PAGE NO |
| NOTATION TABLE |  | 1 |
| LIST OF ABBREVIATION |  | 4 |
| ABSTRACT |  | 5 |
| CHAPTER 1: INTRODUCTION  1.1 GENERAL |  | 6 |
| 1.2 SCOPE OF THE PROJECT |  | 6 |
| 1.3 OBJECTIVE |  | 7 |
| 1.4 EXISTING SYSTEM |  | 7 |

* + 1. EXISTING SYSTEM DISADVANTAGES 8
    2. [LITERATURE SURVEY 8](#_TOC_250025)
  1. [PROPOSED SYSTEM 10](#_TOC_250024)
     1. [PROPOSED SYSTEM ADVANTAGE 13](#_TOC_250023)

CHAPTER 2: PROJECT DESCRIPTION

* 1. [GENERAL 14](#_TOC_250022)
  2. [METHODOLOGIES 14](#_TOC_250021)
     1. MODULES NAME 14
     2. MODULES DIAGRAMS 14
  3. [UI DESIGN 14](#_TOC_250020)
     1. [USER INTERFACE DESIGN 14](#_TOC_250019)
     2. [USERS 15](#_TOC_250018)
  4. [GIVEN INPUT EXPECTED OUTPUT 16](#_TOC_250017)
  5. [TECHNIQUE OR ALGORITHM USED 17](#_TOC_250016)
     1. [PROPOSED ALGORITHM 17](#_TOC_250015)

Department of Computer Science and Engineering, SCET 7

|  |  |  |
| --- | --- | --- |
| CHAPTER 3: REQUIREMENTS ENGINEERING  3.1 GENERAL | 19 |  |
| 3.2 HARDWARE REQUIREMENTS | 20 |  |
| 3.3 SOFTWARE REQUIREMENTS | 20 |  |
| 3.4 FUNCTIONAL REQUIREMENTS | 21 |  |
| 3.5 NON-FUNCTIONAL REQUIREMENTS | 22 |  |
| 3.6 DOMAIN REQUIREMENT | 23 |  |
| CHAPTER 4: SYSTEM DESIGN  4.1 GENERAL | 25 |  |
| 4.2 SYSTEM ARCHITECTURE | 25 |  |
| 4.3 UML | 27 |  |
| 4.3.1 USE CASE DIAGRAM | 27 |  |
| 4.3.2 CLASS DIAGRAM | 28 |  |
| 4.3.3 OBJECT DIAGRAM | 28 |  |
| 4.3.4 COMPONENT DIAGRAM | 29 |  |
| 4.3.5 DEPLOYMENT DIAGRAM | 30 |  |
| 4.3.6 SEQUENCE DIAGRAM | 31 |  |
| 4.3.7 COLLABORATION DIAGRAM | 31 |  |
| 4.3.8 STATE DIAGRAM | 32 |  |
| 4.3.9 ACTIVITY DIAGRAM | 32 |  |
| 4.4 DATA FLOW DIAGRAM | 33 |  |
| * 1. E-R DIAGRAM   2. GUI DESIGN      1. COMPONENTS OF GUI | 34  35 |  |
| 4.6.2 FEATURES OF GUI | 35 |  |
| CHAPTER 5: IMPLEMENTATION  5.1 GENERAL | 37 |  |
| 5.2 IMPLEMENTATION | 37 |  |
| Department of Computer Science and Engineering, SCET |  | 8 |

CHAPTER 6: SNAPSHOTS

* 1. [GENERAL 73](#_TOC_250014)
  2. [OUTPUT SNAPSHOTS 73](#_TOC_250013)

CHAPTER 7: SOFTWARE TESTING

* 1. [GENERAL 76](#_TOC_250012)
  2. [DEVELOPING METHODOLOGIES 76](#_TOC_250011)
  3. [TEST STRATEGY](#_TOC_250010)
     1. [LEVELS OF TESTING 77](#_TOC_250009)
     2. [TYPES OF TESTING 78](#_TOC_250008)
     3. [TEST CASE TYPE – GUI 80](#_TOC_250007)
     4. [TEST DESIGN TECHNIQUES 81](#_TOC_250006)
     5. [TEST ENVIRONMENT 81](#_TOC_250005)
  4. [ACCEPTANCE CRITERIA 82](#_TOC_250004)
     1. [ACCEPTANCE TESTING 83](#_TOC_250003)

CHAPTER 8: CONCLUSION AND REFERENCES

* 1. [CONCLUSION 87](#_TOC_250002)
  2. [FUTURE ENHANCEMENT 87](#_TOC_250001)
  3. [REFERENCES 87](#_TOC_250000)

Department of Computer Science and Engineering, SCET 9

**NOTATION TABLE**

|  |  |  |
| --- | --- | --- |
| Class A | AM  E Cl | ass B |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.N O** | **NOTATION**  **NAME** | **NOTATION** | | **DESCRIPTION** |
| 1. | CLASS |  | | REPRESENTS A COLLACTION OF SIMILER ENTITIES GROUPED TOGETHER |
| + public  -private | CLASS NAME |
| -ATTRIBUTE |
| -ATTRIBUTE |
| 2. | ASSOCIATI ON | Class A Class B | | ASSOCIATION REPRESENTS STATIC RELATIONSHIPS BETWEEN CLASSES.ROLES REPRESENTS THE WAY THE TWO CLASSES  SEE EACH . |
| \*  3. | ACTOR |  | | IT AGGREGATES SEVERAL CLASSES INTO A SINGLE CLASSES. |

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | Aggregation | Class A Cla  ss  A  Class B  Cla ss B | Interaction between the system and external environment |
| 5. | Relation (uses) | uses | Used for additional process communication |
| 6. | Relation (extends) | Ex  extends | tends relationship is used when one use case is similar to another use case  but does a bit more. |
| 7. | Communicatio n |  | Communication between various use cases. |
| 9. | Initial State |  | Initial state of the object |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |

|  |  |  |  |
| --- | --- | --- | --- |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Usecase | Usescase | Interact ion between the system and external environment. |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |

|  |  |  |  |
| --- | --- | --- | --- |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,et c. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

## LIST OF ABBREVIATION

|  |  |  |
| --- | --- | --- |
| **S.No** | **Abbreviation** | **Expansion** |
| **1** | **DB** | **Database** |
| **2** | **HTML** | **HyperText Markup Language** |
| **3** | **CSS** | **Cascading Style Sheets** |
| **4** | **JS** | **JavaScript** |
| **5** | **SQL** | **Structured Query Language** |
| **6** | **UI** | **User Interface** |
| **7** | **UX** | **User Experience** |
| **8** | **SMTP** | **Simple Mail Transfer Protocol** |
| **9** | **UUID** | **Universally Unique Identifier** |
| **10** | **CRUD** | **Create, Read, Update, Delete** |
| **11** | **ORM** | **Object Relational Mapping (if you plan to use it in future)** |
| **12** | **API** | **Application Programming Interface** |

**ABSTRACT**

The *Digital Complaint Box* is a web-based platform developed using HTML, CSS, and JavaScript, designed to simplify the process of submitting complaints across different sectors. This system allows users to select categories such as **Government**, **Colleges**, and **Schools**, and within each, choose sub-sectors like **Water**, **Electricity**, **Road Transport**, and more. By providing a user-friendly interface and organized complaint submission process, the portal ensures that grievances are directed to the appropriate authorities in a structured manner.

The main objective of this platform is to **digitize and streamline traditional complaint systems**, making them more accessible, efficient, and transparent. It reduces the communication gap between users and institutions by offering a centralized solution for complaint lodging. The system is scalable and can be further enhanced with features like login systems, admin dashboards, and complaint tracking in future developments.

This project promotes accountability, saves time, and empowers users to voice their concerns effectively in today’s digital age.

# CHAPTER 1 INTRODUCTION

## GENERAL

The Digital Complaint Box is a simple online platform that allows people to submit complaints easily from anywhere. It helps users select a category like Government, College, or School, and then choose specific problems like Water issues, Electricity problems, Road conditions, or Academic concerns. This makes it easy to send the complaint to the right place.

The main aim of this project is to save time, avoid confusion, and give people a better way to share their problems. Instead of writing letters or going to offices, people can just use this website. It is made using HTML, CSS, and JavaScript, and has a clean and user-friendly design.

This project helps in building a better system where people can speak up and their issues are heard and solved quickly

## SCOPE OF THE PROJECT

The Digital Complaint Box is designed to provide a secure, user-friendly platform for individuals to submit complaints or feedback online. The main scope of this project is to bridge the gap between users and authorities by enabling efficient complaint registration, tracking, and resolution through a centralized system.

It aims to:

* Allow users to submit complaints digitally from anywhere at any time.
* Ensure complaints are stored securely in a database for easy retrieval and tracking.
* Provide admin access to view, manage, and resolve complaints efficiently.
* Offer transparency by showing status updates to the user.
* Reduce manual paperwork and delays in complaint handling.

This system is ideal for institutions like colleges, offices, or public service sectors where people face issues but lack a proper way to report them. It can be further enhanced with features like real-time alerts, data analytics, or chatbot support.

## OBJECTIVE:

The primary objective of the Digital Complaint Box is to provide a simple, accessible, and organized platform for users to register complaints related to different sectors such as Government, Colleges, and Schools. This platform aims to streamline the complaint submission process by allowing users to select specific sectors and sub-sectors, ensuring that their concerns reach the appropriate authorities quickly.

The system encourages transparency and accountability while reducing the delays and confusion of traditional complaint methods. By offering a digital solution, the project empowers individuals to voice their concerns easily and contributes to building a more responsive and citizen-friendly environment.

