ANONYMOUS COMPLAINT SYSTEM

A Project Report
Submitted in partial fulfilment of the
Requirements for the award of the Degree of
BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Anjali Sureshprasad Rai (1068426)

Dineshkumar S. Kumawat (1068404)

Under the esteemed guidance of
Mrs. Nandini Kadam
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
THE SIA COLLEGE OF HIGHER EDUCATION
Affiliated to University of Mumbai
Re-accredited B+ by NAAC
DOMBIVLI (EAST), 421203
MAHARASHTRA
2023-2024

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PRN: _____ Roll No.: _____

Name of the Student

_____

Title of the Project

_____

Name of the Guide

_____

Teaching experience of the Guide:

_____

Is this your first submission? ☐ Yes ☐ No

Signature of the Student Signature of the Guide

Date: ……. Date: ……...

Signature of the Coordinator

Date: ……

THE SIA COLLEGE OF HIGHER EDUCATION
(Affiliated to University of Mumbai)
Re-accredited B+ by NAAC
DOMBIVLI, MAHARASHTRA 421203

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, **"Anonymous Complaint System**", is bona fide work of **Anjali Sureshprasad Rai** bearing Seat No: **1068426** submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

THE SIA COLLEGE OF HIGHER EDUCATION
(Affiliated to University of Mumbai)
Re-accredited B+ by NAAC
DOMBIVLI, MAHARASHTRA 421203

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, **"Anonymous Complaint System"**, is bona fide work of **Dineshkumar S. Kumawat** bearing Seat No: **1068404** submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

Role and Responsibility

In any project, defining roles and responsibilities among team members helps in clarity, accountability, and efficient coordination. With these designed tasks, our two-person team aims to seamlessly navigate the development and implementation of our project. This ensures focused efforts and optimized collaboration, enhancing our potential for delivering a successful solution.

Team Member: Anjali S. Rai

- **UI/UX Design:** Design the user interface for the webpage, ensuring it is intuitive, user-friendly, and aligned with modern design principles.
- **Front-end Development:** Develop the front-end components of the system, translating design mockups into functional user interfaces.
- **Integration:** Collaborate with project partner to integrate the front-end components with the back-end functionalities.
- **Testing:** Assist in testing the system to ensure proper functionality and user experience across different devices and browsers.
- **Documentation:** Contribute to the creation of documentation, including design specifications and user guides, to support system implementation and usage.

Team Member: Dineshkumar S. Kumawat

- **Project Management:** Oversee the entire project from conception to implementation, ensuring conformity to timelines and objectives.
- **System Analysis and Design:** Conduct an analysis of existing complaint management systems and design the architecture and technical specifications for this project.
- **Development:** Lead the development effort, including coding key modules, such as complaint submission, complaint management, and some user interface.
- **Testing:** Conduct thorough testing of the developed system to identify and rectify any bugs or issues.
- **Documentation:** Prepare comprehensive documentation, including technical specifications and user manuals, to aid in system maintenance and user training.

Abstract

This project introduces an Anonymous Complaint System, designed to help students voice their concerns anonymously. This user-friendly online platform allows students to easily submit their Complaint / Grievance. The system has features like where students can select the right committees for their complaints, ensuring that their issues are directed to the respective committee. In cases where committees need solutions for complaints, this system allows transferring of complaints to different committees and regular teachers for resolution. Some other features include a user-friendly interface and real-time tracking of submitted complaints. This guarantees privacy and trust. This process enhances the efficiency of addressing issues while maintaining student confidentiality. The Anonymous Complaint System empowers students to share their concerns without fear of repercussions.

While making this project it helped in learning new and advance technologies of PHP as well as MySQL. By making use of various logic and using with other technologies like JavaScript.

ACKNOWLEDGEMENT

I extend my heartfelt gratitude to all who contributed to this project. Mere this project is part of our curriculum, but it provided extensive learning. It was not just coursework; it was an opportunity to learn and manage software projects.

Sincerest thanks to my Guide, **Prof. Nandini Kadam**, and other faculty members also deserve the same for their invaluable guidance, patience, and expertise throughout the project. Their mentorship has been instrumental in my learning journey.

Special thanks to our Principal, **Dr. Padmaja Arvind,** who made available the required facility resources required for the project work like infrastructure, access to libraries.

I would also like to mention the unsung support of my parents. The contributions of respected faculty members, friends and peers, whether direct or indirect, played a pivotal role in shaping this project.

DECLARATION

I hereby declare that the project entitled, "**ANONYMOUS COMPLAINT SYSTEM**" done at **The S.I.A. College of Higher Education**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background

In today's educational institutions, the effective management of student and staff complaints is vital for maintaining a harmonious and productive learning environment. These complaints can range from academic concerns to issues related to campus facilities, security, or personal matters. In today’s world most colleges and universities are relying on traditional complaint handling methods, such as paper-based forms or in-person reporting. While these methods serve their purpose, but they also often lack efficiency, transparency, and the ability to ensure the confidentiality of complainants. Recognizing the need for a more streamlined and confidential approach to handling complaints, this project seeks to develop an anonymous complaint-based system website. This project recognized the limitations of the current complaint management system and the desire to improve the overall experience for students and staff in addressing their concerns.

Prior work in the field of complaint management systems has mainly revolved around digitizing the complaint submission process. Several colleges and institutions have implemented online complaint forms, allowing students and staff to submit their concerns electronically. However, many of these systems fall short in terms of ensuring the anonymity of complainants, which can be a significant barrier to reporting sensitive issues.

Furthermore, existing systems often lack comprehensive features for efficient complaint resolution, tracking, and reporting. The need for a more integrated approach that combines user-friendliness, confidentiality, and robust functionality has become increasingly apparent.

1.2 Objectives

This project aims to establish a user-friendly online platform for students to anonymously report complaints regarding issues like college infrastructure damage, exam cheating, and bullying. It includes a committee selection feature to ensure competent and timely resolution, enhancing campus safety and accountability.

1.3 Purpose, Scope, and Applicability

1.3.1 Purpose

The purpose of this project is to address the shortcomings of the current complaint handling system and provide a modern, efficient, and confidential solution. By developing an anonymous complaint-based system, we aim to:

- Facilitate a safer environment for reporting sensitive issues.
- Enhance the efficiency of complaint resolution.
- Improve transparency and accountability in addressing complaints.
- Provide a theoretical framework for a more modern approach to complaint management in educational institutions.

1.3.2 Scope

The scope of this project encompasses various aspects:

- Complaint Submission: Users can submit complaints anonymously via the website.
- Complaint Management: College respective committees can efficiently manage and resolve complaints through a structured process.
- Reporting: The system will provide insights and reports to help college administrators make informed decisions.
- User Interface: The project includes the design and implementation of an intuitive user interface.
- Confidentiality: Ensuring the anonymity of complainants is a core element of the project's scope.

1.3.3 Applicability

The applicability of this project extends to both direct and indirect areas:

Direct Application: The anonymous complaint system will directly benefit the college by improving its complaint management processes and fostering a safer reporting environment.

Indirect Application: This project serves as a model for other educational institutions looking to enhance their complaint management systems. It displays modern technology and design principles for future web development projects.

1.4 Achievements

Upon completion of this project, several significant achievements are expected:

- Improved Complaint Handling: The project aims to enhance the efficiency and confidentiality of complaint handling at the college.
- Enhanced User Experience: Users will have a user-friendly platform to submit their complaints.
- Data-Driven Decision-Making: The system will provide valuable data and insights for informed decision-making.
- Contribution to Best Practices: The project may serve as a reference for other educational institutions seeking to modernize their complaint management systems.

The extent to which these goals are achieved will be assessed throughout the project's implementation and evaluation phases.

1.5 Organization of Report

As Chapter 1 answers almost all the What and Why questions of the project i.e. What is the purpose of this project? What would be the scope? Why this system when already there are many systems? So, after covering the What and Why questions it should be also able to cover How to solve the problem? that answers are covered by rest of the chapters.

In Chapter 2 we analyze the existing system that is being used by institutions to see what weakness are in their system. And we can introduce them our software to cover their weakness. Also introducing various advanced technologies for them as and what needed.

In Chapter 3 we break down the issues with the current system into more sub-categories to resolve them in our system. And define the requirements to address the identified issues. After that how the problems are to be resolved for that planning and scheduling is done. By doing that to create a website we would be required various types of hardware and software to run it. And here we also elaborate more about the expected outcome after using the new system. To illustrate the working of new system we are using various models like Data Flow Diagram, Use Case Diagram, etc.

In Chapter 4 we do the designing of modules that are required like "Complaint Submission", "Complaint Management", "Admin Dashboard," etc. and various functionalities also. Then we move towards User Interface (Front-End) part as this helps in designing schemas for database. But we also have take notion of various things like required tables, fields, relationships, etc. and various logical things as well as security issues.

In Chapter 5 we do implementation of modules by various logical sequence of codes by using different languages like PHP and JavaScript and making them work with each other. After creating modules, integrating them with each other and doing various testing. Testing here is first done as Unit testing and then Integrated testing after properly integrating if there is any modification to be made it should be made. After that again Beta testing is done by limited group of people to know any issue that was not discovered earlier and if there is any modification to be made it should be made as this make seamless experience to the user after making it public.

In Chapter 6 we make test reports with the help of various test cases that we made earlier for our project. As test reports help in making solutions for problems that arises when any module isn't working at the next development/ upgradation due to use of any variable or function name as well as it displays how that functionality is working. After that to make user experience seamless about how to use this complaint system we have added documentation/ User Manual.

In Chapter 7 we have concluded this project by giving short significance about project like what can be achieved after using it and its limitation and future scope. As limitation displays some drawbacks and future scope as these helps in coping limitation to make changes in future necessary.

Chapter 2: System Analysis Existing System

2.1 Existing System

Before using the technologies used in our proposed system, it is crucial to understand the existing complaint management system at our college. The current system primarily relies on manual processes, including paper-based forms, in-person reporting, and email communication. While this system serves its purpose to some extent, it exhibited notable shortcomings:

- Lack of Efficiency: Manual handling of complaints led towards delays in resolution.
- Confidentiality Concerns: Complainants often hesitates to come forward due to fears of non-anonymity.
- Limited Reporting: The system lacks data analysis capabilities for informed decision-making.
- Management issues: Some complaints get misplaced as a result it can create unwanted problems.

Understanding these limitations is crucial in guiding our advanced technology selection process and ensuring that the proposed advanced system effectively addresses these issues.

2.2 Proposed System

Our proposed advanced anonymous complaint-based system aims to use cutting-edge technologies to overcome the deficiencies of the existing system. This system will offer the following advanced key features:

- Web-Based Interface: A highly interactive and responsive web interface accessible to users.
- Anonymous Submission: The ability for complainants to submit their concerns without revealing their identity, ensured by advanced encryption techniques.
- Complaint Management: A simple workflow for college authorities to efficiently handle, track, and resolve complaints.
- User Security: Advanced security measures, including storing of complaints in database in Encrypted way, to protect complainant anonymity and data integrity.

2.3 Survey of Technologies

In this section, we provide an in-depth survey of advanced technologies relevant to our project. These advanced technologies encompass various aspects of web development, database management, and security. The following is a summary of the advanced technologies considered:
Front-end Technologies:

- HTML5 and CSS3: Fundamental for creating the website's advanced structure and styling.
- JavaScript and jQuery UI: Advanced UI libraries for creating advanced user interfaces.
- Chart.js: Displaying various charts for complaints received and managed.

Back-end Technologies:

- PHP: A server-side scripting language known for its versatility and scalability.
- MySQL: An advanced open-source relational database management system (RDBMS) for structured data storage.

Other Technologies:

- Advanced Encryption Standards (AES): High-level encryption to protect sensitive data.
- Microservices Architecture: For enhanced system modularity and scalability.

2.4 Justification of Selection of Technology

The selection of an advanced technology stack for our project is guided by several key factors:

- Advanced Features: We aim to provide advanced features and capabilities that align with our project's objectives, such as data protection in complaint management and real-time actions on complaints.
- User Experience: Using front-end technologies having advanced UI frameworks and ability for real time communication, are chosen to create a highly engaging and responsive user experience.
- Advanced Security: Utilizing encryption standards and intrusion detection systems ensure security and data protection.
- Scalability and Performance: Advanced back-end technologies and database scaling techniques are considered to ensure the system's scalability and performance as the college community grows.

Chapter 3: Requirement & Analysis

3.1 Problem Definition

To tackle this problem, we can divide it into several sub-problems:

Inefficient Complaint Submission:

- Students face barriers in submitting complaints due to the reliance on manual methods.
- Fear of backlash by Friends or Teachers.
- When complaints are submitted, it is difficult to know their status.

Lack of Confidentiality:

- Complainants are hesitant to report sensitive issues due to fears of non-anonymity.
- If complainants are being seen filing a complaint, then it becomes very easy to know who filed a complaint.
- As complaints are written in letter it may be seen by other person who would or unintentionally bully or on in other way make psychological distress to complainant's health.

3.2 Requirement Specification

The Requirements Specification describes the actions that should be taken to overcome the existing problems.

- Creating an easy-to-use web-based platform for submitting complaints.
- Implementing a user-friendly interface to reduce barriers to complaint submission.
- System generated and unique Complaint number to know the status.
- Developing a system that allows complainants to submit complaints anonymously.
- Employing advanced encryption techniques to protect complainant identity.
- Only assigned users can see the complaints.

3.3 Planning & Scheduling

Planning and scheduling are essential for any successful software development. As we should know what tasks are to do and how. If any hurdle arises then take proactive measures to maintain the project timeline.

Task 1: Requirement analysis, Research for technology selection and documentation:

As this is the initial phase it should take up to at least 3 weeks for information gathering and selection of project. Also looking upon the existing system and available resources for completion of project. And properly documenting tasks to keep up with the doings as it helps in future when needed for updating with the help of Notion. I think if we schedule tasks it helps in doing our project at earliest.

Task 2: User interface design:

Here with the help of Figma, designing starts as it helps in overview of how the Front-End of project should be looking. It would take up to 2 weeks to visualize the wireframe for this project.

Task 3: Front-end development:

As here actual project implementation starts using HTML, CSS, and JavaScript it may take 7 weeks to develop:

- Form for Complaint Submission and Complaint Status.
- Dashboard for Admin.
- Dashboard for Committees.
- Dashboard for Teachers.

Task 4: Back-end development.

As Front-end comes in around 75% completion we can go for connecting Front-End to Back-End using PHP and MySQL so this phase would take 10 weeks for developing:

- Code in PHP for scripting.
- Schemas for Table.
- Constraints and Datatypes to be Stored in Table.

Task 5: Security implementation.

After completion of previous tasks, for security measures we have to give 2 weeks to avoid any form of data leak or open access. As this is an important integral part of this project.

Task 6: Testing and quality assurance.

Here we do tests on all the modules we created as to see whether they are working as they are expected with tools like Selenium IDE and Burp Suite. Using these tools helps to know if there is a need for changes in the code so we have to allocate 4 weeks.

Task 7: System integration.

Here all the modules are integrated into each other. Again, testing and quality is checked for the whole system. And if needed, necessary changes are done in them before deploying it to the server so again we have to allocate 3 weeks for proper working of all modules and functionalities.

Task 8: Deployment.

Here the project is deployed on server to make it accessible for filing a complaint by students. But here also necessary changes are made to server files for accessibility so giving 2 weeks is enough.

Gantt chart:

A Gantt chart is a project management tool that illustrates work completed over a period of time in relation to the time planned for the work. It typically includes a two sections list of tasks and timeline with schedule bars that visualize work.