## EXISTING SYSTEMS

"existing systems" typically denote the infrastructure or software setup that is currently in use and serving its intended purpose. These include large-scale software applications used by organizations for managing various operations like erp (enterprise resource planning) systems, crm (customer relationship management) systems, and hris (human resource information systems).

**legacy systems**: older, often outdated systems that continue to run critical business functions due to the cost and effort required to replace them.

**embedded systems**: software embedded in hardware devices, such as operating systems running on iot (internet of things) devices, industrial machinery, or consumer electronics.

###### characteristics:

* **Purpose-built**: tailored to specific organizational needs or industry requirements.
* **Stable and reliable**: designed to operate reliably over extended periods with minimal disruption.
* **Integration**: often interconnected with other systems within an organization's it infrastructure

###### Software

"software" refers to programs and applications that run on a computer or other electronic devices, performing specific tasks or functions.

###### types:

**Operating systems (os)**: system software that manages computer hardware and software resources, providing a user interface and facilitating communication between hardware and applications.

**Application software**: programs designed for end-users to perform specific tasks, such as word processors, spreadsheets, email clients, and web browsers.

**Utility software**: tools that assist in managing and maintaining the computer system, such as antivirus programs, backup utilities, and disk defragmenters.

**Development software**: tools used by software developers to create, debug, and maintain applications, such as ides (integrated development environments) and version control systems.

###### lifecycle:

* **Development**: creating and coding software applications based on user requirements and specifications.
* **Testing**: ensuring software reliability, functionality, and security through various testing methodologies.
* **Deployment**: installing and implementing software on target devices or within organizational it infrastructure.
* **Maintenance**: providing ongoing support, updates, and troubleshooting to ensure optimal performance and security.

###### impact:

* **Productivity**: enhancing efficiency and enabling automation of tasks.
* **Innovation**: driving technological advancements and new capabilities.

**Security**: mitigating risks through secure coding practices and regular updates

## LITERATURE SURVEY

**Title:** Online Grievance Redressal System

**Author:** R. Sharma

**Year:** 2020

**Description:** This system allows users to submit their complaints digitally and track the resolution status. It promotes transparency and timely redressal of issues by enabling authorities to manage complaints more efficiently through a centralized portal.

**Title:** E-Complaint Management System for Institutions

**Author:** S. Verma & T. Roy

**Year:** 2019

**Description:** Focused on resolving student grievances in educational institutions, this system enables users to register concerns securely. It maintains a log of issues and helps administration spot common problems, improving institutional accountability.

**Title:** Smart Public Complaint Monitoring System

**Author:** A. Khan & L. Reddy

**Year:** 2021

**Description:** This model enhances complaint resolution speed through real-time alerts and mobile integration. Citizens can categorize complaints, receive updates, and evaluate service responses, especially in municipal and public sectors

**Title:** Digital Complaint Lodging and Tracking System

**Author:** M. Joshi & K. Patil

**Year:** 2022

**Description:** This system enables users to file complaints online with proper categorization and priority tagging. It emphasizes user authentication, admin response time, and data analytics to identify recurring issues and improve governance and service delivery.

## PROPOSED SYSTEM

The proposed system is a **web-based platform** that allows users (like students or employees) to **submit complaints online** in a secure and easy way. It focuses on **user anonymity, complaint tracking**, and **admin response system**.

Key features include:

* **User-friendly Interface** to submit complaints quickly
* **Anonymous mode** for privacy and safety
* **Status tracking** for each complaint
* **Admin dashboard** to manage, review, and respond to complaints
* **Email notifications** for updates (optional feature)

This system aims to **replace manual processes**, reduce delays, and ensure **transparency**

and **accountability** in handling complaints.

## PROPOSED SYSTEM ADVANTAGE

###### User-Friendly Interface

The portal provides a simple and intuitive interface, allowing users to lodge complaints easily without any technical knowledge.

###### Real-Time Tracking

Users can track the status of their complaints in real-time, ensuring transparency and accountability from the admin side.

###### Instant Notifications

Complainants receive updates via email/SMS as their complaint status changes, helping them stay informed at every stage.

###### Paperless and Efficient

Completely digital process eliminates paperwork, reduces manual errors, and speeds up resolution.

# CHAPTER 2

## PROJECT DESCRIPTION

## GENERAL:

The **Digital Complaint Box** is an essential online platform created to help people share their problems and complaints in a simple and organized way. It acts as a one-stop destination where users can select different sectors like **Government**, **Colleges**, or **Schools**, and within them choose specific issues such as **Water**, **Electricity**, **Road Transport**, or **Academic Problems**.

This portal makes it easier for users to report issues without visiting offices or filling physical forms. By offering a clean and user-friendly interface, the system ensures that every complaint reaches the right department quickly and clearly. It helps in saving time, reducing confusion, and improving how complaints are managed.

The **Digital Complaint Box** encourages people to raise their voice and plays a role in building a better, more responsive system where problems are heard and solved efficiently.

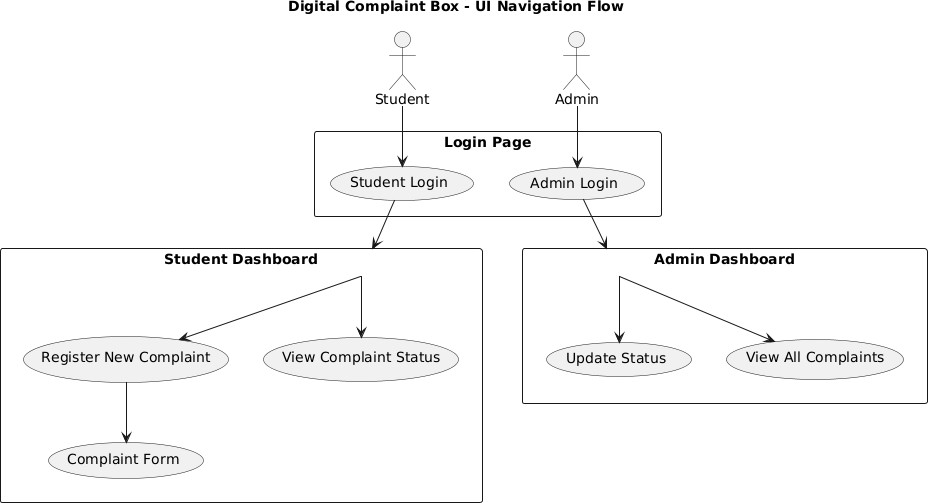
## METHODOLOGIES

1. User Interface Design
2. Users

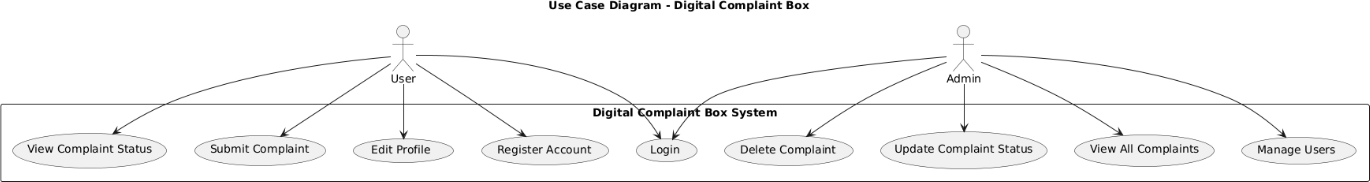
## UI DESIGN

## USER INTERFACE DESIGN

The User Interface (UI) of the Digital Complaint Box is designed to ensure simplicity, clarity, and ease of access for both students and administrators. The homepage provides separate login options for users and admin, each leading to a dedicated dashboard. Once logged in, users can register complaints using a simple form that includes dropdown options for complaint categories, a text area for description, and an optional file upload feature. Submitted complaints can be tracked by their status — such as Pending, In Progress, or Resolved — visible on the user's dashboard. On the admin side, the interface includes functionality to view all complaints, filter them based on category or urgency, and update the status accordingly. The entire layout is responsive and clean, focusing on accessibility across various devices, ensuring that users of all technical levels can interact with the portal efficiently.



## USERS

****

## GIVEN INPUT EXPECTED OUTPUT

### User Interface Design

###### Given Input:

* + - 1. User inputs complaint details like name, type of complaint, description, and location.
      2. User logs in using email/username and password.
      3. User clicks buttons to view status, submit form, or edit info.

###### Expected Output:

1. Proper form submission with success alert or validation error if incomplete.
2. Responsive interface that works on both desktop and mobile.
3. Clear visibility of complaint status, confirmation messages, and smooth navigation.

### User Module

###### Given Input:

* + - 1. User provides valid login credentials.
      2. User submits new complaint via input form.

###### Expected Output:

1. User is logged in and redirected to their dashboard.
2. Complaint is saved in the database and confirmation is shown

## TECHNIQUE OR ALGORITHM USED

## PROPOSED ALGORITHM

**Step 1:** User registers or logs in with valid credentials.

**Step 2:** Upon successful login, user accesses the complaint submission form.

**Step 3:** User fills in complaint details (type, description, location, etc.).

**Step 4:** System validates the entered information. If invalid, prompt user to correct it. **Step 5:** Valid complaint data is saved in the database, and a unique complaint ID is generated.

**Step 6:** User receives confirmation with complaint ID.

**Step 7:** User can track the status of their complaint using the complaint ID.

**Step 8:** System fetches and displays the current status (Pending, In Progress, Resolved).

**Step 9:** User logs out after completing actions.

## CHAPTER 3 REQUIREMENTS ENGINEERING

### General:

In this section, the overall architecture of the proposed Digital Complaint Box framework is discussed in detail (as shown in Figure 1). Initially, users from various sectors such as Government, Colleges, and Schools raise complaints individually related to specific issues like Water, Electricity, Road Transport, or Academic Problems. Each complaint is processed locally through a user-friendly interface, which guides the user in selecting the correct category and sub-category.

Once the user submits the complaint, the data is securely transmitted to a centralized server that classifies and redirects the complaint to the appropriate authority or department based on the chosen category. The server maintains a database of complaints and uses it to track the status, responses, and resolution time.

This architecture provides several benefits:

* + - Efficient Categorization and Redirection: Since complaints are pre-filtered by category and sub-category, the system ensures that the issue reaches the correct department, speeding up the resolution process.
    - User-Friendly and Decentralized Input: Complaints can be registered from any location and device, eliminating the need for physical visits to government offices or college administrations.
    - Transparency and Accountability: Each complaint is logged with a unique ID and can be tracked for updates, increasing trust in the system.
    - Privacy and Security: The system ensures that user data is kept confidential. Only the necessary details are forwarded to authorities, maintaining user privacy throughout the process.
    - Data Insight and Analytics: Over time, the collected complaint data can be used to generate reports and identify common issues, helping departments make better decisions and improvements.

## HARDWARE REQUIREMENTS

The hardware setup is important to run the Digital Complaint Box smoothly. It needs enough processing power, memory, and storage to handle real-time complaint reporting and database operations efficiently.

Minimum Requirement Recommended Requirement Processor: Intel i3 or equivalent Intel i5 or higher RAM: 4 GB 8 GB or more

Storage: 100 GB HDD 256 GB SSD

Operating System: Windows 7 / Linux / macOS Windows 10 or later Internet Speed: 2 Mbps 10 Mbps or higher

## SOFTWARE REQUIREMENTS

#### SOFTWARE REQUIREMENTS

To ensure the Digital Complaint Box runs efficiently, the following software components are needed. These tools help build, manage, and maintain the system’s functionalities smoothly.

Minimum Requirement Recommended Requirement

Operating System: Windows 7 / Linux / macOS Windows 10 or later

Web Browser: Chrome, Firefox, Edge (latest versions) Chrome or Firefox Backend Database: MySQL or SQL Server MySQL 8.0 or higher

Frontend Technologies: HTML, CSS, JavaScript Latest versions of HTML5, CSS3,

JavaScript (ES6+)

Server Environment: Apache / XAMPP / Node.js XAMPP or Node.js latest

stable version IDE/Code Editor: Notepad++, VS Code Visual Studio Code or Sublime Text

## FUNCTIONAL REQUIREMENTS

**Functional requirements** outline the specific actions or behaviors that the **Digital Complaint Box** system must perform to meet user needs and achieve its intended purpose.

They focus on what the system should do in terms of **input (complaints submission), processing (storing & managing data), and output (admin responses or status updates).**

**Types:**

1. **Basic Functional Requirements:**

These are core functionalities essential for the complaint box to fulfill its main purpose:

* + Allowing users to submit complaints.
  + Storing complaints in the database.
  + Providing admin access to view/manage complaints.

1. **Derived Functional Requirements:**

These arise from interactions between core features and user roles:

* + Automatic complaint ID generation.
  + Marking complaint status (e.g., Pending, Resolved).
  + Filtering complaints by department or type.

**Importance of Functional Requirements:**

* + **Guidance for Development:**

Gives clear roadmap to developers about features to implement like submission form, admin dashboard, etc.

* + **Basis for Testing:**

Functional requirements act as a checklist to test if features like complaint submission and status update are working as expected.

* + **Alignment with User Needs:**

Ensures the system actually solves the problem of anonymous or tracked complaint submission and resolution.

* + **Communication Tool:**

Helps in explaining the system flow to stakeholders like faculty, HODs, and students.

**Requirement Example:**

**Requirement:**

Users should be able to submit a complaint through the complaint box.

**Description:**

When a user fills the complaint form and clicks the "Submit" button, the complaint should be stored in the system along with user details, timestamp, and complaint ID.

**Acceptance Criteria:**

## NON-FUNCTIONAL REQUIREMENTS (NFRS)

**Non-functional requirements (NFRs)** specify how the system should behave in terms of quality and constraints. Unlike functional requirements that describe what the system should do, NFRs define how well it should perform, covering areas like reliability, usability, security, and performance.

###### Types of Non-Functional Requirements:

1. **Performance:**

Specifies how quickly the system should respond to actions like complaint submissions or admin operations.

*Example:* The system should respond within 2 seconds after a complaint is submitted.

###### Reliability:

Defines the ability of the system to remain functional over time, including uptime and failure recovery.

*Example:* System uptime should be 99.5%, and it should auto-recover from failures within 5 minutes.

###### Usability:

Describes how easy it is for users (students/admins) to interact with the system. *Example:* The interface should be simple, intuitive, and accessible from both desktop and mobile.

###### Security:

Specifies protection from unauthorized access and ensures data privacy.

*Example:* Login authentication is required for admins, and user data should be encrypted.

###### Scalability:

Refers to the system’s ability to handle an increasing number of users and complaints.

*Example:* The system should handle up to 1000 users without performance issues.

###### Maintainability:

Describes how easily the system can be updated or fixed.

*Example:* The code should be modular and documented to allow quick changes or bug fixes.

###### Compatibility:

Defines how well the system works with different browsers, devices, and platforms. *Example:* It should run smoothly on Chrome, Firefox, and Edge, and be responsive on all screen sizes.

###### Legal and Regulatory:

Ensures the system follows privacy rules and institutional policies.

*Example:* User complaints should be stored securely, and offensive content should be flagged.

## DOMAIN REQUIREMENT

**Contextual Needs**: They are specific to the domain or industry in which the software system will be used. These requirements are influenced by the unique characteristics, practices, regulations, and expectations of that domain.

**Constraints and Limitations**: They encompass the limitations or constraints imposed by the domain, which may include legal requirements, industry standards, technological limitations, or cultural considerations.

**Functional and Non-Functional Requirements**: They include both functional requirements (what the system should do) and non-functional requirements (quality attributes such as performance, scalability, security, usability) that are tailored to meet the domain-specific needs.

###### Healthcare Domain:

**Privacy and Security**: Compliance with HIPAA regulations for patient data protection.

**Interoperability**: Integration with electronic health record (EHR) systems.

**Clinical Workflow**: Support for specific workflows and processes used by healthcare professionals.

###### Financial Services Domain:

**Security and Compliance**: Adherence to financial regulations such as GDPR, PCI-DSS, or SEC rules.

**Performance**: Real-time processing for high-frequency trading systems. **Scalability**: Handling large volumes of transactions during peak times. **Education Domain**:

**Accessibility**: Compliance with accessibility standards for educational materials.

**User Roles and Permissions**: Differentiated access levels for students, teachers, administrators

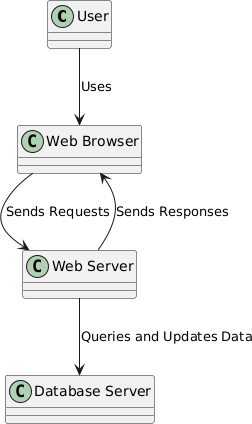
# CHAPTER 4

## SYSTEM DESIGN

* 1. **GENERAL**

The **Digital Complaint Box** is an online platform designed to help students and staff easily report issues, grievances, or suggestions within the college environment. It offers features such as complaint submission, status tracking, and secure communication, ensuring that users can raise their concerns quickly and confidently. The portal allows users to file complaints anonymously if needed, access support resources, and receive updates on the resolution process. With a focus on transparency and responsiveness, it aims to improve the campus atmosphere by addressing problems efficiently. By using technology, this platform fosters a safer and more supportive college community, making it easier for everyone to be heard and helped

## SYSTEM ARCHITECTURE

****

Components of the Diagram

###### User

* + Represents the person who interacts with the web application through a device.

###### Web Browser

* + Software application used by the user to access and interact with web content.
  + Communicates with the web server by sending requests and receiving responses.

###### Web Server

* + Hosts the web application and processes incoming requests from the web browser.
  + Responsible for generating responses based on the requests it receives.
  + Can query and update information stored in the database server.

###### Database Server

* + Stores data for the application, such as user information, content, etc.
  + The web server queries this server to retrieve data or updates it based on user actions.

Processes Illustrated

###### Uses:

* + Indicates the relationship between the user and the web browser.

###### Sends Requests/Sends Responses:

* + Illustrates the communication loop between the web browser and the web server. The web browser sends requests for information, and the web server responds with the requested data or confirmation of an action.

###### Queries and Updates Data:

* + Shows how the web server interacts with the database server to manage data necessary for fulfilling user requests.

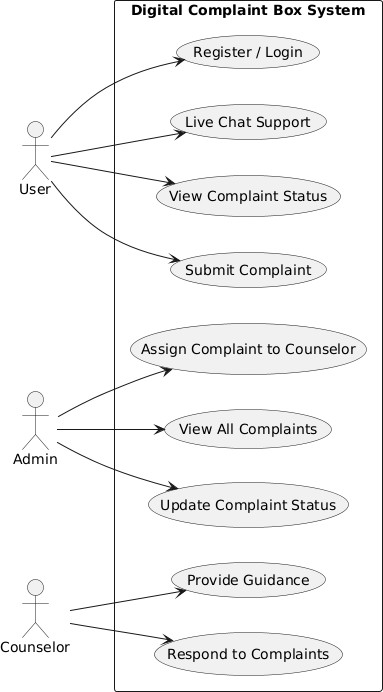
###### Summary of Interaction Flow

* The user uses a web browser to send requests to the web server.
* The web server processes these requests, potentially querying the database server for additional data or to perform updates.
* After processing, the web server sends a response back to the web browser, which is then presented to the user.