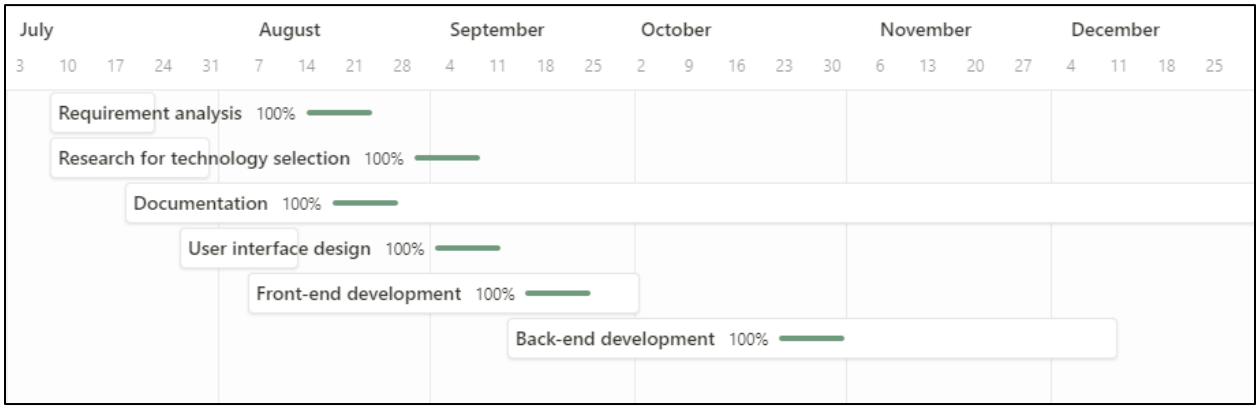


Fig. 3.1: GANTT Chart (1)

Fig. 3.2: GANTT Chart (2)

3.4 Software and Hardware Requirement:

3.4.1 Software Requirement

For Development

Table 3.1: Software Requirement for Development

Operating System:	Windows/ Kali Linux (Latest Version)
Browser	Firefox Browser Developer Edition
Coding Language	Front-End: HTML, CSS, JavaScript Back-End: PHP, MariaDB
Tools:	Microsoft Visual Studio Code (Coding purpose) Microsoft Word (Documentation purpose) Notion (Keeping record and managing Tasks) StarUML (Drawing Models) Figma (Designing) Selenium WebDriver and Burp Suite Community Edition (Testing) XAMPP

For Implementation

Table 3.2: Software Requirement for Implementation

Operating System:	Kali Linux/ Ubuntu (linux based only)
Browser	Google Chrome/ Firefox ESR
Language	PHP (7.4.33),
Tools:	cPanel (For Site management) FileZilla (For transferring files) MySQL (7.4.33) Apache HTTP Server (2.4.57) phpMyAdmin (4.9.11)

3.4.2 Hardware Requirement

For Development

Table 3.3: Hardware Requirement for Development

RAM:	16 GB
Storage:	500 GB NVMe SSD
Processor:	Any 3.4 GHz with 8 Cores
Peripherals	Keyboard Mouse Monitor (1920 x 1080 Pixels) Internet Connection

For Implementation

Table 3.4: Hardware Requirement for Implementation

RAM:	Minimum 512 MB
Storage:	Minimum 128 GB SSD
Processor:	Any 3.4 GHz with 8 Cores
Peripherals	Keyboard Mouse Monitor (1920 x 1080 Pixels) Internet Connection

3.5 Preliminary Product Description

As we have already seen the requirements and objectives of these systems, now we can see the functions and operations of this system.

Functions in this system are:

- Unique Complaint Number Generation.
- Assigned Committee Member only can see complaints.
- Students can see the status of their complaint.
- Committee Member can easily transfer complaints to another Member.
- Dashboard and Roles would be different as assigned by Admin.

This system works as an intermediatory between Students and College authorities to make their voice heard through a proper channel. There are various operations in this system like:

- Filing complaints by selecting concerned Committee.
- Able to see Complaint Status; as well as showing status like received/ pending/ closed and remarks if provided.
- Login ID and Password creation by Admin.
- Admin can “Create User”, “Update User”, and “Delete User”.
- Admin only can change role of User for ease of complaint management.
- Committee Member can act on complaints, and it would update status of complaints.
- If the Committee Member cannot provide resolution or need assistance it can be transferred.
- If Students mistakenly selects different Committee, then Committee Member can send to the appropriate committee with the help of Admin.

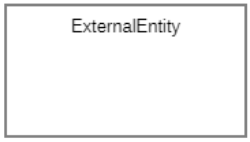
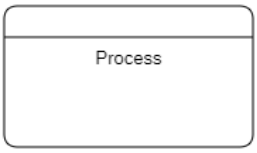
3.6 Conceptual Models:

3.6.1 Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one.

The symbols depict the four components of data flow diagrams:

Table 3.5: Data Flow Diagram

	**External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
	**Process:** any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as "Submit payment."
	**Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."
	**Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."

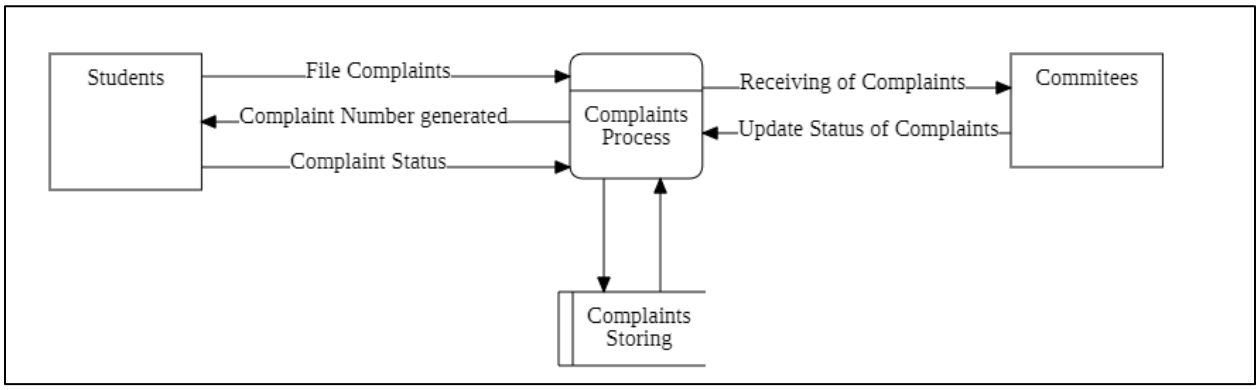


Fig. 3.3: 0-Level DFD

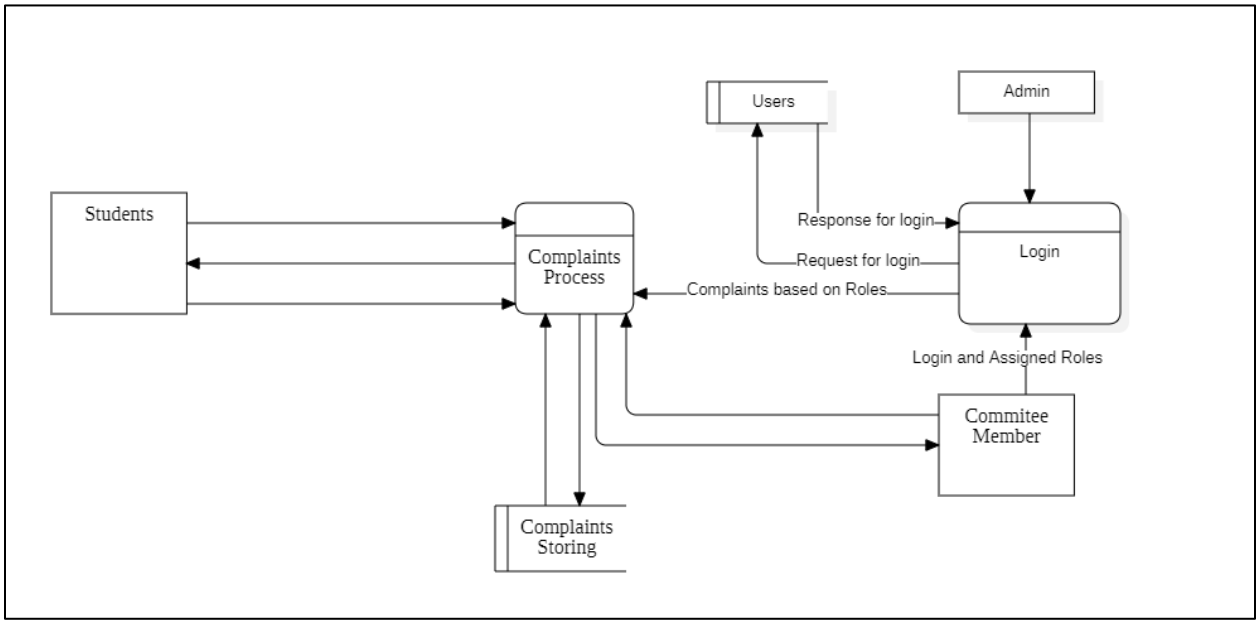


Fig. 3.4: 1-Level DFD

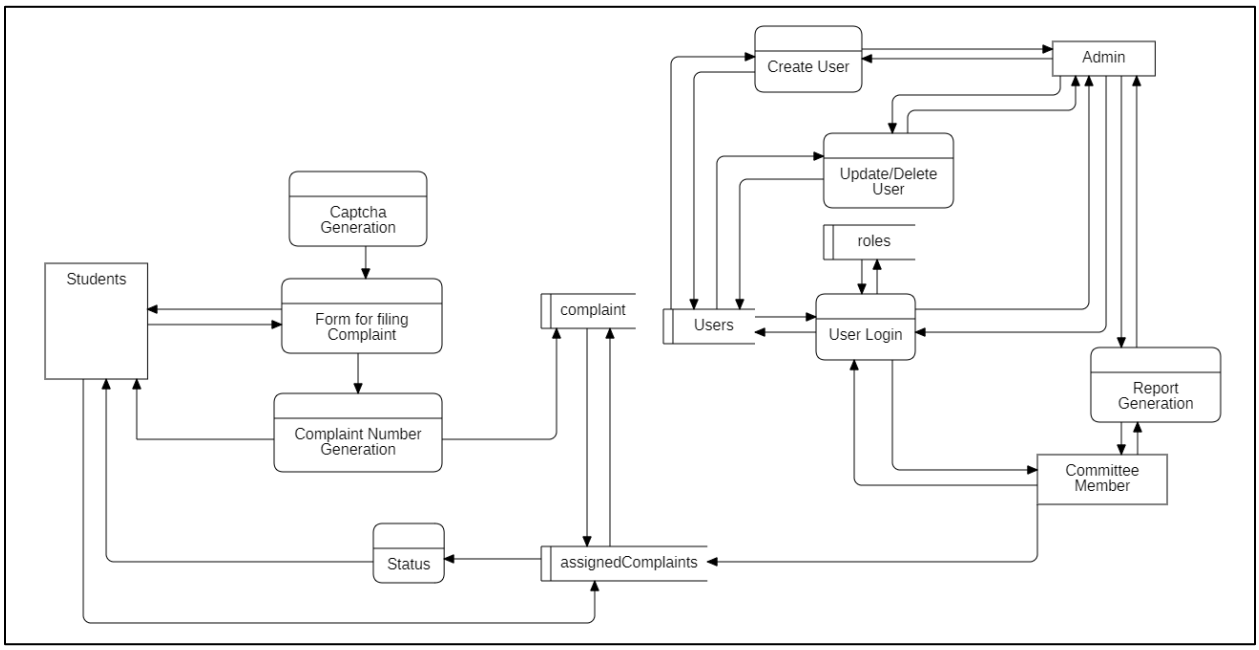


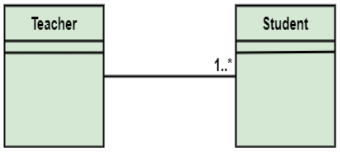
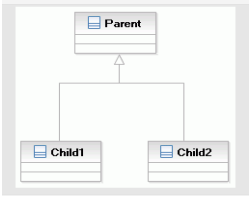
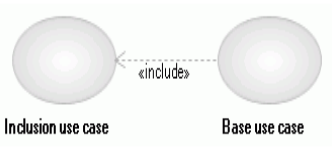
Fig. 3.5: 2-Level DFD

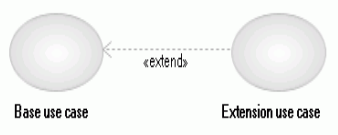
3.6.2 Use Case Diagram

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

The symbols depict the components of Use Case Diagram:

Table 3.6: Use Case Diagram

	**Use case**: A use case describes a function that a system performs to achieve the user's goal. A use case must yield an observable result that is of value to the user of the system.
	Actor: An actor represents a role of a user that interacts with the system that you are modeling. The user can be a human user, an organization, a machine, or another external system.
	**Association relationships**: In UML models, an association is a relationship between two classifiers, such as classes or use cases, that describes the reasons for the relationship and the rules that govern the relationship.
	Generalization relationships: In UML modeling, a generalization relationship is a relationship in which one model element (the child) is based on another model element (the parent). Generalization relationships are used in class, component, deployment, and use-case diagrams to indicate that the child receives all of the attributes, operations, and relationships that are defined in the parent.
	Include relationships In UML modeling, an include relationship is a relationship in which one use case (the base use case) includes the functionality of another use case (the inclusion use case). The include relationship supports the reuse of functionality in a use-case model.

	Extend relationships In UML modeling, you can use an extend relationship to specify that one use case (extension) extends the behavior of another use case (base). This type of relationship reveals details about a system or application that are typically hidden in a use case.

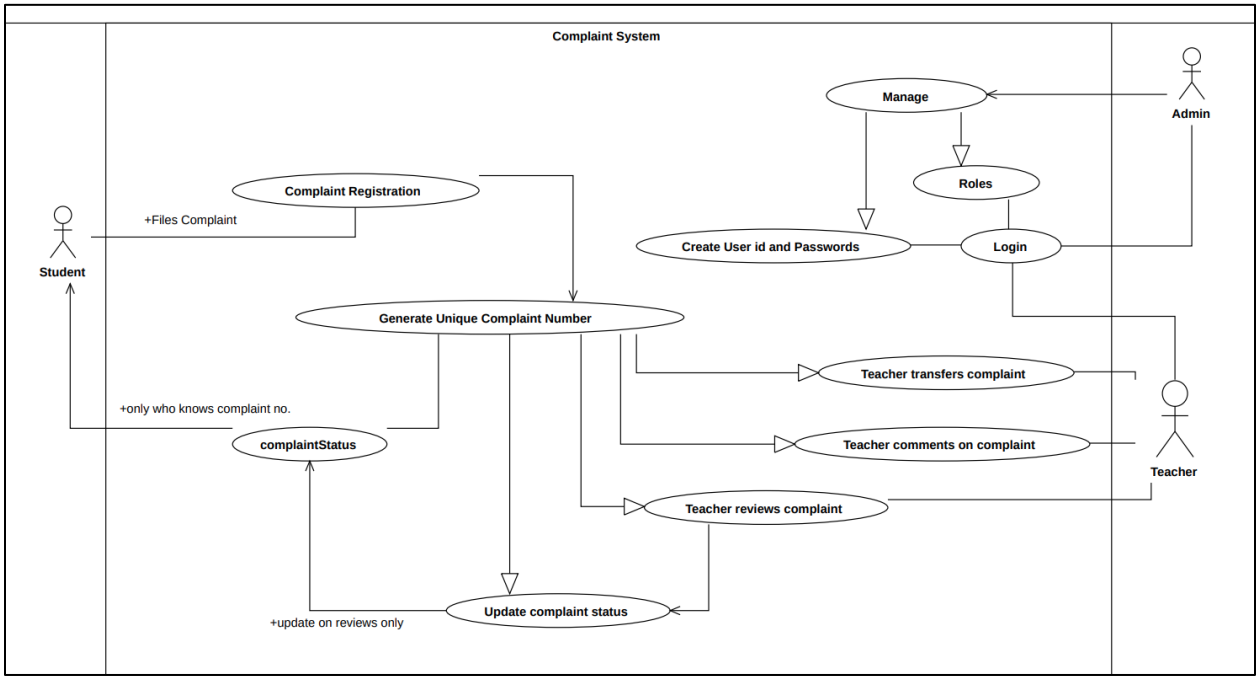


Fig. 3.6: Use Case Diagram

3.6.3 Entity Relationship Diagram

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

The symbols depict the components of Entity Relationship Diagram:

Entity: An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

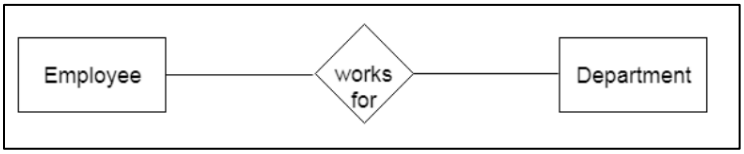


Fig. 3.7: Entity in ERD

Attribute: The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

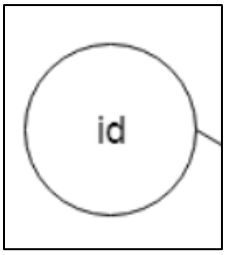


Fig. 3.8: Attribute in ERD

Key Attribute: The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.

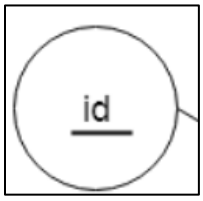


Fig. 3.9: Key Attribute in ERD

Composite Attribute: An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

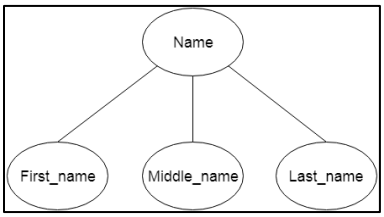


Fig. 3.10: Composite Attribute in ERD

Relationship: A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

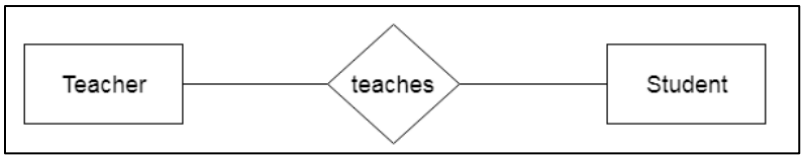


Fig. 3.11: Relationship in ERD

One-to-One Relationship: When only one instance of an entity is associated with the relationship, then it is known as one-to-one relationship.

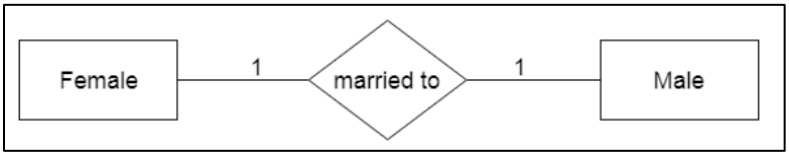


Fig. 3.12: One-to-One Relationship in ERD

One-to-many relationship**:** When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

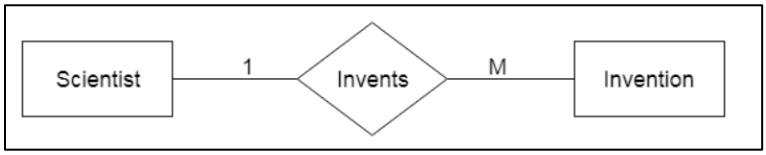


Fig. 3.13: One-to-Many Relationship in ERD

Many-to-one relationship: When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

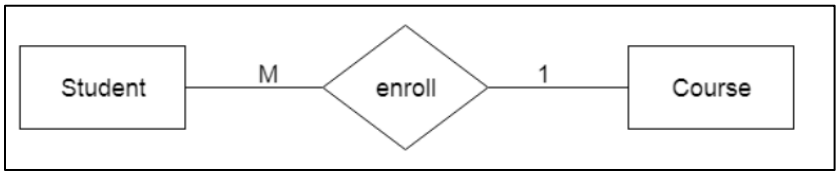


Fig. 3.14: Many-to-One Relationship in ERD

Many-to-many relationship: When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

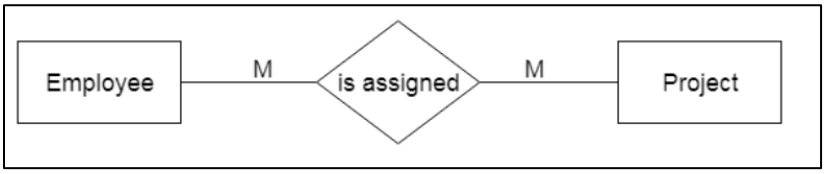


Fig. 3.15: Many-to-Many Relationship in ERD

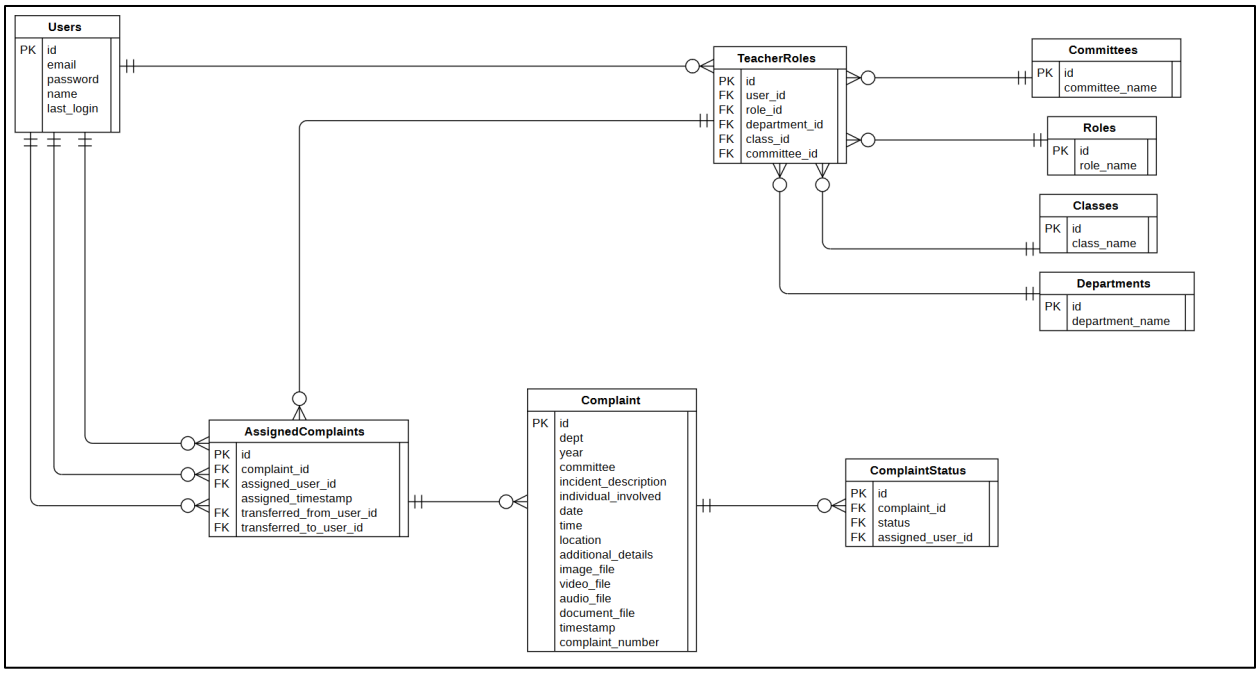


Fig. 3.16: ERD for Anonymous Complaint System

Chapter 4: System Design

4.1 Basic Modules:

Complaint Submission:

In this module, users can submit complaints anonymously. It includes forms for providing complaint details, options for Committee selection, and uploading any supporting documents.

Admin Dashboard:

Administrators have access to an administrative dashboard for system configuration, user management, and monitoring.

Complaint Management:

College authorities can access this module to manage complaints. Features include complaint assignment, status tracking, communication with complainants, and resolution updates.

Data Analysis and Reporting:

This module enables authorized users to generate reports and insights from complaint data. Users can specify parameters for report generation and view visualizations of complaint trends.

4.2 Data Design:

4.2.1 Schema Design:

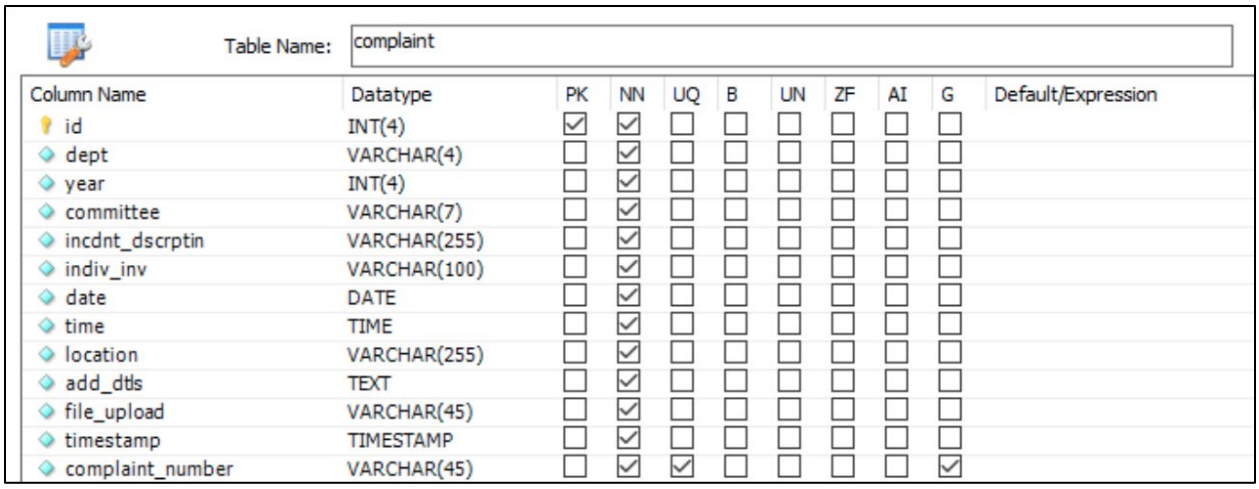


Table Name: complaint

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(4)	☑	☑	☐	☐	☐	☐	☐	☐	
dept	VARCHAR(4)	☐	☑	☐	☐	☐	☐	☐	☐	
year	INT(4)	☐	☑	☐	☐	☐	☐	☐	☐	
committee	VARCHAR(7)	☐	☑	☐	☐	☐	☐	☐	☐	
incdnt_dscrptin	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
indiv_inv	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
date	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
time	TIME	☐	☑	☐	☐	☐	☐	☐	☐	
location	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
add_dtls	TEXT	☐	☑	☐	☐	☐	☐	☐	☐	
file_upload	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
timestamp	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	
complaint_number	VARCHAR(45)	☐	☑	☑	☐	☐	☐	☐	☑	

Fig. 4.1: Schema for complaint Table

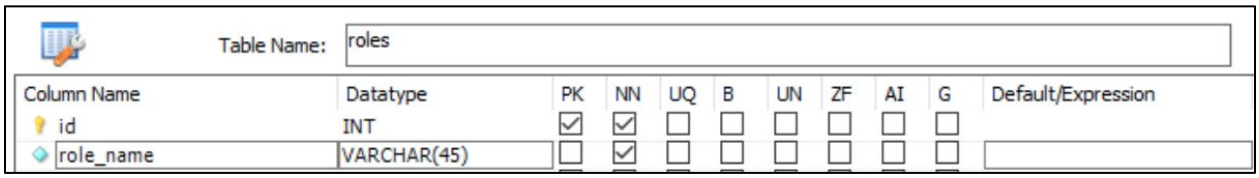


Table Name: roles

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☑	☑	☐	☐	☐	☐	☐	☐	
role_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	

Fig. 4.2: Schema for roles Table

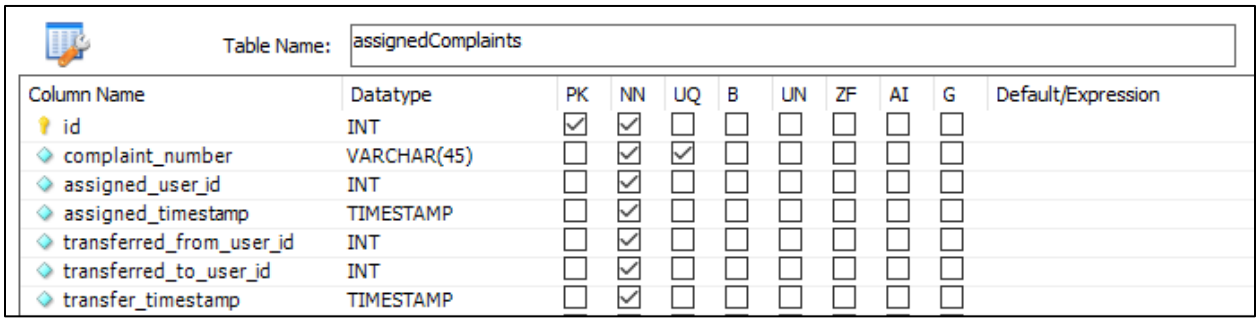


Table Name: assignedComplaints

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☑	☑	☐	☐	☐	☐	☐	☐	
complaint_number	VARCHAR(45)	☐	☑	☑	☐	☐	☐	☐	☐	
assigned_user_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
assigned_timestamp	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	
transferred_from_user_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
transferred_to_user_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
transfer_timestamp	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	

Fig. 4.3: Schema for assignedComplaints Table

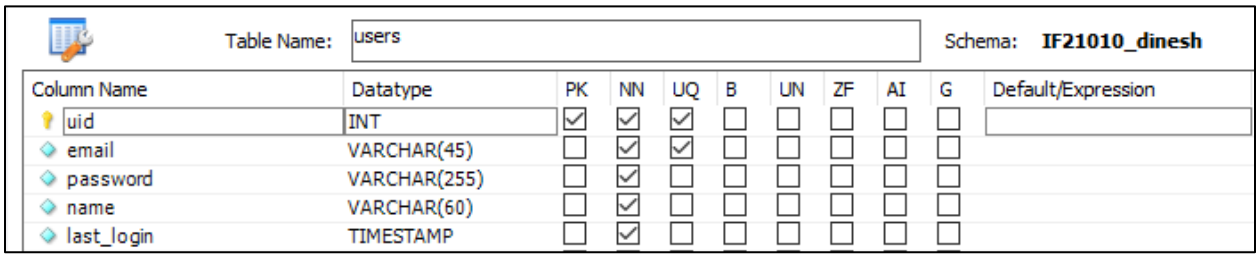


Fig. 4.4: Schema for users Table

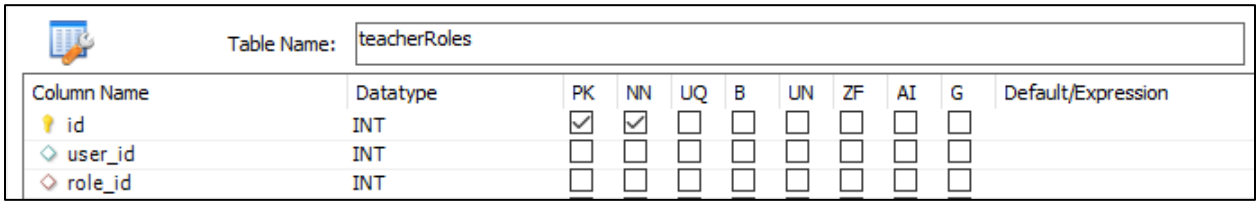


Fig. 4.5: Schema for teacherRoles Table

4.2.2 Data Integrity and Constraints:

Table 4.1: Data Integrity and Constraints

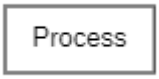
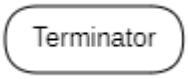
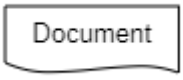
Table	Field Name	Specification
users	email	It is marked as NOT NULL and has a UNIQUE KEY, ensuring that each user has a distinct and non-null email address.
	password	It is marked as NOT NULL, ensuring that these fields are mandatory.
	name	
	last_login	It is a TIMESTAMP that records the last login time of users.
roles	role_name	It is marked as NOT NULL, ensuring that each role has a name.
teacherRoles	user_id	These are foreign keys linking to the Users and Roles tables to maintain data integrity.
	role_id	
complaintStatus	complaint_id	These are foreign keys linked to Complaints and Users tables for maintaining data integrity.
	assigned_user_id	
	status_user_id	
	timestamp	It is a TIMESTAMP that records the time when any action is taken on complaints.
assignedComplaints	complaint_id	These are foreign keys related to complaints and users tables.
	assigned_user_id	
	transferred_from_user_id	These are foreign keys linked to users table for tracking transfer of complaints.
	transferred_to_user_id	
	assigned_timestamp	These create timestamp to record the assignment and transfer of complaints.
	transfer_timestamp	
complaint	complaint_number	It is a unique identifier for each complaint, ensuring its uniqueness and it NOT NULL.
	timestamp	It generate value of the current timestamp when submitting Complaint.

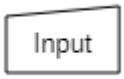
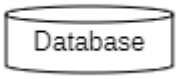
4.3 Procedural Design:

4.3.1 Logic Diagrams:

To illustrate Logic Diagram, we use Flow Chart as its simple to understand and depicts a process, system or computer algorithm. It is widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams.