This architecture showcases the fundamental interactions in a typical web application, facilitating communication between the user, web browser, web server, and database server**.**

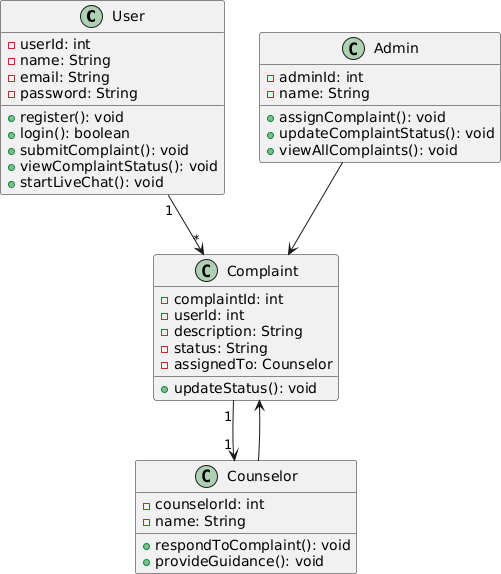
## UML

* + 1. **USE CASE DIAGRAM**

****

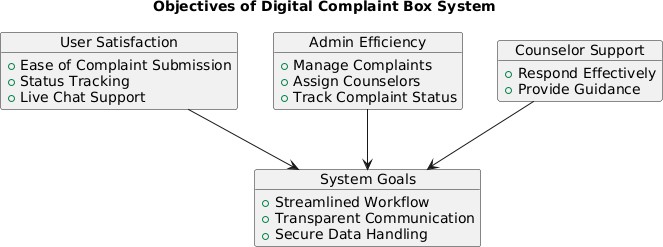
* + - * Register / Login
      * Users can create an account or log in to access system features.
      * Live Chat Support
      * Real-time chat feature to assist users instantly.
      * View Complaint Status
      * Users can check the current status of their submitted complaints.
      * Submit Complaint
      * Allows users to lodge a new complaint into the system.
      * Assign Complaint to Counselor
      * Admin assigns incoming complaints to the appropriate counselor.
      * View All Complaints
      * Admin can view the list of all complaints submitted by users.
      * Update Complaint Status
      * Admin updates the progress or resolution stage of complaints.
      * Provide Guidance
      * Counselors offer helpful advice and support based on the issue.
      * Respond to Complaints
      * Counselors send replies or solutions to the users' complaints.

## CLASS DIAGRAM

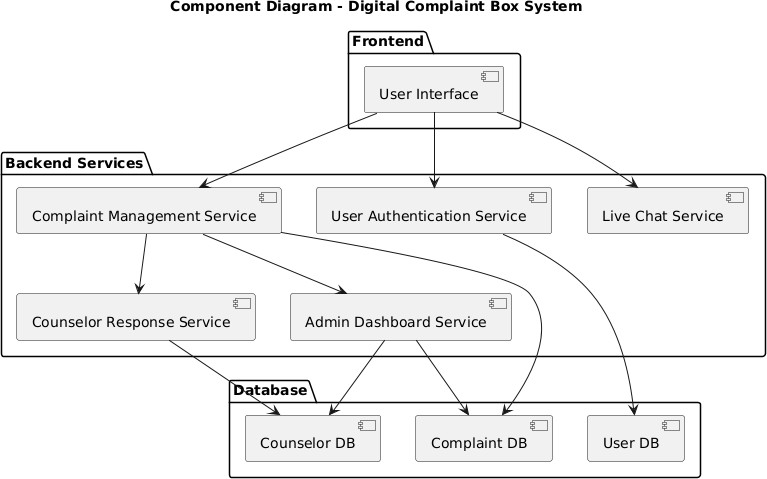
****

* + - * **User**: A person who registers, logs in, submits complaints, and checks their status.
      * **Complaint**: A class that stores complaint details, status, and the counselor assigned.
      * **Admin**: The authority who views all complaints, updates their status, and assigns them to counselors.
      * **Counselor**: A person responsible for responding to complaints and giving proper guidance.
      * **Relationship (User → Complaint)**: A user can create one or more complaints in the system.
      * **Relationship (Admin → Complaint)**: The admin manages all complaints by assigning and updating them.
      * **Relationship (Counselor → Complaint)**: A counselor handles and responds to assigned complaints.

## OBJECT DIAGRAM

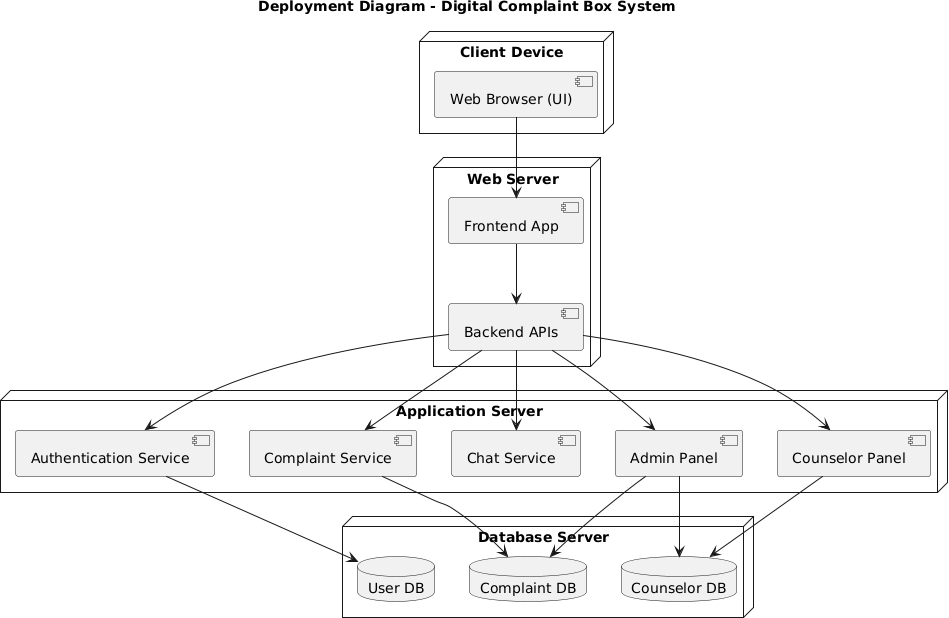
****

* + 1. **COMPONENT DIAGRAM**

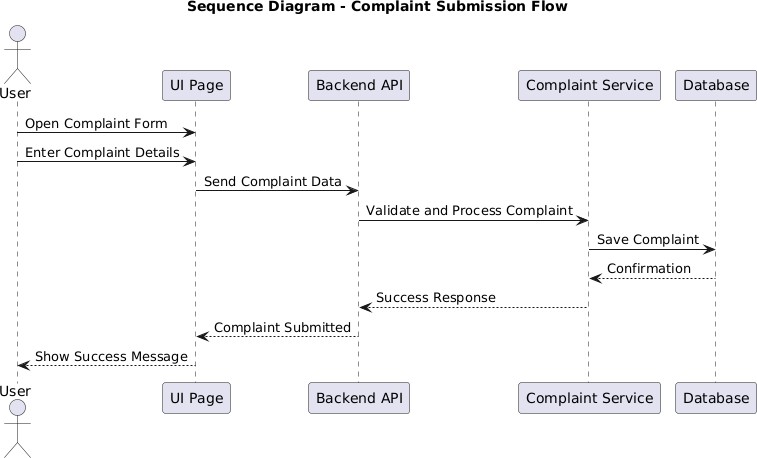
****

* + - * **User Interface**: The front-facing component where users interact (register, submit complaints, etc.).
      * **User Authentication Service**: Verifies and logs in users securely.
      * **Complaint Management Service**: Handles complaint creation, updates, and assignments.
      * **Live Chat Service**: Enables real-time support chat for users.
      * **Admin Dashboard Service**: Lets admin view, assign, and monitor complaints.
      * **Counselor Response Service**: Counselors can respond and give advice on complaints.
      * **User DB**: Stores user details securely.
      * **Complaint DB**: Stores all complaint records and statuses.
      * **Counselor DB**: Keeps counselor details and their responses.

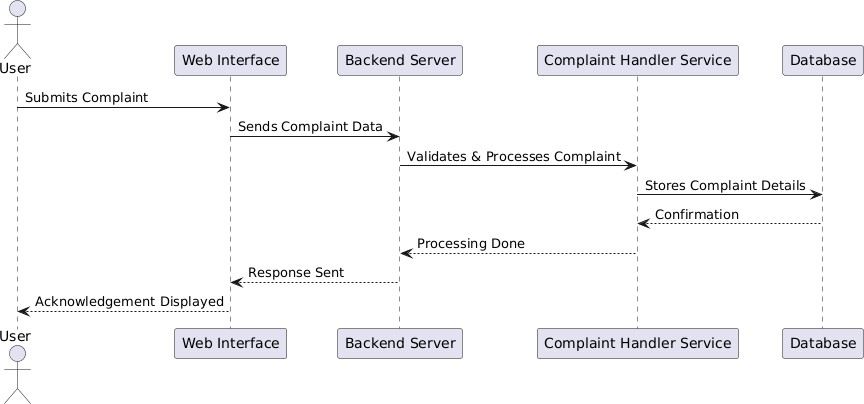
## DEPLOYMENT DIAGRAM

****

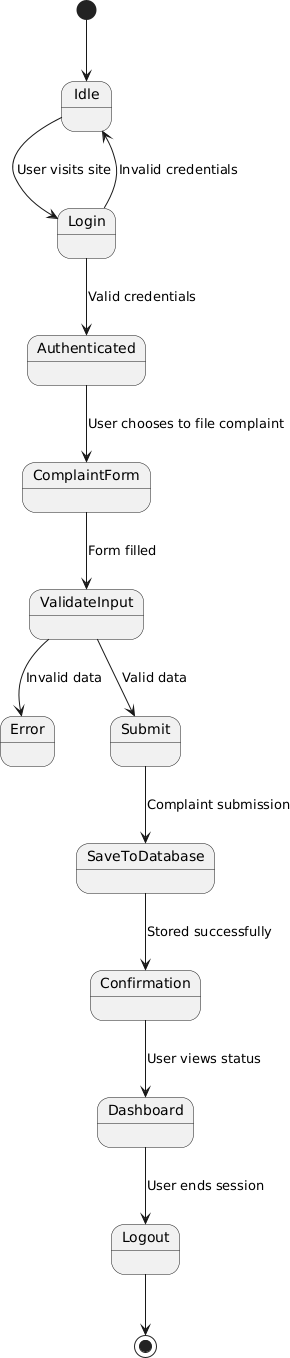
* + - * **Client Device:** Where users access the system via a browser (phone/laptop).
      * **Web Server:** Hosts frontend and APIs, acts as a bridge between UI and backend.
      * **Application Server:** Contains business logic like complaint handling, login, chat, admin tools, etc**.**
      * **Database Server:** Stores data persistently — users, complaints, counselor info, etc
    1. **SEQUENCE DIAGRAM**