Table 4.2: Flow Chart

	Flow: This symbol represents flow of shapes in between them.
	Process: Represented as a rectangle, the process or action symbol shows a specific process, action or function. It can help show the basic tasks or actions that need completing.
	Terminator: This symbol represents the starting and end points and the potential outcomes of a process. It typically functions as an elongated oval with a single starting and ending point. It may contain the word "Start" or "End" within the symbol itself.
	Decision: Shaped as a rhombus, this symbol helps indicate a question that results in a "yes" or "no" answer, in addition to a possible "true" or "false" situation. Depending on the answer to the proposed question, the flowchart can then separate into various branches to complete the outline of the workflow.
	Delay: A delay symbol is an elongated half-oval that indicates a delay within a process. Programmers may use this symbol to indicate the specific length of a delay within the software development process.
	Document: This symbol is a rectangle with its bottom side in a wave, representing the input or output of a document. For example, you might use this symbol to outline a document input, including receiving an email or report. You can also use it to represent a document output, which might include producing a presentation or project.

	Display: A display symbol within a flowchart indicates where the chart is to display the data within a process to a user. It has a shape similar to a delay symbol and represents the flow of information within a process. It can be useful for complex processes that require user input.
	Manual Input: This flowchart symbol represents the manual input of data, such as typing data into a specific field or form. For example, if you sign up for a new email account, the login fields require you to enter your data manually.
	Database: This cylinder-shaped flowchart symbol represents stored data. This data often allows for user searches and filtering capabilities. For example, it may represent the data within a real estate app that allows users to search and filter results by price, location and other search criteria.
	Stored Data: Indicates a step where data gets stored.

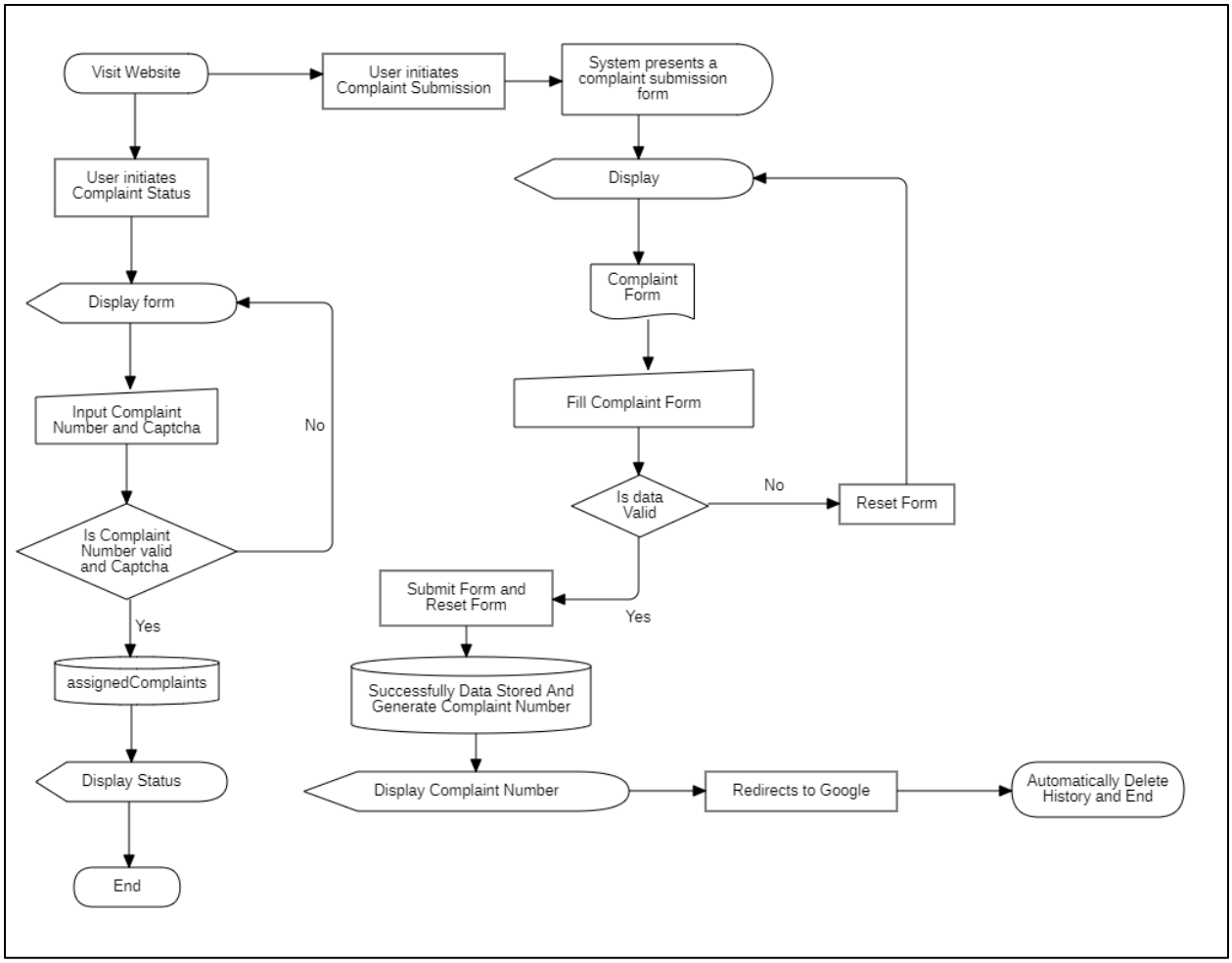


Fig. 4.6: Flow Chart for Complaint Submission

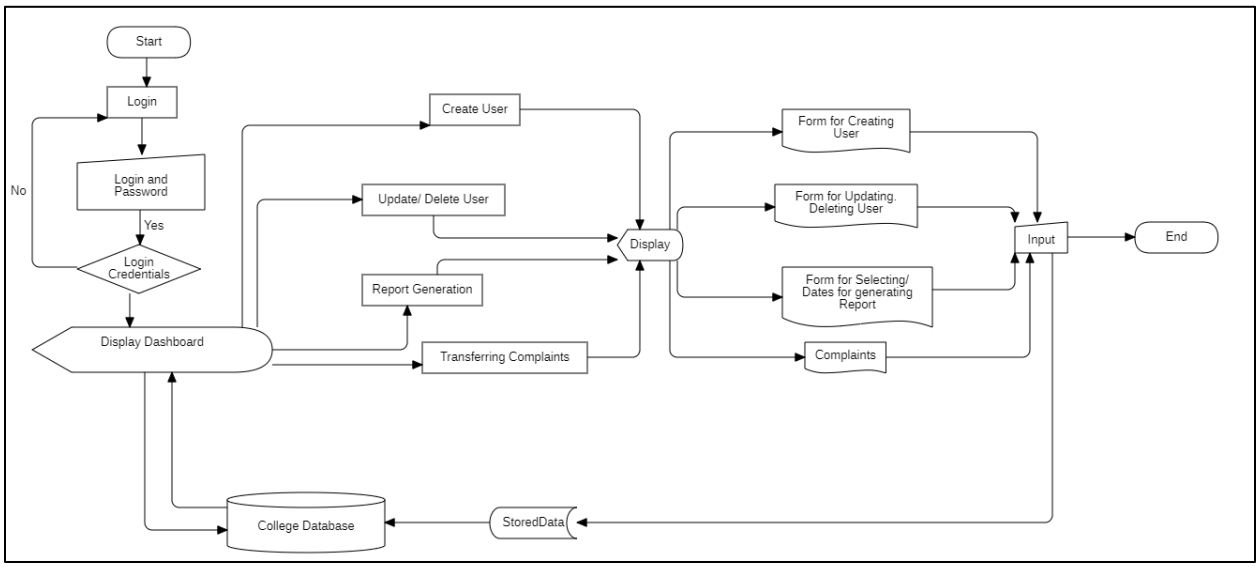


Fig. 4.7: Flow Chart for Admin

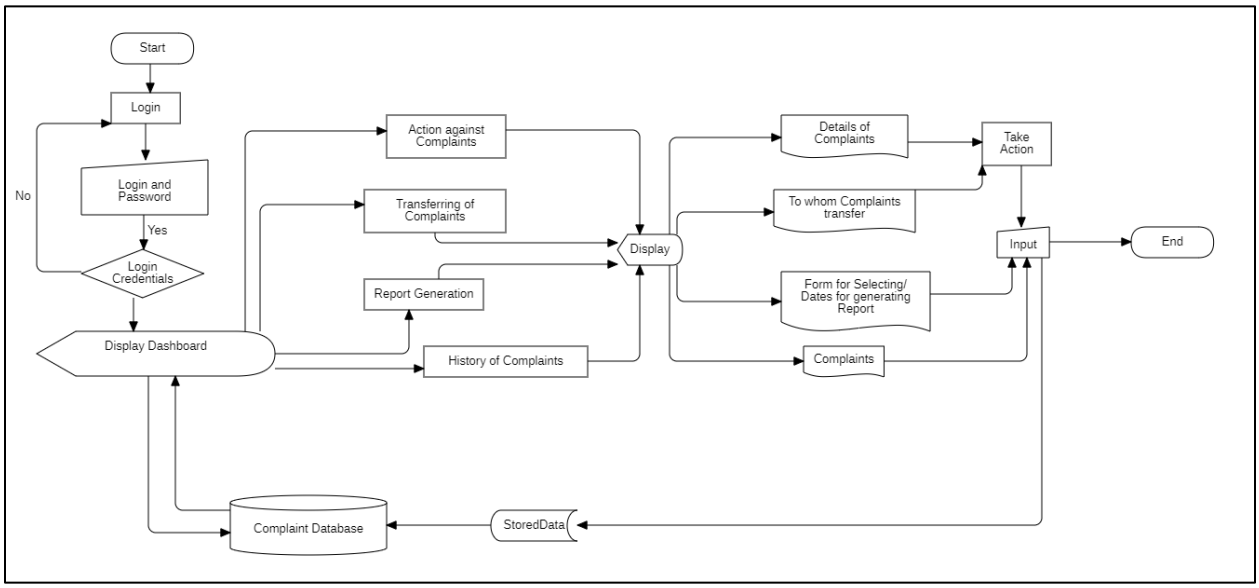


Fig. 4.8: Flow Chart for Committee Member

4.3.2 Algorithms Design:

Algorithm for all modules

Complaint Submission Module:

Input:

Complaint Details

Supporting Documents (optional)

Captcha

Output:

- Confirmation Message

If complaint validation is done generate

ComplaintID = GenerateUniqueID()

Store ComplaintDetails and ComplaintID

If SupportingDocuments exist:

Store SupportingDocuments

Return ConfirmationMessage and ComplaintID

Login and Dashboard Module:

Input:

LoginCredentials

Output:

Check authentication for Credentials

If Authentication Successful:

Provide Access to Admin/ Committee Member Dashboard

Display Admin/ Committee Member Functions and Controls

Return AccessGrantedMessage

Else:

Display AuthenticationFailedMessage

Complaint Management Module:

Input:

Select Complaint

Output:

Display ComplaintData

AssignedComplaints

ComplaintStatusUpdates

Transferring of Complaints (if needed)

For each Complaint in ComplaintData:
Assign Complaint to another Member if needed
Update ComplaintStatus

Report Generation Module:

Input:

ReportParameters

Select Date

Select Committee (Role based)

Output:

GeneratedReport

4.4 User Interface Design:

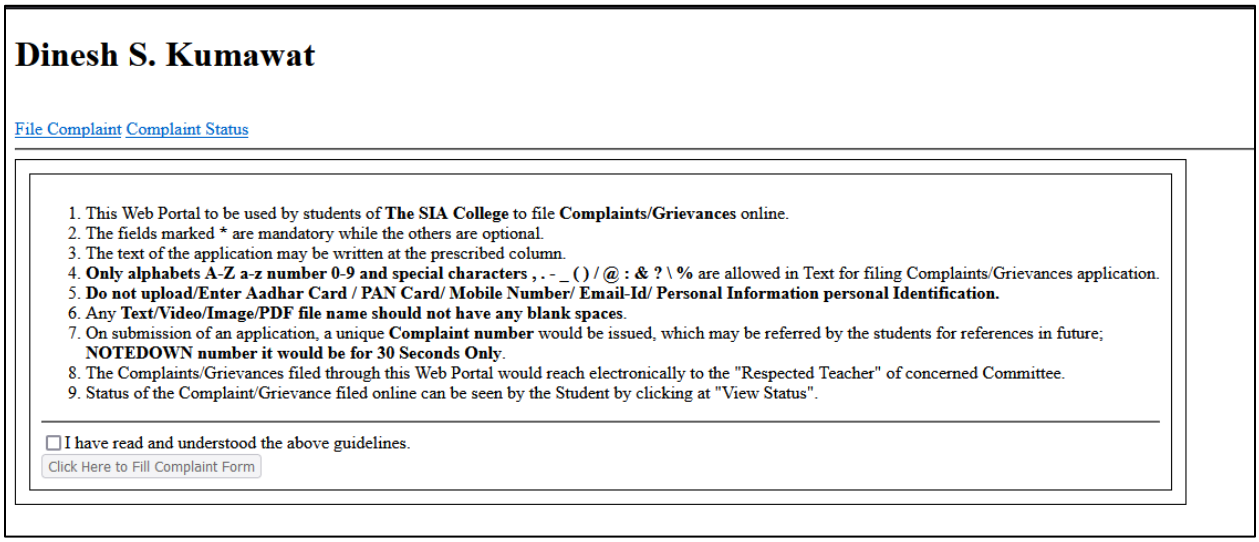


Fig. 4.9: Index page for Complaint Submission (Sketch)

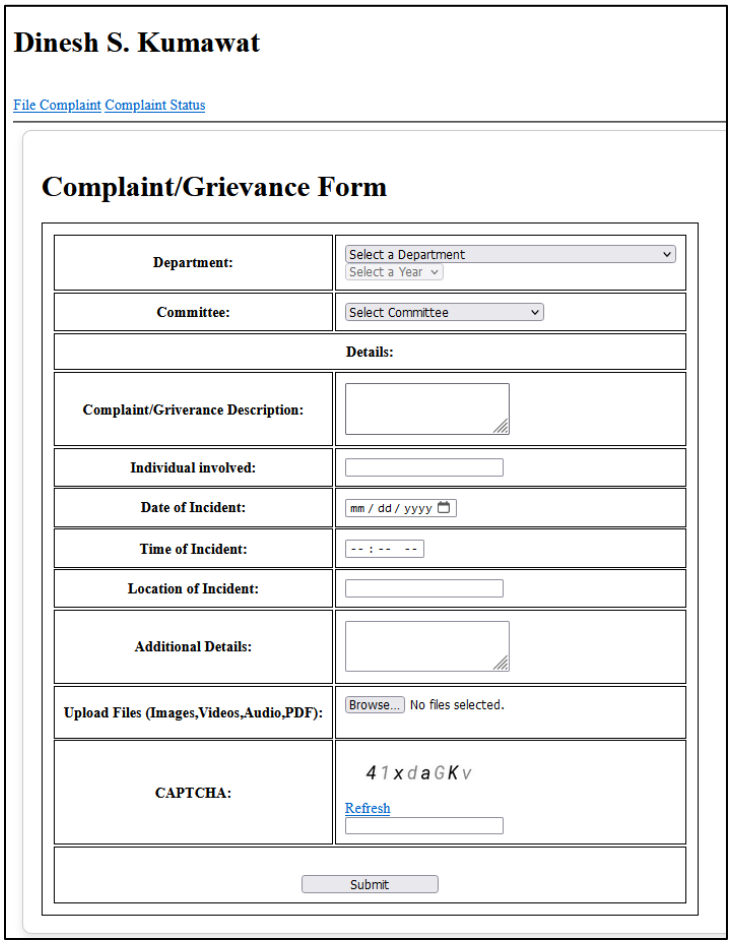


Fig. 4.10: Form for filing Complaint (Sketch)

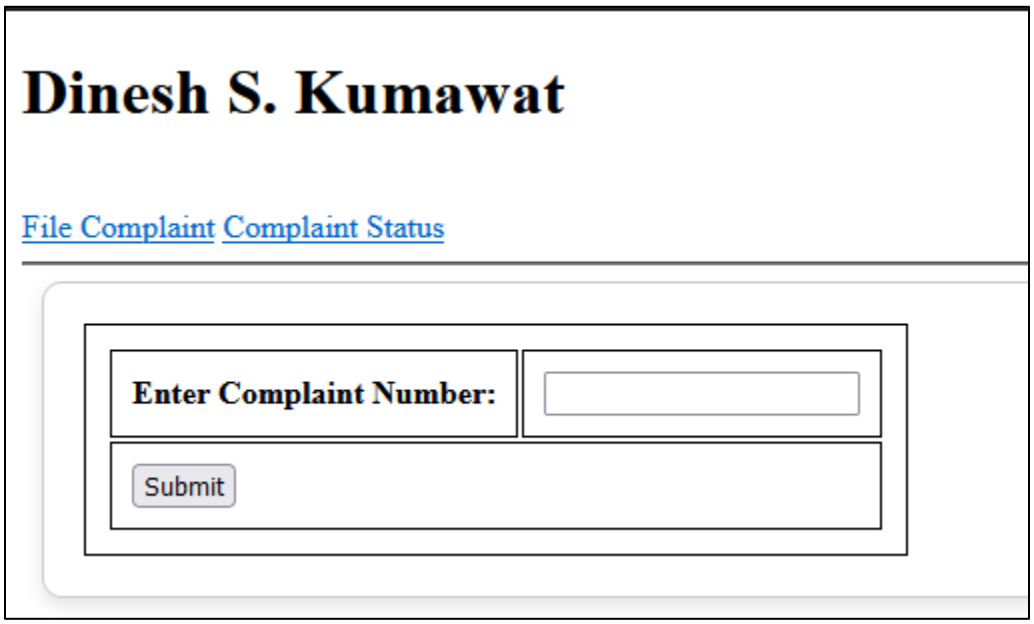


Fig. 4.11: Form to check Status (Sketch)

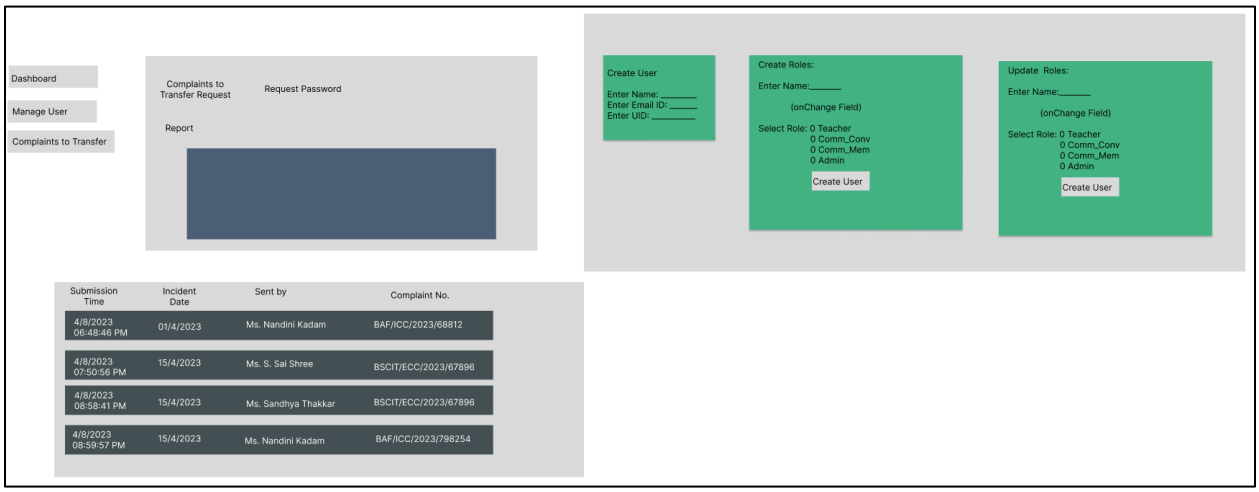


Fig. 4.12: Admin Dashboard (Sketch)

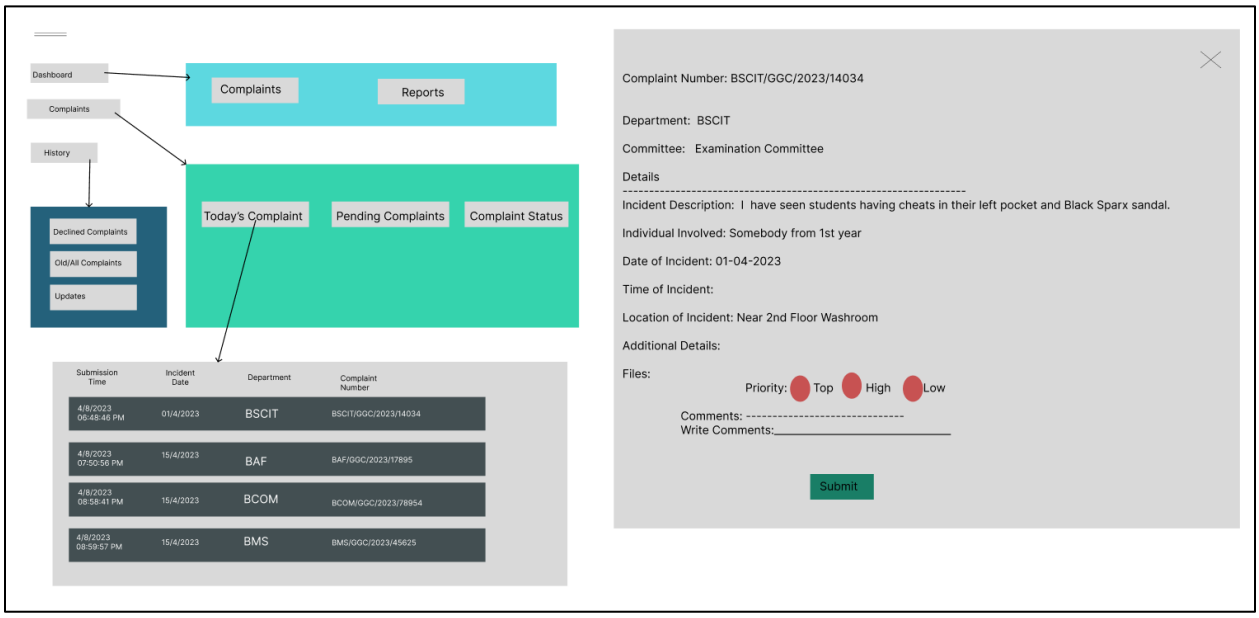


Fig. 4.13: Committee Member Dashboard (Sketch)

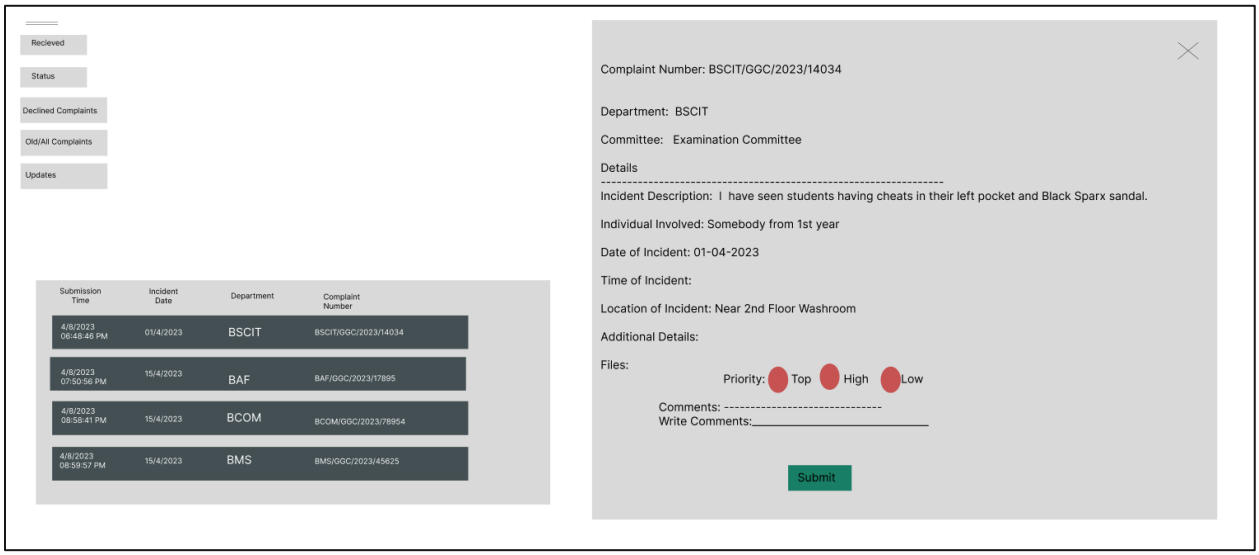


Fig. 4.14: Teachers Dashboard (Sketch)

4.5 Security Issues:

Data Encryption (AES) Implementation: Using Advanced Encryption Standard (AES) encryption to protect sensitive data both at rest (in the database) and during transmission (over the network).

Input Validation (Regex) Implementation: Employ regular expressions (regex) to validate and sanitize user inputs. Reject any inputs that contain unwanted characters or malicious code, preventing SQL injection and other attacks.

Authentication and Authorization Implementation: Implement strong authentication mechanisms (e.g., multi-factor authentication) to ensure only authorized users can access the system. Assign appropriate roles and permissions to users based on their responsibilities.

Access Controls: Implementation: Enforce strict access controls and least privilege principles. Users should only have access to the data and functionalities necessary for their roles. This is achieved through role-based access control (RBAC).

Password Policies: Enforcing strong password policies, requiring users to create complex passwords and regularly update them. Hash and salt stored passwords to prevent data leaks in case of breaches.

Session Management: Implement secure session management to protect user sessions from session hijacking or fixation attacks. Use secure tokens and expiring sessions.

Secure File Uploads: Implementation: When allowing file uploads, restrict file types to known, safe formats, and scan uploaded files for malware or malicious code.

Security Training: - Implementation: Train system users and administrators in security best practices to reduce the likelihood of accidental security breaches.

4.6 Test Cases Design

Table 4.3: Test Cases Design

Sr No	Test Case	Description	Expected Result
1	Input Sanitization	Forms would allow only Regular expressions for input. If any character is being entered it would give alert. Valid Input: A-Z, a-z, 0-9 and special characters, - _ ( ) / @ : & ? \ % Invalid Input:< > + ~ ` and other	If valid character is entered, then take input if not don't take.
2	Captcha Generator	Captcha test is designed to determine if an online user is really a human and not a bot.	It should be of 8 characters.
3	Complaint Number Generator	It would be generated only when a complaint is valid like data should not have any unwanted characters other than allowed.	According to the complaint filed it should generate Complaint No. like taking data of Department, Year, and random 5 digits at the end
4	User Creation	User creation is done by Admin it should note Email Id should be in "abc@thesiacollege.com" and automatically generating password.	Only valid Email Id should be entered and when successful creation it should display password on the same.
5	Dynamic Login	On successful Sign-in it would take to Dashboard and if not then Error message to enter login Credentials.	If successful login display Dashboard as per Role assigned.
6	Session Management	No user should get access of files another than role has been assigned.	If trying to get access it should display index page.

Chapter 5: Implementation & Testing

5.1 Implementation Approaches

This project is being implemented by making various modules and then integrating them with each other. As these approaches help in making error free code and most effective way of making project. Steps used for implementation:

1. Defining and creating functionality for each module.
2. After making functional modules, integrate them with each other as these is the most effective way to do.
3. Making integrating them with each other can sometimes cause errors so to make it correct/ rectify it is the best approach to follow.

5.2 Coding Details and Code Efficiency

Functionalities in this project:

1) Complaint Submission Module:

```
//Login Form

    <div class="loginFormContainer">
    <div class="loginForm " style="max-width: fit-content; border: 1px solid black; margin: 0 auto">
    <?php if(isset($error)) echo "<p>$error</p>"; ?>
    <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
    <table>
    <tbody>
    <tr>
    <th><label for="email">Email:</label></th>
    <td> <input type="email" id="email" name="email" required>
    </td>
    </tr>
    <tr>
    <th><label for="password">Password:</label></th>
    <td> <input type="password" id="password" name="password" required>
    </td>
    </tr>
    <tr>
```

```
<td colspan="2"> <div class="submit-btn-container">
<input type="submit" value="Login">
</div>
</td>
</tr>
</tbody>
</table>
</form>
</div>
</div>

<?php
session_start();
require_once "db_connect.php";

// Function to securely hash passwords
function hashPassword($password) {
return password_hash($password, PASSWORD_BCRYPT);
}

// Function to verify password
function verifyPassword($password, $hashedPassword) {
return password_verify($password, $hashedPassword);
}

// Check if form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
$email = htmlspecialchars($_POST["email"]);
$password = htmlspecialchars($_POST["password"]);
$stmt = $conn->prepare("SELECT id, name, email, password, role, committee
FROM users3 WHERE email = ?");
$stmt->execute([$email]); // You can directly pass an array to execute()
$result = $stmt->fetch(PDO::FETCH_ASSOC);
// Check if user exists
if ($result) {
if (verifyPassword($password, $result["password"])) {
$_SESSION["user_id"] = $result["id"];
$_SESSION["name"] = $result["name"]; // Store the name in the session
$_SESSION["email"] = $result["email"];
$_SESSION["role"] = $result["role"];
$_SESSION["committee"] = $result["committee"];
// Redirect to appropriate folder based on role
switch ($result["role"]) {
case 'admin':
```

```
header("Location: /admin/index.php");
break;
case 'cc':
header("Location: /cc/index.php");
break;
case 'cm':
header("Location: /cm/index.php");
break;
default:
// Handle unknown role
break;
}
exit;
} else {
$error = "Invalid email or password";
}
} else {
$error = "Invalid email or password";
header("Location: https://www.google.com/");
exit;
}
}
include "header.php";
?>
```

```
// Complaint Form
<form id="complaintForm" action="complaint_handler.php" method="post"
enctype="multipart/form-data" autocomplete="off">
<table>
  <tbody>
  <tr>
  <th>Department: <span style="color:#FF0000"><b>*</b></span></th>
  <td>
  <select id="dept" onchange="populateYears()" name="dept" required >
  <option value="">Select a Department</option>
  <option value="baf">Bachelor Of Accounting And Finance</option>
  <option value="bbi">Bachelor Of Commerce - Banking And Insurance</option>
  <option value="bcom">Bachelor Of Commerce</option>
  <option value="bscit">Bachelor Of Science - Information Technology</option>
  <option value="bmm">Bachelor of Arts in Multimedia and Mass
  Communication</option>
  <option value="bms">Bachelor Of Management Studies</option>
  </select>
<br>
```

```
<select id="year" disabled="" name="year" required>
<option value="">Select a year</option>
</select>
</td>
</tr>
<tr>
<th>Committee: <span style="color:#FF0000"><b>*</b></span></th>
<td> <select id="cmite" name="cmite" required>
<option value="">Select Committee</option>
<option value="exc">Examination Committee</option>
<option value="icc">Internal Complaint Committee</option>
<option value="ggc">Grievance Redressal Committee</option>
</select>
</td>
</tr>
<tr>
<th style="text-align: center" colspan="2">Details:</th>
</tr>
<tr>
<th>Complaint/Griverance Description:
<span style="color:#FF0000"><b>*</b></span></th>
<td><textarea id="incident_description" name="incdnt_dscrptin"
required></textarea></td>
</tr>
<tr>
<th>Individual involved: <span style="color:#FF0000"><b>*</b></span></th>
<td><input type="text" id="involved" name="indiv_inv" required></td>
</tr>
<tr>
<th>Date of Incident: <span style="color:#FF0000"><b>*</b></span></th>
<td><input type="date" id="incdnt_date" name="date" required></td>
</tr>
<tr>
<th>Time of Incident:</th>
<td><input type="time" id="incdnt_time" name="time"></td>
</tr>
<tr>
<th>Location of Incident: <span style="color:#FF0000"><b>*</b></span></th>
<td>
<select id="locationSelector" name="location" required
onchange="showOptions()">
<option value="">Select Location</option>
<option value="inside">Inside College Campus</option>
<option value="outside">Outside College Campus</option>
</select>
<div id="cmap" style="display: none;">
```

```
<div id="map"></div>
        <div id="coordinates" class="coordinates-container">
        Latitude: <span id="latitude"></span>, Longitude: <span
        id="longitude"></span>
        <input type="hidden" id="latitudeInput" name="latitude">
        <input type="hidden" id="longitudeInput" name="longitude">
        </div>
</div>
<div id="floorOptions" style="display: none;">
        <select id="floorSelector" name="floor" onchange="showRooms()">
        <option value="">Select Floor</option>
        <option value="ground">Ground Floor</option>
        <option value="first">First Floor</option>
        <option value="second">Second Floor</option>
        <option value="third">Third Floor</option>
        <option value="fourth">Fourth Floor</option>
        </select>
</div>
<div id="roomOptions" style="display: none;">
        <select id="roomSelector" name="room">
        <option value="">Select Room No.</option>
        </select>
</div>
</td>
</tr>
<tr>
<th>Additional Details:</th>
<td><textarea id="addit_dtls" name="add_dtls"></textarea></td>
</tr>
<tr>
<th>Upload Files (Images,Videos,Audio,PDF): <br><i>( File size shouldn't be
greater then 10MB. )</i></th>
<td>
<input type="file" name="file_upload" id="file_upload" accept="image/*,
video/*, audio/*, .pdf, .doc, .docx, .txt">
<ul id="fileList"></ul>
</td>
</tr>
<tr>
<th>CAPTCHA:<span style="color:#FF0000"><b>*</b></span>
<br><i>(it is HIGHLY sensitive; if DOUBT refresh CAPTCHA)</i>
</th>
<td>
<img src="captcha.php" alt="CAPTCHA Image" id="captchaImg"><br>
<a href="javascript:void(0)" onclick="refreshCaptcha()">Refresh</a><br>
<input type="text" id="captchaInput" name="captcha" required><br>
```

```
    </td>
    </tr>
    <tr>
    <td colspan="2">
    <div class="submit-btn-container">
    <input type="submit" value="Submit">
</div>
</td>
</tr>
</tbody>
</table>
</form>
</div>


<? php

function sanitizeInput($input) {
return htmlspecialchars(trim($input), ENT_QUOTES, 'UTF-8');
}
function handleFileUpload($fieldName, $complaintNumber){
if (isset($_FILES[$fieldName]) && $_FILES[$fieldName]['error'] ===
UPLOAD_ERR_OK) {
$fileTmpPath = $_FILES[$fieldName]['tmp_name'];
$fileName = $_FILES[$fieldName]['name'];
$fileType = $_FILES[$fieldName]['type'];

// Generate a unique file name by concatenating the complaint number
$fileNameWithComplaintNumber = "{$complaintNumber}_{$fileName}";

// Choose the destination folder based on the file type
$destinationFolder = '';
if (strpos($fileType, 'image/') === 0) {
$destinationFolder = 'uploads/images/';
} elseif (strpos($fileType, 'video/') === 0) {
$destinationFolder = 'uploads/videos/';
} elseif (strpos($fileType, 'audio/') === 0) {
$destinationFolder = 'uploads/audios/';
} else {
$destinationFolder = 'uploads/documents/';
}

// Create the destination folder if it doesn't exist
if (!file_exists($destinationFolder)) {
mkdir($destinationFolder, 0755, true);
}
```

```
$fileDestination = $destinationFolder . $fileNameWithComplaintNumber;

if (move_uploaded_file($fileTmpPath, $fileDestination)) {
return $fileDestination;
} else {
// Return an error message if file upload fails
return "Error uploading $fieldName.";
}
}
return null;
}


function verifyCaptcha($userInput) {
if (isset($_SESSION['captcha']) && !empty($userInput)) {
$storedCaptcha = $_SESSION['captcha'];
return $userInput === $storedCaptcha;
}
return false;
}

function generateComplaintNumber($dept, $committee) {
$currentYear = date('Y');
$randomDigits = str_pad(mt_rand(0, 99999), 5, '0', STR_PAD_LEFT);
$complaintNumber = strtoupper("$dept/$committee/$currentYear/$randomDigits");
$sanitizedComplaintNumber = preg_replace('/[^a-zA-Z0-9_]/', '_',
$complaintNumber);
return $sanitizedComplaintNumber;
}


// Handling form submission
if ($_SERVER["REQUEST_METHOD"] === "POST") {
// Validate and sanitize user input here (implement your validation logic)
$dept = isset($_POST["dept"]) ? sanitizeInput($_POST["dept"]) : "";
$year = isset($_POST["year"]) ? sanitizeInput($_POST["year"]) : "";
$committee = isset($_POST["cmite"]) ? sanitizeInput($_POST["cmite"]) : "";
$incdnt_dscrptin = isset($_POST["incdnt_dscrptin"]) ?
sanitizeInput($_POST["incdnt_dscrptin"]) : "";
$indiv_inv = isset($_POST["indiv_inv"]) ? sanitizeInput($_POST["indiv_inv"]) :
"";
$date = isset($_POST["date"]) ? sanitizeInput($_POST["date"]) : "";
$time = isset($_POST["time"]) ? sanitizeInput($_POST["time"]) : "";
$location = isset($_POST["location"]) ? sanitizeInput($_POST["location"]) : "";
$add_dtls = isset($_POST["add_dtls"]) ? sanitizeInput($_POST["add_dtls"]) : "";
```

```
// Concatenate room number and coordinates based on user selection
if ($location === "inside") {
$room_no = isset($_POST["room"]) ? sanitizeInput($_POST["room"]) : ""; //
Assuming room number is selected
$location_value = $room_no; // Store room number as location value for inside
location
} elseif ($location === "outside") {
$latitude = isset($_POST["latitude"]) ? sanitizeInput($_POST["latitude"]) : "";
$longitude = isset($_POST["longitude"]) ? sanitizeInput($_POST["longitude"]) : "";
$location_value = $latitude . ', ' . $longitude; // Combine latitude and longitude as
location value for outside location
}

// If location value is still empty, display an error message
if (empty($location_value)) {
$response = array('success' => false, 'message' => 'Location value is required.');
echo json_encode($response);
exit; // Stop further execution
}


// CAPTCHA verification
$captchaValue = isset($_POST['captcha']) ? $_POST['captcha'] : '';
if (!verifyCaptcha($captchaValue)) {
// CAPTCHA verification failed
$response = array('success' => false, 'message' => 'Incorrect CAPTCHA! Please try
again.');
echo json_encode($response);
unset($_SESSION['captcha']);
exit;
}

// Generate complaint number
$complaint_number = generateComplaintNumber($dept, $committee);


// Handle file uploads and get the file paths
$file_upload = handleFileUpload('file_upload', $complaint_number);
$location_value = "";

// Prepare and execute the SQL statement using prepared statements
$stmt = $conn->prepare("INSERT INTO complaint2 (dept, year, committee,
incdnt_dscrptin, indiv_inv, date, time, location, add_dtls, file_upload,
complaint_number) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
if ($stmt) {
```

```
        $stmt->bind_param( "sssssssssss", $dept, $year, $committee, $incdnt_dscrptin,
        $indiv_inv, $date, $time, $location_value, $add_dtls, $file_upload,
        $complaint_number );
        if ($stmt->execute()) {
        $stmt->close();
        $conn->close();
        header('Content-Type: application/json');
        $response = array('success' => true, 'message' => 'Complaint submitted
        successfully!', 'complaintNumber' => $complaint_number);
        echo json_encode($response);
        exit; // Stop further execution
        } else {
        $response = array('success' => false, 'message' => 'Error executing query: ' . $stmt-
>error);
        echo json_encode($response);
        exit; // Stop further execution
        }
        } else {
        $response = array('success' => false, 'message' => 'Error preparing statement: ' .
        $conn->error);
        echo json_encode($response);
        exit;
        }


        <script>
        function initializeMap() {
        var map = L.map('map').setView([19.2092927, 73.0994759], 13);
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a
        href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        }).addTo(map);
        var selectedMarker = null;
        map.on('click', function(e) {
        if (selectedMarker) {
        map.removeLayer(selectedMarker);
        }
        var lat = e.latlng.lat.toFixed(6);
        var lng = e.latlng.lng.toFixed(6);
        selectedMarker = L.marker([lat, lng]).addTo(map);
        map.panTo([lat, lng]);
        // Update the hidden input fields with latitude and longitude values
        document.getElementById('latitude').textContent = lat;
        document.getElementById('longitude').textContent = lng;
        document.getElementById('latitudeInput').value = lat;
        document.getElementById('longitudeInput').value = lng;
```

```
});
}

function showOptions() {
var locationSelector = document.getElementById("locationSelector");
var floorOptions = document.getElementById("floorOptions");
var mapContainer = document.getElementById("cmap");
var roomOptions = document.getElementById("roomOptions");
roomOptions.style.display = "none";
if (locationSelector.value === "inside") {
floorOptions.style.display = "block";
mapContainer.style.display = "none";
} else if (locationSelector.value === "outside") {
floorOptions.style.display = "none";
mapContainer.style.display = "block";
initializeMap();
} else {
floorOptions.style.display = "none";
mapContainer.style.display = "none";
} }

function showRooms() {
var floorSelector = document.getElementById("floorSelector");
var roomOptions = document.getElementById("roomOptions");
if (floorSelector.value !== "") {
roomOptions.style.display = "block";
var roomSelector = document.getElementById("roomSelector");
roomSelector.innerHTML = "";
if (floorSelector.value === "ground") {
roomSelector.innerHTML += '<option value="G01">Room No. G01</option>';
roomSelector.innerHTML += '<option value="G02">Room No. G02</option>';
roomSelector.innerHTML += '<option value="G03">Room No. G03</option>';
roomSelector.innerHTML += '<option value="G04">Room No. G04</option>';
roomSelector.innerHTML += '<option value="G05">Room No. G05</option>';
roomSelector.innerHTML += '<option value="G06">Room No. G06</option>';
} else if (floorSelector.value === "first") {
roomSelector.innerHTML += '<option value="BMMSR">BMM Staff
Room</option>';
roomSelector.innerHTML += '<option value="ITS">IT Staff Room</option>';
roomSelector.innerHTML += '<option value="MgR">Management
Room</option>';
roomSelector.innerHTML += '<option value="CoSR">Common
StaffRoom</option>';
roomSelector.innerHTML += '<option value="101">Room 101</option>';
roomSelector.innerHTML += '<option value="102">Room 102</option>';
roomSelector.innerHTML += '<option value="103">Room 103</option>';
```

```
roomSelector.innerHTML += '<option value="104">Room 104</option>';
roomSelector.innerHTML += '<option value="105">Room 105</option>';
roomSelector.innerHTML += '<option value="106">Room 106</option>';
roomSelector.innerHTML += '<option value="107">Room 107</option>';
} else if (floorSelector.value === "second") {
roomSelector.innerHTML += '<option value="NCC">NCC Room</option>';
roomSelector.innerHTML += '<option value="IOT">IOT Lab</option>';
roomSelector.innerHTML += '<option value="SERVER">Server Room</option>';
roomSelector.innerHTML += '<option value="LIBRARY">Library</option>';
roomSelector.innerHTML += '<option value="MScLab">M.Sc. I.T. Lab</option>';
roomSelector.innerHTML += '<option value="MdLab">Media Lab</option>';
roomSelector.innerHTML += '<option value="ComLab">Commerce
Lab</option>';
roomSelector.innerHTML += '<option value="BScLab">B.Sc. I.T. Lab</option>';
} else if (floorSelector.value === "third") {
roomSelector.innerHTML += '<option value="IQAC">IQAC Room</option>';
roomSelector.innerHTML += '<option value="301">Room 301</option>';
roomSelector.innerHTML += '<option value="302">Room 302</option>';
roomSelector.innerHTML += '<option value="303">Room 303</option>';
roomSelector.innerHTML += '<option value="304">Room 304</option>';
roomSelector.innerHTML += '<option value="305">Room 305</option>';
roomSelector.innerHTML += '<option value="306">Room 306</option>';
roomSelector.innerHTML += '<option value="307">Room 307</option>';
roomSelector.innerHTML += '<option value="308">Room 308</option>';
roomSelector.innerHTML += '<option value="309">Room 309</option>';
roomSelector.innerHTML += '<option value="310">Room 310</option>';
roomSelector.innerHTML += '<option value="311">Room 311</option>';
roomSelector.innerHTML += '<option value="312">Room 312</option>';
roomSelector.innerHTML += '<option value="313">Room 313</option>';
} else if (floorSelector.value === "fourth") {
roomSelector.innerHTML += '<option value="Auditorium">Auditorium</option>';
roomSelector.innerHTML += '<option value="401">Room 401</option>';
roomSelector.innerHTML += '<option value="402">Room 402</option>';
roomSelector.innerHTML += '<option value="403">Room 403</option>';
roomSelector.innerHTML += '<option value="404">Room 404</option>';
roomSelector.innerHTML += '<option value="405">Room 405</option>';
} } else {
roomOptions.style.display = "none";
}
}
});
</script>
```

// Captcha Generation

```
<?php
session_start();
```

```
// Generate a random CAPTCHA string
$randomString = generateRandomString(8);
// Store the CAPTCHA string in the session
$_SESSION["captcha"] = $randomString;
$width = 150;
$height = 50;
$im = imagecreatetruecolor($width, $height);
$bgColor = imagecolorallocate($im, 255, 255, 255);
$black = imagecolorallocate($im, 0, 0, 0);
$grey = imagecolorallocate($im, 128, 128, 128);
imagefilledrectangle($im, 0, 0, $width, $height, $bgColor);
$font = __DIR__ . '/RobotoMonoItalicVariableFont.ttf';
$fontSize = 15;
$textBox = imagettfbbox($fontSize, 0, $font, $randomString);
$textWidth = $textBox[4] - $textBox[0];
$textHeight = $textBox[1] - $textBox[5];
$x = ($width - $textWidth) / 2-5;
$y = ($height - $textHeight) / 2 + $fontSize;
for ($i = 0; $i < strlen($randomString); $i++) {
   $char = $randomString[$i];
   $color = ($i % 2 === 0) ? $black : $grey;
   imagettftext($im, $fontSize, 0, $x + $i * $fontSize, $y, $color, $font, $char);
}
header('Content-type: image/png');
imagepng($im);
imagedestroy($im);
function generateRandomString($length = 8) {
   $characters =
'qwertyuiopasdfghjklzxcvbnm74185296930QWERTYUIOPASDFGHJKLZXCVB
NM';
   $charactersLength = strlen($characters);
   $randomString = '';
   for ($i = 0; $i < $length; $i++) {
      $randomString .= $characters[rand(0, $charactersLength - 1)];
   }
   return $randomString;
}
?>
```

// **Status of Complaint**

```
<div class="flest" style="max-width: fit-content; border: 1px solid black; margin: 0
auto">
<form  method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
<?php if (!isset($complaint_number) && !isset($error_message)): ?>
<table border="1">  <tbody>
<tr> <th scope="row">Enter Complaint Number:</th>
<td><input type="text" name="txtI"></td>
</tr>
<tr> <th scope="row" colspan="2"><input type="submit" value="Submit"></th>
</tr>
</tbody>
</table>
<?php endif; ?>
<?php if (isset($complaint_number)): ?>
<table border="1">
<tbody> <tr>
<th scope="row">Entered Complaint Number:</th>
<td><?php echo $complaint_number; ?></td>
</tr>
<tr> <th scope="row">Status:</th>
<td><?php echo $status; ?></td>
</tr>
<tr> <th scope="row">Remarks:</th>
<td><?php echo $remarks1; ?></td>
</tr>
</tbody>
</table>
<?php endif; ?>
<?php if (isset($error_message)): ?>
<table border="1"> <tbody>
<tr>
<th scope="row">Enter Complaint Number:</th>
<td><input type="text" name="txtI"></td>
</tr>
<tr> <div class="submit-btn-container">
<th scope="row" colspan="2"><input type="submit" value="Submit"></th>
</div>
</tr>
<tr style="color: red;"> <th scope="row" colspan="2"><?php echo $error_message;
?></th>
</tr>
</tbody>
</table>
```

```
        <?php endif; ?>
        </form>
        </div>

        // Check if the form is submitted
        <? php
        if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // Get the complaint number from the form
        $txtI = $_POST["txtI"];
        $sql = "SELECT * FROM assignedcomplaints WHERE complaint_number = ?";
        $stmt = $conn->prepare($sql);   $stmt->bind_param("s", $txtI);
        $stmt->execute();  $result = $stmt->get_result();
        if ($result->num_rows > 0) {   $row = $result->fetch_assoc();
        $complaint_number = $row["complaint_number"];
        $status = $row["status"]; $remarks1 = $row["remarks1"];
        } else { $error_message = "Complaint not found"; }
$stmt->close();  $conn->close();  ?>
```