****

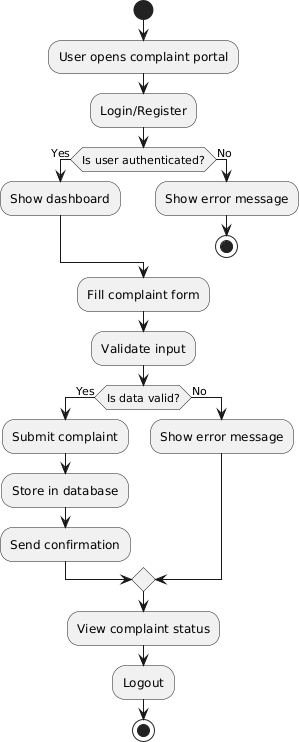
* + 1. **COLLABORATION DIAGRAM**

****

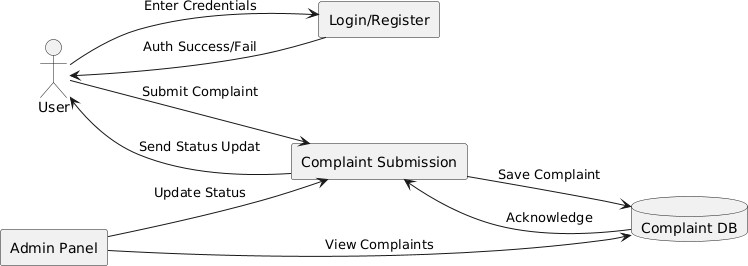
* + 1. **STATE DIAGRAM**

****

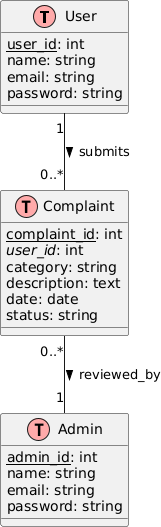
* + 1. **ACTIVITY DIAGRAM**

****

* 1. **DATA FLOW DIAGRAM**

****

* 1. **E-R DIAGRAM**

****

* **User table** → Registers and submits complaints
* **Complaint table** → Linked to user and admin
* **Admin table** → Handles complaint review
* Relationships clearly shown using cardinality (1, 0..\*)

## GUI DESIGN

The Graphical User Interface (GUI) of the Digital Complaint Box is designed to be intuitive and user-friendly, ensuring easy navigation for all users. It includes a straightforward complaint submission form with clearly labeled fields and a prominent submit button to facilitate quick complaint entries. Users can effortlessly track the status of their complaints through a dedicated section. For administrators, a separate panel provides efficient tools for managing, assigning, and updating complaints. The design emphasizes simplicity, clarity, and responsiveness to support seamless usage across devices

## COMPONENTS OF GUI

* + - * Dropdown Menus (for complaint categories)
      * Status Indicators (Pending, In Progress, Resolved)
      * Navigation Menus/Tabs (for switching between sections)
      * Complaint List/Table (for users and admins)
      * Action Buttons (Update Status, Assign Complaint)
      * Attachment Upload Option (optional)
      * Feedback Messages (e.g., “Complaint submitted successfully”)
      * Responsive Layout (for mobile and desktop use)

## FEATURES OF GUI

* + - * User-friendly and intuitive interface
      * Clear and simple complaint submission form
      * Real-time complaint status tracking
      * Easy navigation between different sections
      * Responsive design for mobiles and desktops
      * Secure login for users and admins (if implemented)
      * Color-coded status indicators for easy understanding
      * Attachment upload for adding evidence/photos
      * Quick feedback messages to guide users
      * Admin panel with efficient complaint management tools
  1. **GENERAL**

# CHAPTER 5:

## IMPLEMENTATION

The Digital Complaint Box project is implemented using a user-centric approach, focusing on simplicity and efficiency. The system allows users to submit complaints through a structured form, which are then stored securely in a backend database. Admins can access a dedicated panel to manage, assign, and update complaints. The implementation ensures smooth communication between the frontend interface and backend services, using APIs to fetch and update data in real-time. Emphasis is placed on creating a responsive and accessible design, making the system usable across different devices and platforms. Proper validation and error handling are integrated to maintain data integrity and provide a seamless user experience.

## IMPLEMENTATION

### Index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Digital complaint Box</title>

<link rel="icon" href="logoimage.webp">

<link rel="stylesheet" href="/style.css">

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Atkinson+Hyperlegible+Next:ital,wght @0,200..800;1,200..800&family=Carrois+Gothic+SC&family=Lato:ital,wght@0,100;0,3 00;0,400;0,700;0,900;1,100;1,300;1,400;1,700;1,900&family=Liter&family=Noto+Sans+

Gothic&family=Open+Sans:ital,wght@0,300..800;1,300..800&family=Poppins:ital,wght @0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500

;1,600;1,700;1,800;1,900&family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,9

00;1,100;1,300;1,400;1,500;1,700;1,900&display=swap" rel="stylesheet">

</head>

<body>

<header>

<section class="header">

<div class="logo-name">

<h1>DIGITAL COMPLAINT-BOX</h1>

</div>

<div class="nav">

<nav>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">About us</a></li>

<li><a href="#">Contact</a></li>

<li><a href="#">More</a></li>

</ul>

</nav>

</div>

</section>

</header>

<section class="main-section">

<main class="main-sec">

<!-- quote section -->

<div class="section-div">

<div class="herologo"></div>

</div>

<div class="information">

<h1 class="motive">Motive ?</h1>

<p class="motive-para"><b style="color: black;">"Our website</b> provides a sector-wise categorized digital complaint management system, ensuring efficient redressal of issues related to government services, educational institutions (colleges & schools), and more. By streamlining the complaint process, we connect users with the appropriate authorities,<b style="color: black;">making grievance resolution faster, more transparent, and hassle-free.</b> "</p>

</div>

<div class="how-it-works">

<h3>How its works</h3>

<button class="Explore"><a href="explore.html">Explore</a></button>

</div>

<div class="sector-section">

<div class="governament sector">

<h3>🏛 Government Services</h3>

<p>"Register complaints related to government offices, public services, and administrative issues. Ensure your concerns reach the right authorities."</p>

<button><a href="/government/government">File a Complaint

→</a></button>

</div>

<div class="college sector">

<h3>🏛 College Complaints</h3>

<p>"Raise concerns about college facilities, faculty, exams, administration, or any academic issues. Your feedback helps improve the institution."</p>

<button><a href="/college/college">File a Complaint →</a></button>

</div>

<div class="school sector">

<h3>-'’" • School & College Issues</h3>

<p>"Report concerns related to school and college facilities, faculty, administration, or student services. Help improve the academic environment."</p>

<button><a href="/school/school">File a Complaint →</a></button>

</div>

</div>

</main>

<!-- foter part is start from here -->

<footer class="footer">

<div class="footer-container">

<div class="footer-left">

<h3>Digital Complaint System</h3>

<p>Raise your concerns and help build a better community. Your complaints matter!</p>

</div>

<div class="footer-right">

<h4>Quick Links</h4>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">File a Complaint</a></li>

<li><a href="#">About Us</a></li>

<li><a href="#">Contact</a></li>

</ul>

</div>

</div>

<div class="footer-bottom">

<p>&copy; 2025 Digital Complaint System | All Rights Reserved</p>

</div>

</footer>

</section>

</body>

</html>

### Style.css

\*{

margin: 0;

padding: 0;

border: border box;

font-family: "Open Sans", sans-serif; font-optical-sizing: auto;

font-weight: 600; font-style: normal;

font-variation-settings: "wdth" 100;;

}

body{

/\* background-image: url("complaintlofo.jpg"); \*/ background-size: cover;

overflow: visible;

background: linear-gradient(to right, #e0e7ff, #f8fafc);

}

/\* header part start from here \*/

.header{ margin: 5px;

/\* position: fixed; top: 10px; \*/ height: 100px; width: 100%;

border-bottom-right-radius: 10px; border-bottom-left-radius:10px ; background:#E6E6E6;

display: flex;

justify-content: space-between;

/\* background-image: url("complaintheaderlogo.webp");; background-size: cover; \*/

}

/\* logo name part started from here \*/

.logo-name{ margin-left: 20px; margin-top: 25px; height: 100px; display: flex;

justify-content: center;

background: linear-gradient(135deg, #0602ff, #6610f2);

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

}

/\* nav part started from here \*/

.nav{

margin-top: 15px;

}

.nav nav ul{ height: 100px; width: 500px; display: flex;

justify-content: space-evenly; margin: 10px;

list-style: none;

}

nav ul li a{ opacity: 0.6;

text-decoration: none; color: rgb(0, 0, 0); font-size: 1.38rem; transition: 200ms;

}

nav ul li a:hover{ opacity: 1;

}

/\* media query for responsive look \*/ @media (max-width: 768px) {

.header {

flex-direction: column; height: auto;

padding: 10px;

}

.nav nav ul {

flex-direction: column; align-items: center; width: 100%;

}

nav ul li { margin: 5px 0;

}

}

/\* here section part is starting from here \*/

.main-section{ position: relative;

/\* top: 180px; \*/

height: 100vh; width: 100%;

}

.main-sec{ position: relative; top: 200px; display: flex;

flex-direction: column; justify-content: center; align-items: center; margin-bottom: 25px; height: 100vh;

width: 100%;

/\* background-color: #ccc; border: 2px solid black; \*/ margin-top: 40px;

text-align: center; font-size: 2rem;

}

/\* information section started with here \*/

.information{ height: 300px; width: 100%; display: flex;

flex-direction: column; justify-content: center; align-items: center;

}

.motive{

font-family: "Lato", sans-serif;

font-weight: 700; font-style: normal; color: #B91C1C;

}

.motive-para{ color: #374151;

font-family: "Roboto", sans-serif; font-optical-sizing: auto;

font-weight: 300; font-style: normal;

font-variation-settings: "wdth" 100;

line-height: 1.4; gap: 3px; padding: 8px;

}

/\* hero sectio div \*/

.section-div{ height: 2000px; width: 100%; display: flex;

justify-content: center; align-items: center;

}

.herologo{ height: 500px; width: 500px;

background-image: url("/images/herologoocomplaint.avif"); background-size: cover;

border-radius: 50px;

}

/\* how it works section \*/

.how-it-works{ height: 400px; width: 100%;

/\* border: 2px solid black; \*/

/\* background-color: green; \*/

}

.how-it-works h3{ color:darkblue; font-size: 4rem; margin: 10px;

}

.how-it-works h3{ transition: 200ms;

}

.how-it-works h3:hover{ text-decoration: underline;

}

.Explore{ height: 45px; width: 120px; color: black;

border-radius: 20px; background-color: #fff; font-size: 1.67rem;

transition: transform .4s ease-in-out; cursor: pointer;

border: none;

}

.Explore a{

text-decoration: none; color: black;

}

.Explore:hover{

transform: translate(-10px);

box-shadow: 2px 4px -3px black;

}

.how-it-works button{ background-color:#FFF5E4; text-decoration: none;

}

.step-wise{ height: 200px; width: 100%;

border: 2px solid black; background-color: green;

}

/\* sector section started with here \*/

.sector-section{ height: 600px; width: 100%; display: flex;

flex-direction: row;

justify-content: space-around; align-items: center;

margin: 10px;

}

.sector{

height: 400px;

width: 300px; margin: 10px; border-radius: 20px;

border: 2px solid black;

}

.sector:hover{

box-shadow: 2px 4px 8px black;

}

.governament h3{ font-size: 2rem; margin: 20px; color:#2E86C1;

}

.governament p{ font-size: 1.25rem; margin: 10px;

text-align: start; color: #154360;

}

.governament button {

background-color: #2E86C1; /\* Royal Blue \*/ border: none;

padding: 10px 20px; border-radius: 6px;

transition: background-color 0.3s ease, transform 0.2s ease; cursor: pointer;

}

.governament button a {

text-decoration: none;

font-family: "Liter", sans-serif; font-weight: 400;

font-style: normal; color: white;

display: inline-block;

}

.governament button:hover { background-color: #1B4F72; transform: scale(1.05);

}

.governament button a:hover { color: #F1C40F;

font-weight: bold;

}

/\* college sector start from here \*/

.college h3{

font-size: 2rem; margin: 20px; color: #D35400;

}

.college p{

font-size: 1.25rem; margin: 10px;

text-align: start; color: #873600;

}

.college button {

background-color: #D35400; /\* Dark Orange \*/ border: none;

padding: 10px 20px; border-radius: 6px;

transition: background-color 0.3s ease, transform 0.2s ease; cursor: pointer;

}

.college button a {

text-decoration: none;

font-family: "Liter", sans-serif; font-weight: 400;

font-style: normal; color: white;

display: inline-block;

}

.college button:hover { background-color: #873600; transform: scale(1.05);

}

.college button a:hover { color: #F39C12;

font-weight: bold;

}

/\* school part is start from here \*/

.school h3{

font-size: 2rem; margin: 20px;

color: #28B463;

}

.school p{

font-size: 1.25rem; margin: 10px;

text-align: start; color: #145A32;

}

.school button {

background-color: #28B463; border: none;

padding: 10px 20px; border-radius: 6px;

transition: background-color 0.3s ease, transform 0.2s ease; cursor: pointer;

}

.school button a {

text-decoration: none;

font-family: "Liter", sans-serif; font-weight: 400;

font-style: normal; color: white;

display: inline-block;

}

.school button:hover { background-color: #145A32; transform: scale(1.05);

}

.school button a:hover { color: #A3E4D7;

font-weight: bold;

}

/\* footer part is start from here \*/

.footer {

position: relative; top: 500px; height: 220px; width: 100%;

background-color: #1C1C1C; color: white;

padding: 20px 0; text-align: center;

}

.footer-container { display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap; max-width: 1100px; margin: auto; padding: 10px 20px;

}

.footer-left h3 { font-size: 1.5rem; color: #F39C12;

}

.footer-left p { font-size: 1rem;

color: #CCCCCC;

max-width: 400px;

}

.footer-right h4 { font-size: 1.3rem; color: #3498DB;

}

.footer-right ul { list-style: none; padding: 0;

}

.footer-right ul li { margin: 5px 0;

}

.footer-right ul li a { color: #A3E4D7;

text-decoration: none;

transition: color 0.3s ease-in-out;

}

.footer-right ul li a:hover { color: #F1C40F;

}

.footer-bottom { margin-top: 20px;

border-top: 1px solid #444; padding-top: 10px;

font-size: 0.9rem; color: #BBBBBB;

}

### Package.json

{

"name": "digital-box",

"version": "1.0.0", "main": "electricity.js", "scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"repository": { "type": "git",

"url": "git+https://github.com/codewithghouse/DIGITAL-COMPLAINT-BOX.git"

},

"keywords": [],

"author": "ghouse pasha", "license": "ISC",

"bugs": {

"url": "https://github.com/codewithghouse/DIGITAL-COMPLAINT-BOX/issues"

},

"homepage": "https://github.com/codewithghouse/DIGITAL-COMPLAINT- BOX#readme",

"description": "", "dependencies": { "ejs": "^3.1.10",

"express": "^5.1.0",

"nodemon": "^3.1.10"

}

}

### Index.js

const express= require("express"); const app= express();

const path = require("path"); let port = 3000;

// ejs setting app.set("views","ejs");

app.set("views",path.join( dirname,"views"));

// static files setting

app.use(express.static(path.join( dirname,"public")));

// server making

app.get("/",((req,res)=>{ res.render("index.ejs"); console.log("render the index.html")

}))

//government sector settng app.get("/government/:service", (req, res) => { let {service}= req.params; console.log("service giver successfully"); res.render("governament.ejs",{service});

});

//inside subsector setting app.get("/government/:service/:subservice",((req,res)=>{ let { subservice}= req.params;

if(subservice =="water"){ res.render("water.ejs",{subservice});

}else if(subservice=="electricity"){

res.render("electricity.ejs",{subservice});

} else{

res.send("the page you are finding is no available");

}

// res.render("water.ejs",{subservice});

// console.log("the water page is render ")

}))

//the server is listening in the port app.listen(port,(req,res)=>{

console.log(`the server is listening in the port ${port}`);

});