2) Admin Module:

```
// Display Chart

        <div>
        <canvas id="complaintsChart" width="400" height="200"></canvas>
        </div>
        <script>
        document.addEventListener('DOMContentLoaded', function() {
        let complaintsChart;
        // Function to fetch complaints data
        function fetchComplaints(committee) {
        if (committee === 'all') {
        return fetch('fetcher.php')
        .then(response => response.json())
        .catch(error => console.error('Error fetching data:', error));
        } else {
        return fetch('fetch_complaints.php?committee=' + committee)
        .then(response => response.json())
        .catch(error => console.error('Error fetching data:', error));
        }
        }

        // Function to create and update the chart
        function updateChart(data, stacked, subtitleText) {
```

```
const chartCanvas = document.getElementById('complaintsChart');
const ctx = chartCanvas.getContext('2d');
// Clear existing chart
if (complaintsChart) {
complaintsChart.destroy();
}
// Create new chart
complaintsChart = new Chart(ctx, {
type: 'bar',
data: data,
options: {
plugins: {
subtitle: {
display: true,
text: subtitleText
}
},
scales: {
xAxes: [{
stacked: stacked
}],
yAxes: [{
stacked: stacked,
ticks: {
beginAtZero: true
} }] } }
});
}

// Fetch complaints data from server
fetchComplaints(committee)
.then(data => {
let subtitleText = '';
if (committee === 'all') {
subtitleText = 'All Committee Status';
// Generate chart data for all committees
const chartData = {
labels: data.map(committeeData => committeeData.committee),
datasets: [
{
label: 'Posted',
backgroundColor: 'rgba(255, 99, 132, 0.5)',
data: data.map(committeeData => committeeData.posted)
},
{
label: 'Acknowledged',
```

```
backgroundColor: 'rgba(54, 162, 235, 0.5)',
data: data.map(committeeData => committeeData.acknowledged)
},
{
label: 'Closed',
backgroundColor: 'rgba(255, 206, 86, 0.5)',
data: data.map(committeeData => committeeData.closed)
}
]
};
updateChart(chartData, true, subtitleText); // Stack bars for all committees
} else {
let fullCommitteeName = '';
switch (committee) {
case 'exc':
fullCommitteeName = 'Examination Committee';
break;
case 'icc':
fullCommitteeName = 'Internal Complaint Committee';
break;
case 'ggc':
fullCommitteeName = 'Grievance Redressal Committee';
break;
default:
fullCommitteeName = 'Unknown Committee';
break;
}
subtitleText = 'Selected Committee: ' + fullCommitteeName;

// Generate chart data for the selected committee
const statusCounts = {};
data.forEach(item => {
if (statusCounts[item.status]) {
statusCounts[item.status]++;
} else {
statusCounts[item.status] = 1;
}
});
const chartData = {
labels: Object.keys(statusCounts),
datasets: [{
label: 'Number of Complaints',
data: Object.values(statusCounts),
backgroundColor: [
'rgba(255, 99, 132, 0.5)',
'rgba(54, 162, 235, 0.5)',
```

```
'rgba(255, 206, 86, 0.5)'
],
borderColor: [
'rgba(255, 99, 132, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)'
],
borderWidth: 1
}]
};
updateChart(chartData, false, subtitleText); // Simple bars for individual
committee
}
})
.catch(error => console.error('Error fetching data:', error));
}

// Event listener for committee selection
const committeeSelect = document.getElementById('committeeSelect');
committeeSelect.addEventListener('change', function() {
const selectedCommittee = this.value;
displayComplaints(selectedCommittee);
});

// Display complaints for the default selection (All Committees)
displayComplaints('all');
});

</script>
<div>
<canvas id="complaintsChart" width="400" height="200"></canvas>
```

//function

```
<?php
function RandomPassword($length = 10) {
$characters =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRS
TUVWXYZ';
$password = '';
for ($i = 0; $i < $length; $i++) {
$password .= $characters[rand(0, strlen($characters) - 1)];
}
return $password;
}
function get_total_all_records()
{
include('db.php');
$statement = $connection->prepare("SELECT * FROM users3");
$statement->execute();
$result = $statement->fetchAll();
return $statement->rowCount();
}
if (isset($_POST["operation"])) {
if ($_POST["operation"] == "Add") {
$role = $_POST["role"];
$committee = ($role === "admin") ? null : $_POST["committee"];
// Generate random password
$password = RandomPassword();
// Hash password
$hashedPassword = password_hash($password,
PASSWORD_DEFAULT);
$statement = $connection->prepare("
INSERT INTO users3 (name, email, role, committee, password)
VALUES (:name, :email, :role, :committee, :password)");
$result = $statement->execute(
array(
':name' => $_POST["name"],
':email' => $_POST["email"],
':role' => $role,
':committee' => $committee,
':password' => $hashedPassword
)
);
echo json_encode(array('email' => $_POST["email"], 'password' =>
$password));
}
```

```
if ($_POST["operation"] == "Edit") {
$role = $_POST["role"];
$committee = ($role === "admin") ? null : $_POST["committee"];

// If role is 'admin', reset committee to null
if ($role === "admin") {
$committee = null;
}

$statement = $connection->prepare("
UPDATE users3
SET name = :name, email = :email, role = :role, committee = :committee
WHERE id = :id");
$result = $statement->execute(
array(
':name' => $_POST["name"],
':email' => $_POST["email"],
':role' => $_POST["role"],
':committee' => $committee, // Use the modified $committee value
':id' => $_POST["member_id"]
) ); }  }
?>
<?php
include('db.php');
include('function.php');
$query = '';
$output = array();
$query .= "SELECT id, name, email, role, committee FROM users3 ";
if(isset($_POST["search"]["value"])) {
$query .= 'WHERE name LIKE "%'.$_POST["search"]["value"].'%" ';
$query .= 'OR email LIKE "%'.$_POST["search"]["value"].'%" ';
}
if(isset($_POST["order"])) {
$query .= 'ORDER BY '.$_POST['order']['0']['column'].'
'.$_POST['order']['0']['dir'].' ';
} else {
$query .= 'ORDER BY id ASC ';
}
if($_POST["length"] != -1) {
$query .= 'LIMIT ' . $_POST['start'] . ', ' . $_POST['length'];
}
$statement = $connection->prepare($query);
$statement->execute();
$result = $statement->fetchAll();
$data = array();
$filtered_rows = $statement->rowCount();
```

```
// Mapping abbreviations to full names
$role_mapping = array(
'admin' => 'Admin',
'cm' => 'Committee Member',
'cc' => 'Committee Convener',
'icc' => 'Internal Complaint Committee',
'ggc' => 'Grivernace Redressal Committee',
'exc' => 'Examination Committee'
);
$committee_mapping = array(
'exc' => 'Examination Committee',
'icc' => 'Internal Complaint Committee',
'ggc' => 'Grivernace Redressal Committee',
);
foreach($result as $row) {
$sub_array = array();
$sub_array[] = $row["id"];
$sub_array[] = $row["name"];
$sub_array[] = $row["email"];
// Convert role abbreviation to full name using mapping array
$role = isset($role_mapping[$row["role"]]) ?
$role_mapping[$row["role"]] : $row["role"];
// Convert committee abbreviation to full name using mapping array
$committee = isset($committee_mapping[$row["committee"]]) ?
$committee_mapping[$row["committee"]] : $row["committee"];
$sub_array[] = $role;
$sub_array[] = $committee;
$sub_array[] = '<button type="button" name="update" id="'.$row["id"].'"
class="btn btn-primary btn-sm update"><i class="glyphicon glyphicon-
pencil"></i>Edit</button>'; // Removed extra closing button tag
$sub_array[] = '<button type="button" name="delete" id="'.$row["id"].'"
class="btn btn-danger btn-sm delete">Delete</button>';
$data[] = $sub_array;
}
$output = array(
"draw"            =>   intval($_POST["draw"]),
"recordsTotal"    =>   $filtered_rows,
"recordsFiltered"  =>   get_total_all_records(),
"data"            =>   $data );
echo json_encode($output);
?>
```

3) Committee Convener and Committee Member Module:

//User Dasboard

```
<div id="dashboard" class="tab-content  active">
<h2>Assigned Complaints to:    <?php
if(isset($_SESSION["committee"])) {
// Check if the user is an admin and not associated with any committee
if($_SESSION["role"] === 'admin' && $_SESSION["committee"] === null) {
echo 'You are an admin and are not associated with any committee. Be responsible.';
} else {
// Otherwise, display the committee description based on the committee code
switch($_SESSION["committee"]) {
case 'icc': echo 'Internal Complaint Committee';
break;
case 'ggc': echo 'Grievance Redressal Committee';
break;
case 'exc': echo 'Examination Committee';
break;
default:
// For other committees, display the committee code as it is
echo $_SESSION["committee"];
break;
} } }
?>
</h2>
<table>
<tr>
<td> <label for="startDate">From Date:</label>
</td>
<td> <input type="date" id="startDate" name="startDate">
</td>
<td> <label for="endDate">To Date:</label>
</td>
<td> <input type="date" id="endDate" name="endDate">
</td>
<td colspan="2"> <button type="button" class="btn btn-primary" style="display:
block; margin: 0 auto;" onclick="displayGraph()">Display Graph</button>
</td>
</tr>
</table>
<div>
<canvas id="complaintsChart" width="fit-content" height="fit-content"></canvas>
</div>
```

```
<script>
document.addEventListener('DOMContentLoaded', function() {
let complaintsChart;
// Function to fetch & display complaints based on selected committee & date range
function displayComplaints(committee, startDate, endDate) {
// Fetch complaints data from server with selected committee & date range
fetch(`fetcher.php?committee=${committee}&startDate=${startDate}&endDate=${endDate}`)
.then(response => response.json())
.then(data => {
const statusCounts = {};
data.forEach(item => {
if (statusCounts[item.status]) {
statusCounts[item.status]++;
} else {
statusCounts[item.status] = 1;
}
});

// Update or create chart
updateChart(statusCounts);
})
.catch(error => console.error('Error fetching data:', error));
}

// Function to update or create the chart
function updateChart(data) {
const chartCanvas = document.getElementById('complaintsChart');
const ctx = chartCanvas.getContext('2d');

// Clear existing chart
if (complaintsChart) {
complaintsChart.destroy();
}

// Create new chart
complaintsChart = new Chart(ctx, {
type: 'bar',
data: {
labels: Object.keys(data),
datasets: [{
label: 'Number of Complaints',
data: Object.values(data),
backgroundColor: [
'rgba(255, 99, 132, 0.5)',
```

```
'rgba(54, 162, 235, 0.5)',
'rgba(255, 206, 86, 0.5)'
],
borderColor: [
'rgba(255, 99, 132, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)'
],
borderWidth: 1
}]
},
options: {
scales: {
yAxes: [{
ticks: {
beginAtZero: true
}
}]
}
}
});
}

// Function to handle report generation
function displayGraph() {
const committee = "<?php echo $_SESSION['committee']; ?>"; // Get the selected committee from PHP session
const startDate = document.getElementById('startDate').value;
const endDate = document.getElementById('endDate').value;
displayComplaints(committee, startDate, endDate);
}

// Initially display complaints for the selected committee and default date range (today's date)
const today = new Date();
const defaultStartDate = today.toISOString().split('T')[0]; // Get today's date in yyyy-mm-dd format
const defaultEndDate = today.toISOString().split('T')[0]; // Get today's date in yyyy-mm-dd format
document.getElementById('startDate').value = defaultStartDate; // Set start date input field to today's date
document.getElementById('endDate').value = defaultEndDate;
displayGraph(); // Display complaints for today initially
document.getElementById('startDate').addEventListener('change', displayGraph);
document.getElementById('endDate').addEventListener('change', displayGraph);
});
```

```
</script>
</div>
</div>
```

//Functions

```
<?php
function get_total_all_records()
{
include('db.php');
$statement = $connection->prepare("SELECT * FROM complaint2");
$statement->execute();
$result = $statement->fetchAll();
return $statement->rowCount();
}

function fetch_complaints_based_on_role_and_committee($role, $committee) {
global $connection;

$stmt = $connection->prepare("SELECT * FROM complaint2 WHERE role =
:role AND committee = :committee");
$stmt->bindParam(':role', $role);
$stmt->bindParam(':committee', $committee);
$stmt->execute();

if ($stmt->errorCode() != '00000') {
$errorInfo = $stmt->errorInfo();
echo "Error executing query: " . $errorInfo[2];
return array();
}

return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
?>

<?php
session_start();
include('db.php');
// Function to fetch total records based on committee
function get_total_all_records($committee) {
global $connection;
$statement = $connection->prepare("SELECT * FROM complaint2 WHERE
committee = ?");
```

```
$statement->execute([$committee]);
$result = $statement->fetchAll();
return $statement->rowCount();
}

// Check if specific user from committee conveners is logged in
if(isset($_SESSION["committee"])) {
$total_records = get_total_all_records($_SESSION["committee"]);
} else {
// Handle the scenario where the committee is not set
$total_records = 0; // Default value for total records
}
$query = '';
$output = array();
$query .= "SELECT c.*, a.status FROM complaint2 c LEFT JOIN
assignedcomplaints a ON c.complaint_number = a.complaint_number WHERE
c.committee = ?";
// search condition only for complaint number if searched
if(isset($_POST["search"]["value"])) {
$query .= " AND c.complaint_number LIKE '%" . $_POST["search"]["value"] .
"%' ";
}
// Order complaints by status: Posted > Acknowledged > Closed
$query .= ' ORDER BY FIELD(a.status, "Posted", "Acknowledged", "Closed"),
c.timestamp ASC';
// pagination
if($_POST["length"] != -1) {
$query .= ' LIMIT ' . $_POST['start'] . ', ' . $_POST['length'];
}
$statement = $connection->prepare($query);
$statement->execute([$_SESSION["committee"]]); // Pass committee from
session
$result = $statement->fetchAll();
$data = array();
$filtered_rows = $statement->rowCount();

foreach($result as $row) {
$sub_array = array();

$sub_array[] = $row["id"];
$sub_array[] = $row["year"];
$sub_array[] = $row["complaint_number"];
$sub_array[] = $row["incdnt_dscrptin"];
$sub_array[] = $row["indiv_inv"];
$sub_array[] = '<button type="button" name="update" id="'.$row["id"].'"
class="btn btn-primary btn-sm update"><i class="glyphicon glyphicon-pencil">
```

```
</i>Add Remarks</button></button>';

// Fetch status from the database based on complaint number
$status = fetch_status_from_database($row["complaint_number"]);



// Display status as a colored div
$status_html = '<div class="status-div" style="background-color:
'.$status['color'].'; text-align: center;">'.$status['text'].'</div>';
$sub_array[] = $status_html;
$data[] = $sub_array;
}

$output = array(
"draw"            =>   intval($_POST["draw"]),
"recordsTotal"    =>   $filtered_rows,
"recordsFiltered"   =>   $total_records, // Use the value obtained from
get_total_all_records() function
"data"            =>   $data
);
echo json_encode($output);

?>

<?php
include('db.php');
function fetch_status_from_database($complaint_number) {
global $connection;
$statement = $connection->prepare("SELECT status FROM assignedcomplaints
WHERE complaint_number = :complaint_number");
$statement->bindParam(':complaint_number', $complaint_number);
$statement->execute();
if ($statement->errorCode() != '00000') {
$errorInfo = $statement->errorInfo();
echo "Error executing query: " . $errorInfo[2];
return array('color' => 'gray', 'text' => 'Unknown');
}
$status = $statement->fetch(PDO::FETCH_ASSOC);

// If status is found
if ($status) {
// Determine color and text based on status
switch ($status['status']) {
case 'Posted':
$result = array('color' => '#9ad0f5', 'text' => 'Posted');
```

```
break;
case 'Acknowledged':
$result = array('color' => '#ffb1c1', 'text' => 'Acknowledged');
break;
case 'Closed':
$result = array('color' => '#ffe6aa', 'text' => 'Closed');
break;
default:
$result = array('color' => 'gray', 'text' => 'Unknown');
break;
}
} else {
$result = array('color' => 'red', 'text' => 'Unknown');
}

return $result;
}

?>

<?php
include('db.php');
include('function.php');
if (isset($_POST["additional_remarks"]) &&
isset($_POST["complaint_number"]) && isset($_POST["status"])) {
$serializedRemarks = json_encode($_POST["additional_remarks"]);

// Update the remarks and status in the database
$statement = $connection->prepare("
UPDATE assignedcomplaints
SET remarks1 = :remarks, status = :status
WHERE complaint_number = :complaint_number");
$result = $statement->execute(
array(
':remarks' => $serializedRemarks,
':status' => $_POST["status"],
':complaint_number' => $_POST["complaint_number"]
)
);

if (!$result) {
echo "Error: " . $statement->errorInfo()[2];
} else {
echo "Remarks and status updated successfully.";
}
} else {
```

```
echo "Additional remarks, complaint number, or status field is missing in the
form submission.";
} } }




<script>
$(document).ready(function()
{
$('#add_button').click(function()
{
$('#member_form')[0].reset();
$('.modal-title').text("Add New Details");
$('#action').val("Last");
$('#operation').val("Last");
});

var dataTable = $('#member_table').DataTable
({
"paging":true,
"processing":true,
"serverSide":true,
"order": [],
"info":true,
"ajax":{
url:"fetchAll.php",
type:"POST"
},
"columnDefs":[
{
"targets":'_all',
"orderable":false,
},
],
"createdRow": function (row, data, index) {
// Set the serial number for each row
$('td', row).eq(0).html(index + 1);
},
});

$(document).on('submit', '#member_form', function(event){
event.preventDefault();
var complaint_number = $('#complaint_number').val();
var remarks = $('#additional_remarks').val();
var status = ($('#action').val() === "Add Remarks and Save") ? "Posted" :
```

```
"Closed";

if( complaint_number != '') {
var formData = new FormData(this);
formData.append('complaint_number', complaint_number);
formData.append('remarks', remarks);
formData.append('status', status); // Add status to the form data

$.ajax({
url: "insertupdated.php",
method: 'POST',
data: formData,
contentType: false,
processData: false,
success: function(data) {
$('#member_form')[0].reset();
$('#userModal').modal('hide');
dataTable.ajax.reload();
alert("Form submitted successfully!");
}
});
} else {
alert("Complaint number is a required field");
}
});

$(document).on('click', '.update', function() {
var member_id = $(this).attr("id");
$.ajax({
url: "fetch_single.php",
method: "POST",
data: {member_id: member_id},
dataType: "json",
success: function(data) {
// Populate the form fields
$('#complaint_number').val(data.complaint_number); // Set the complaint number
$('#userModal').modal('show');
$('#complaint_number_label').text(data.complaint_number); // Set the text content
of the label
$('#id').text(data.id);
$('#dept').text(data.dept);
$('#year').text(data.year);
$('#committee').text(data.committee);
$('#incdnt_dscrptin').text(data.incdnt_dscrptin);
$('#indiv_inv').text(data.indiv_inv);
$('#dateTime').text(data.dateTime); // Set the text content of the span with id
```

```
"dateTime"
$('#year').text(data.year);
$('#location').text(data.location);
$('#add_dtls').text(data.add_dtls);
$('#timestamp').text(data.timestamp);
$('#remarks1').text(data.remarks);
$('#status').text(data.status);
// Clear the value of additional_remarks field
$('#additional_remarks').val('');
if (data.file_upload != null) {
var files = data.file_upload.split(',');
var filesHtml = '';
files.forEach(function(file) {
var filePath = file.replace('/cc/', '/');
var fileUrl = 'https://dev.cln35h.in/' + filePath;
var fileName = file.split('/').pop(); // Extract file name
var fileNameWithoutPrefix = fileName.substring(fileName.lastIndexOf('_') + 1);
// Remove prefix
filesHtml += '<a href="' + fileUrl + '" target="_blank">' + fileNameWithoutPrefix
+ '</a><br>';
});
$('#displayFiles').html(filesHtml);
} else {
$('#displayFiles').html('No files available');
}

$('.modal-title').text("Complaint Details");
$('#member_id').val(member_id);
$('#action').val("Close Status");
$('#operation').val("Edit");
}
});
});

});
```