**init.sql**

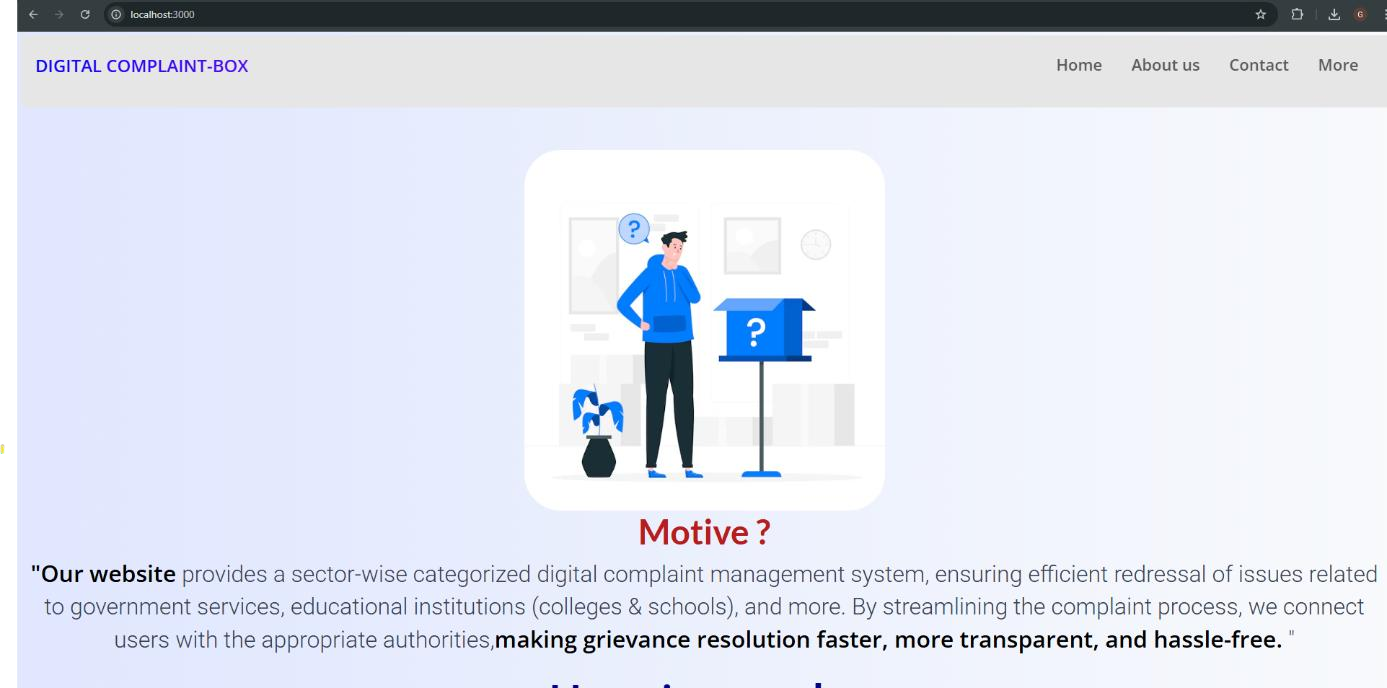
## GENERAL

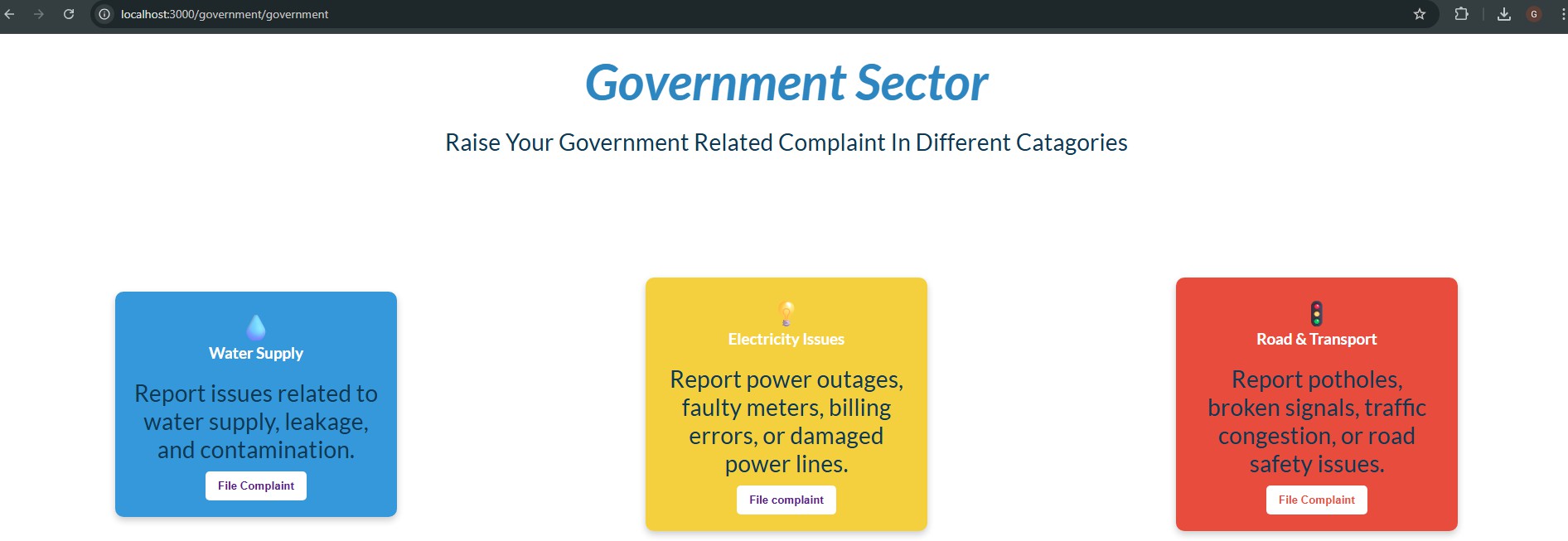
# CHAPTER 6:

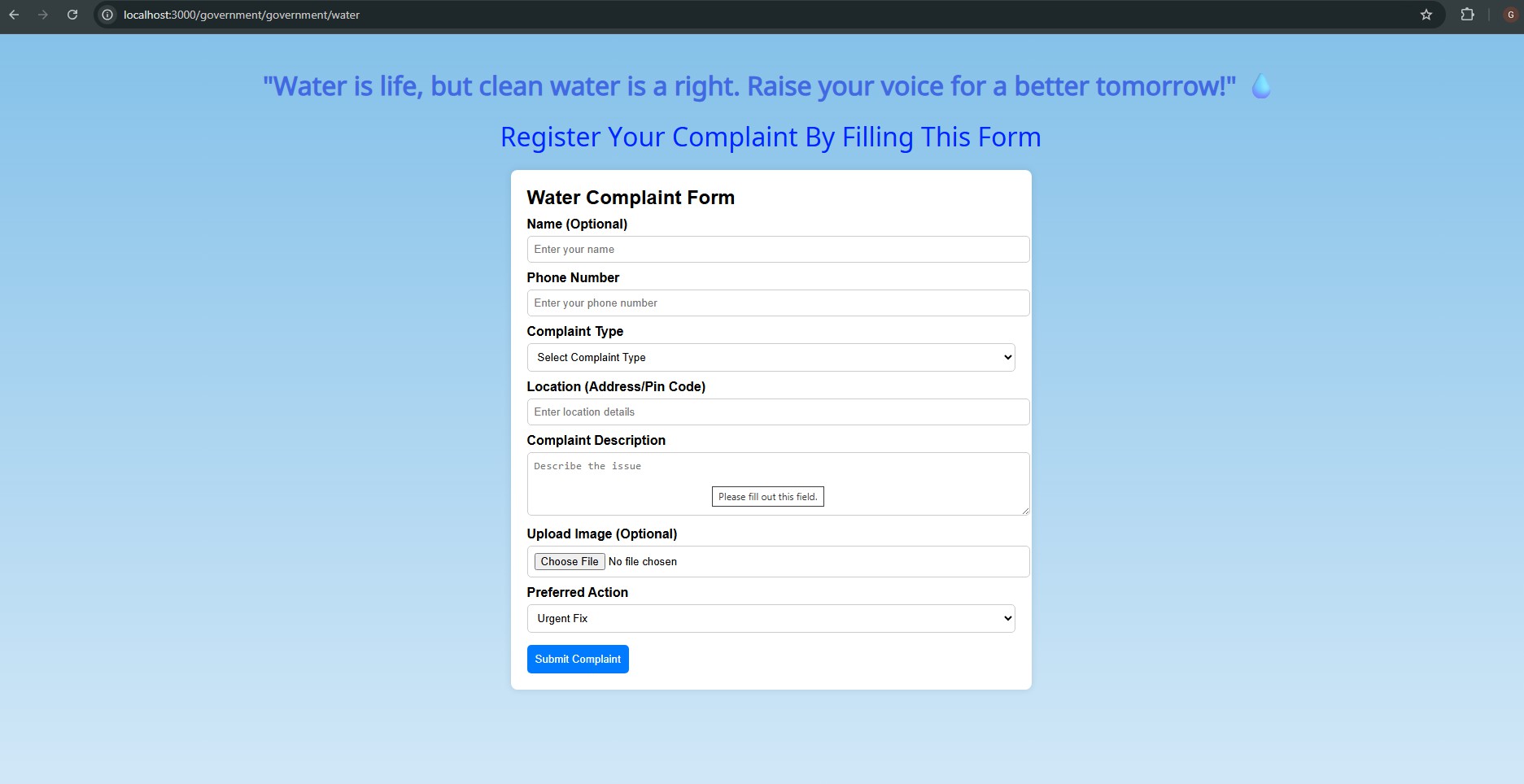
## SNAPSHOTS

This section includes screenshots of the Digital Complaint Box application’s main interfaces. The snapshots provide a visual representation of the user and admin panels, demonstrating the overall design, layout, and key features. They help in understanding the flow of the application and give a clear idea of how the system operates.

## OUTPUT SNAPSHOTS

****



****

## GENERAL

# CHAPTER 7:

## SOFTWARE TESTING

Software testing is performed to ensure that the Digital Complaint Box functions correctly and meets all specified requirements. It involves verifying the usability, functionality, and performance of the application across different devices and browsers. Both manual and automated testing techniques can be used to identify and fix bugs, validate input fields, and ensure smooth user interactions. Testing also includes checking the security of user data and the reliability of complaint submission and tracking features. The goal is to deliver a stable and error-free system for all users.

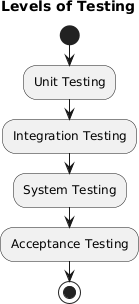
## DEVELOPING METHODOLOGIES

The development process of the Digital Complaint Box follows a structured methodology ensuring systematic progress from design to deployment. Alongside development, multiple testing phases are conducted to guarantee software quality:

* **Unit Testing:** Individual components or functions are tested independently to verify their correctness.
* **Integration Testing:** Different modules are tested together to ensure they work seamlessly as a whole.
* **Security Testing:** The application is tested to identify vulnerabilities and protect user data from unauthorized access.
* **Performance Testing:** The system's responsiveness and stability under different loads are evaluated to maintain smooth operation.
* **End-to-End Testing:** The complete workflow is tested from the user’s perspective to ensure all parts function correctly together.

## TEST STRATEGY

## LEVELS OF TESTING

****

Testing a software application is a multi-layered process to make sure it functions correctly and efficiently at every stage of development. Each level targets specific areas and helps catch bugs early, reducing costly fixes later. The main levels of testing include:

###### Unit Testing:

This is the first level where individual components or functions are tested in isolation. The goal is to ensure that each small part of the application works exactly as expected. It’s usually done by developers using automated tests or manual checks. Unit testing helps catch logical errors early and improves code quality.

###### Integration Testing:

After unit testing, integration testing checks how different modules or components work together. Sometimes, components individually work fine, but when combined, issues like data flow errors or interface mismatches arise. Integration testing verifies smooth communication between modules and ensures that combined parts perform as intended.

###### System Testing:

This level tests the complete and fully integrated application. It verifies that the entire system meets the specified requirements and works correctly as a whole. System testing covers all functional and non- functional requirements including usability, performance, and security. It is usually performed by a specialized testing team.

###### Acceptance Testing:

The final phase before deployment, acceptance testing is done by end- users or clients. It validates whether the system satisfies business needs and is ready for real-world use. This level often involves real-world scenarios and user feedback to confirm the software’s readiness and reliability.

## TYPES OF TESTING

###### Unit Testing

* + - * + Purpose: To verify that individual components or functions perform as expected.
        + Scope: Focused on the smallest parts of the code such as a single function or module. Usually performed by developers during the development phase.

###### Functional Testing

* + - * + Purpose: To check that all application features work according to the specified requirements.
        + Scope: Covers the entire functional area of the application, such as complaint submission, status tracking, and admin operations.

###### Usability Testing

* + - * + Purpose: To ensure that the user interface is easy to use, intuitive, and accessible.
        + Scope: Focuses on user interactions, navigation flow, and overall user experience.

###### Security Testing

* + - * + Purpose: To identify vulnerabilities and protect the application from unauthorized access and data breaches.
        + Scope: Includes authentication, authorization, data encryption, and security modules.

###### Performance Testing

* + - * + Purpose: To evaluate the system’s response time, stability, and scalability under different load conditions.
        + Scope: Tests application speed and resource usage, especially when multiple users use the system simultaneously.

###### Compatibility Testing

* + - * + Purpose: To ensure the application works correctly across different browsers, devices, and operating systems.
        + Scope: Involves cross-browser testing, mobile vs. desktop environments, and various OS platforms.

###### Regression Testing

* + - * + Purpose: To confirm that new code changes or bug fixes do not break existing functionality.
        + Scope: Covers the entire application, with special focus on previously working features.

###### End-to-End Testing

* + - * + Purpose: To test the complete workflow from the user's perspective to ensure smooth operation.
        + Scope: Involves testing the full system process, such as filing a complaint through to its resolution.

## TEST CASE TYPE – GUI

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL**  **No** | **Test Action** | **User Action** | **Expected**  **Output** | **Actual**  **Result** | **Remarks** |
| **1** | Login button visibility | Open login page | Login button should be visible and clickable | As expected | pass |
| **2** | Complaint form visibility | Navigate to complaint submission page | All form fields should be clearly visible | As expected | pass |
| **3** | Submit button behavior | Fill complaint form and click “Submit” | Complaint should be submitted and success message should show | As expected | pass |
| **4** | Error messages display | Submit form without filling required fields | Appropriate error messages should appear | As expected | pass |
| **5** | Responsive design check | Open the  app on different screen sizes | Layout should adjust properly without issu | As expected | pass |

## TEST DESIGN TECHNIQUES

Test design techniques are structured methods used to create effective test cases. These techniques help in identifying test scenarios, reducing redundancy, and improving overall test coverage. Below are the commonly used techniques:

###### Black Box Testing

* + Focuses on the functionality of the application without considering the internal code structure.
  + Testers provide inputs and observe outputs to validate system behavior.
  + Commonly used for UI testing and functionality validation.
  + Example: Submitting a complaint form with valid and invalid data to verify form behavior.

###### White Box Testing

* + Involves testing internal structures or workings of an application.
  + Requires knowledge of code, logic, and internal paths.
  + Typically used for unit testing by developers.
  + Example: Testing the function that processes and stores complaint data.

###### Equivalence Partitioning

* + Divides input data into valid and invalid partitions.
  + Reduces the number of test cases while maintaining effectiveness.
  + Example: Testing complaint ID:
    - Valid: Numeric ID within range.
    - Invalid: Empty or special character input.

###### Boundary Value Analysis

* + Tests input values at the edge of input ranges.
  + Effective for catching off-by-one errors and similar bugs.
  + Example: Checking the maximum length limit for the complaint description field.

###### Error Guessing

* + Based on tester’s intuition and experience to guess likely areas of failure.
  + Helps find issues not covered by formal techniques

## TEST ENVIRONMENT

Setup for Digital Complaint Box

### Component Configuration

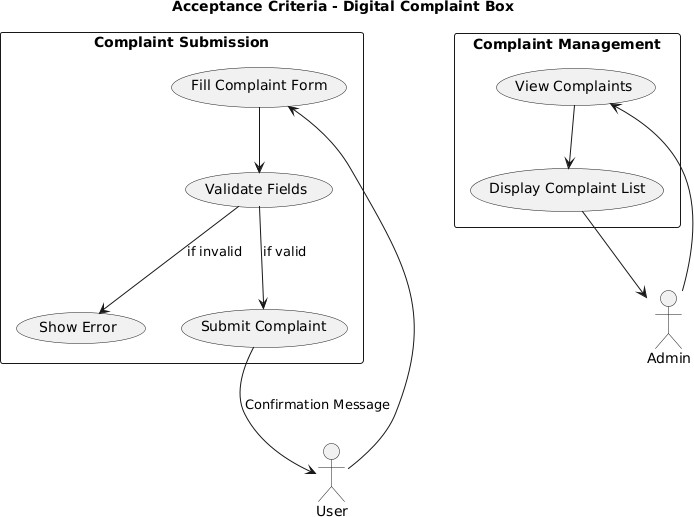
Hardware AWS EC2(4 CPUs ,16GB RAM )

Software Node.js v18, GoogleChrome , Psotman

Network HTTPS via NGNIX,RESTful APIs

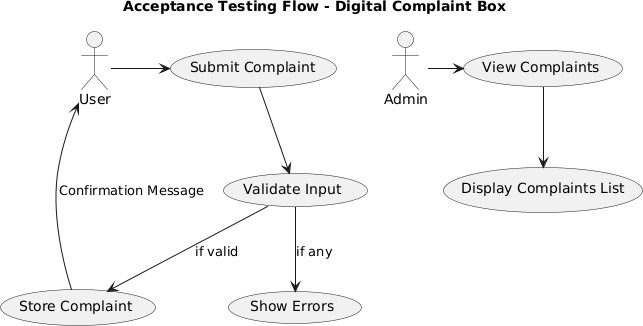
## ACCEPTANCE CRITERIA

* User can submit a complaint form with all required fields filled correctly.
* System validates mandatory fields and shows error messages for missing or invalid data.
* After successful submission, the user receives a confirmation message.
* Admin can view all submitted complaints in a clear, tabular format.
* System stores complaint data securely and allows easy retrieval.
* UI must be responsive and work on both desktop and mobile devices.



## ACCEPTANCE TESTING

* **Purpose:** To verify the system satisfies all user needs and business requirements.
* **Performed By:** End users or clients to validate functionality and usability.
* **Scope:** Includes testing all critical features like complaint submission, validation, and admin complaint listing.
* **Outcome:** System is accepted if it works correctly without major defects; otherwise, issues are reported for fixing.



# CHAPTER 8:

## CONCLUSION AND REFERENCES

## CONCLUSION

**In conclusion**, the *Digital Complaint Box* project serves as an innovative and efficient platform for addressing grievances in a transparent and accessible manner. By offering features such as secure complaint submission, real-time tracking, and streamlined admin handling, it empowers users to voice their concerns confidently. The system promotes accountability and open communication, ensuring that issues are heard and resolved effectively. With a user-friendly interface, data privacy measures, and potential for future enhancements like AI integration and mobile accessibility, the platform stands as a modern solution to bridge the gap between individuals and authorities.

Ultimately, this initiative contributes to creating a more responsive, aware, and digitally connected environment.

## FUTURE ENHANCEMENT

**Future Enhancements for Digital Complaint Box (Web Platform)**

* + 1. **Progressive Web Applications (PWAs):**
       - **Offline Support:** Allow users to draft and view complaints even without internet using service workers.
       - **Push Notifications:** Notify users instantly about complaint status updates or admin responses.
       - **Performance Boost:** Improve load times and smooth access across all devices with smart caching.
    2. **Responsive Design and Mobile Optimization:**
       - **Mobile-First Design:** Ensure the platform works flawlessly on smartphones for easy complaint filing.
       - **Adaptive Layouts:** Adjust UI automatically for tablets, laptops, and desktops.
       - **Efficient Loading:** Use lazy loading and compressed images for fast mobile performance.
    3. **Voice Search and AI Integration:**
       - **Voice-Based Complaint Filing:** Enable voice input for filing complaints for better accessibility.
       - **AI-Powered Suggestions:** Recommend relevant categories as users type complaints.
       - **Chatbot Assistant:** Add a chatbot to assist users 24/7 with common queries or complaint tracking.

## REFERENCES

###### HTML, CSS, JavaScript:

MDN Web Docs, “HTML, CSS, and JavaScript Basics,” <https://developer.mozilla.org/>, Accessed on: 29 May 2025.

###### Node.js:

Node.js Official Documentation, https://nodejs.org/en/docs/, Accessed on: 29 May 2025.

###### Express.js:

Express.js Guide, https://expressjs.com/en/starter/installing.html, Accessed on: 29 May 2025.

###### EJS (Embedded JavaScript):

EJS Official Documentation, <https://ejs.co/>, Accessed on: 29 May 2025.

##### SQL:

W3Schools, “SQL Tutorial,” https:/[/www.w3schools.com/sql/,](http://www.w3schools.com/sql/) Accessed on: 29 May 2025.

###### College Materials:

Lecture notes and project guidance, Shadan College of Engineering and Technology, 2024-2025.