4) Report Module:

```
// Admin
        <div class="container-xl table table-striped" id="reportResult"></div>
        <script>
        function generateReport() {
        var committee = document.getElementById("committeeSelect1").value;
        var status = document.getElementById("statusSelect").value;
        var startDate = document.getElementById("startDate").value;
        var endDate = document.getElementById("endDate").value;
        var xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function() {
        if (xhr.readyState === XMLHttpRequest.DONE) {
        if (xhr.status === 200) {
        document.getElementById("reportResult").innerHTML = xhr.responseText;
        } else {
        console.error('Error: ' + xhr.status);
        }
        }
        };
        xhr.open('GET', 'generateReport.php?committee=' + committee + '&status=' +
        status + '&startDate=' + startDate + '&endDate=' + endDate, true);
        xhr.send();
        }
        </script>

        <?php
        // Check connection
        if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
        }
        $startDate = isset($_GET['startDate']) ? $_GET['startDate'] : '';
        $endDate = isset($_GET['endDate']) ? $_GET['endDate'] : '';
        $committee = isset($_GET['committee']) ? $_GET['committee'] : '';
        $status = isset($_GET['status']) ? $_GET['status'] : '';
        // Mapping of committee abbreviations to full names
        $committeeNames = array(
        'exc' => 'Examination Committee',
        'icc' => 'Internal Complaint Committee',
        'ggc' => 'Grievance Redressal Committee'
        );

        $sql = "SELECT c.complaint_number, c.committee, c.timestamp, a.status
        FROM complaint2 c
        INNER JOIN assignedcomplaints a ON c.complaint_number =
        a.complaint_number
```

```
WHERE c.timestamp BETWEEN '$startDate 00:00:00' AND '$endDate
23:59:59'";
// If a specific committee is selected, add a condition to filter by that committee
if ($committee != 'all') {
$sql .= " AND c.committee LIKE '%$committee%'";
}
// If a specific status is selected, add a condition to filter by that status
if ($status != 'all') {
$sql .= " AND a.status = '$status'";
}
$result = mysqli_query($conn, $sql);
if (!$result) {
die("Error executing query: " . mysqli_error($conn));
}
if (mysqli_num_rows($result) > 0) {
echo "<table class='table table-stripped'>";
echo "<tr><th>Serial No.</th><th>Complaint Number</th><th>Committee
Name</th><th>Timestamp</th><th>Status</th></tr>";
$serialNumber = 1;
while ($row = mysqli_fetch_assoc($result)) {
echo "<tr>";
echo "<td>".$serialNumber."</td>";
echo "<td>".$row["complaint_number"]."</td>";
// Display full name of the committee
echo "<td>".$committeeNames[$row["committee"]]."</td>";
echo "<td>".$row["timestamp"]."</td>";
echo "<td>".$row["status"]."</td>";
echo "</tr>";
$serialNumber++;
}
echo "</table>";
} else {
echo "No results found.";
}

mysqli_close($conn);
?
```

// Committee Convener and Committee Member

```
<div  class="container-xl table table-striped" style="max-width: fit-content;
border: 1px solid black; margin: 0 auto">
<form id="reportForm">
<table class="table table-stripped">
</tr>
<tr>
<td>
<label for="statusSelect">Select Status:</label>
</td>
<td>
<select id="statusSelect" name="status">
<option value="all">All Statuses</option>
<option value="Posted">Posted</option>
<option value="Acknowledged">Acknowledged</option>
<option value="Closed">Closed</option>
</select>
</td>
</tr>
<tr>
<td>
<label for="startDate">From Date:</label>
</td>
<td>
<input type="date" id="startDate" name="startDate">
</td>

</tr>
<tr>
<td>
<label for="endDate">To Date:</label>
</td>
<td>
<input type="date" id="endDate" name="endDate">
</td>
</tr>
<tr>
<td colspan="2">
<button type="button" class="btn btn-primary" style="display: block; margin:
0 auto;" onclick="generateReport()">Generate Report</button>


</td>
</tr>
</table>
```

```
</form>
<div class="container-xl table table-striped" id="reportResult"></div>

<script>
function generateReport() {
var status = document.getElementById("statusSelect").value;
var startDate = document.getElementById("startDate").value;
var endDate = document.getElementById("endDate").value;

// AJAX request
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
if (xhr.readyState === XMLHttpRequest.DONE) {
if (xhr.status === 200) {
document.getElementById("reportResult").innerHTML = xhr.responseText;
} else {
console.error('Error: ' + xhr.status);
}
}
};
// Get committee from session
var committee = "<?php echo isset($_SESSION['committee']) ?
$_SESSION['committee'] : ''; ?>";
xhr.open('GET', 'generateReport.php?committee=' + committee + '&status=' +
status + '&startDate=' + startDate + '&endDate=' + endDate, true);
xhr.send();
}

function printReport() {
var printContent = document.getElementById("reportResult").innerHTML;
var committeeDescription = "<?php
if(isset($_SESSION["committee"])) {
if($_SESSION["role"] === 'admin' && $_SESSION["committee"] === null) {
echo 'You are an admin and are not associated with any committee. Be
responsible.';
} else {
switch($_SESSION["committee"]) {
case 'icc': echo 'Internal Complaint Committee';
break;
case 'ggc': echo 'Grievance Redressal Committee';
break;
case 'exc': echo 'Examination Committee';
break;
default:
echo $_SESSION["committee"];
break;
```

```
}
}
}
?>";

var printWindow = window.open('', '_blank');
printWindow.document.open();
printWindow.document.write('<html><head><title>Print
Report</title></head><body>');
printWindow.document.write('<h5>Reports is generated for ' +
committeeDescription + '</h5>');
printWindow.document.write(printContent);
printWindow.document.write('</body></html>');
printWindow.document.close();
printWindow.print();
}
</script>
</div>
<?php

if (!$conn) {
die("Connection failed: " . mysqli_connect_error());
}

$startDate = isset($_GET['startDate']) ? $_GET['startDate'] : '';
$endDate = isset($_GET['endDate']) ? $_GET['endDate'] : '';
$committee = isset($_GET['committee']) ? $_GET['committee'] : '';
$status = isset($_GET['status']) ? $_GET['status'] : '';

// Mapping of committee abbreviations to full names
$committeeNames = array(
'exc' => 'Examination Committee',
'icc' => 'Internal Complaint Committee',
'ggc' => 'Grievance Redressal Committee'
);

$sql = "SELECT c.complaint_number, c.committee, c.timestamp, a.status
FROM complaint2 c
INNER JOIN assignedcomplaints a ON c.complaint_number =
a.complaint_number
WHERE c.timestamp BETWEEN ? AND ?";

$params = array("$startDate 00:00:00", "$endDate 23:59:59");
$types = "ss";

if ($committee != 'all') {
```

```
$sql .= " AND c.committee = ?";
$params[] = $committee;
$types .= "s";
}
if ($status != 'all') {
$sql .= " AND a.status = ?";
$params[] = $status;
$types .= "s";
}
$stmt = mysqli_prepare($conn, $sql);
if (!$stmt) {
die("Error preparing statement: " . mysqli_error($conn));
}
mysqli_stmt_bind_param($stmt, $types, ...$params);
$result = mysqli_stmt_execute($stmt);
if (!$result) {
die("Error executing query: " . mysqli_error($conn));
}
$result = mysqli_stmt_get_result($stmt);
if (mysqli_num_rows($result) > 0) {
echo "<table class='table table-stripped'>";
echo "<tr><th>Serial No.</th><th>Complaint Number</th><th>Committee
Name</th><th>Timestamp</th><th>Status</th></tr>";
$serialNumber = 1;
while ($row = mysqli_fetch_assoc($result)) {
echo "<tr>";
echo "<td>".$serialNumber."</td>";
echo "<td>".$row["complaint_number"]."</td>";
// Display full name of the committee
echo "<td>".$committeeNames[$row["committee"]]."</td>";
echo "<td>".$row["timestamp"]."</td>";
echo "<td>".$row["status"]."</td>";
echo "</tr>";
$serialNumber++;
}
echo "</table>";
echo "<button type='button' class='btn btn-primary no-print' style='display:
block; margin: 0 auto;' onclick='printReport()'>Print Report</button>";
} else {
echo "No results found.";
}
mysqli_stmt_close($stmt);
mysqli_close($conn);
?>
```

5.2.1 Code Efficiency

Code can be made efficient using various ways they are:

1. SQL queries: Database operations are often a bottleneck in web applications. Minimizing the number of database queries can significantly improve performance. One way to achieve this is by using SQL joins to fetch related data in a single query rather than making multiple queries. Prepared statements protect against SQL injection attacks and can also improve performance by reducing the overhead of repeatedly parsing and compiling SQL queries.
2. Using AJAX: AJAX helps in loading content as per the need and not loading client as well as server-side with lots of data that may result into crashing the system/ failure/ delaying the response timing.
3. Using loops: Loops are often a source of inefficiency, especially if they involve many iterations. Consider ways to minimize loop iterations or optimize loop logic to reduce execution time.

5.3 Testing Approach

Here testing is done with a modified top-down approach as this helps in solving problems by one part of testing starts from top to bottom as we are integrating units downward, and the second part from bottom to top for selected components declared as critical units. As there are 4 different modules and integrating them with each other is in itself time consuming process. So, before integration we should also test individual modules that are working or Unit testing, then we should move towards integration of modules. As we integrate them and try to make it work with each other, may sometimes result in functions working with one module and not with another so rectifying those functions can make working of each functionality again.

To do testing various cases and methodologies are prepared; sometimes these are done manually and at times it can be done with various automated software's also. Using test cases gives an idea about defining test cases on module and which type of testing should be done on them. If there is an error, it should be rectified or resolved at the earliest if not then it can cause problems in working of a normal flow as well as not giving great experience to the user.

5.3.1 Unit Testing

Unit testing on Complaint Submission Module:

- Test the functionality of complaint submission form.
- Test the validation of complaint details.
- Test the Committee selection options.
- Test the uploading of supporting documents.
- Ensure that complaints can be submitted anonymously.

Unit testing on Admin Module:

- Test the functionality of login and session configuration.
- Test user management functionalities such as adding, updating, and deleting users.
- Test for generating reports as all committee or committee wise; appropriate selection of date.

Unit testing on Committee Convener and Committee Member Module:

- Test the functionality of login and session configuration.
- Test proper complaints are displayed to the user who is assigned to that committee.
- Test assigned user can according to their role do activity like update remarks and change status with it.
- Test for generating reports as all committee or committee wise; appropriate selection of date.

5.3.2 Integrated Testing

Integration of Modules:

- Combine the complaint submission module, admin dashboard module, and complaint management module into a special testing environment.
- Ensure that all modules interact correctly with each other as per the system's design.

End-to-End Testing:

- Perform end-to-end tests to simulate real user scenarios, starting from submitting a complaint anonymously, through admin management, to resolution.
- Test various user flows to ensure seamless navigation and functionality across different modules.

Interoperability Testing:

- Check the interoperability between different components of the system.
- Verify that data flows smoothly between modules without loss or corruption.

Error Handling and Recovery:

- Test error handling mechanisms when unexpected events occur during the integration of modules.
- Ensure that the system can recover gracefully from errors without compromising data integrity or functionality.

Boundary Testing:

- Test the application's limits by providing inputs at the boundaries of acceptable ranges.
- Check how the system behaves when it reaches or exceeds its limits, such as maximum number of concurrent users or maximum file size for document uploads.

Performance Testing:

- Evaluate the performance of the integrated system under various load conditions.
- Measure response times, throughput, and resource utilization to ensure that the application can handle expected levels of traffic without degradation in performance.

Security Testing:

- Conduct security testing to identify vulnerabilities and ensure that sensitive data is protected.
- Test for common security threats such as SQL injection, cross-site scripting (XSS), and authentication bypass.

Compatibility Testing:

- Verify that the application works correctly across different browsers, operating systems, and devices.
- Test for compatibility with various versions of PHP and other relevant software dependencies.

Regression Testing:

- Perform regression testing to ensure that new changes or updates to the system do not introduce any unintended side effects or regressions.
- Re-run previously executed test cases to validate the stability of the integrated application.

5.3.3 Beta Testing

Beta Version Deployment:

- Ensure that the beta version of the complaint system is a representative build of the final product. While it may not include all features or be entirely bug-free, it should be stable enough for testing.
- Provide clear instructions to beta testers on how to access the beta version, including any login credentials or access codes required.

Testing Scenarios:

- Develop a variety of testing scenarios that cover different aspects of the complaint system's functionality. These scenarios should mimic real-world usage scenarios and encourage beta testers to explore all features of the system.
- Include tasks such as submitting different types of complaints, assigning complaints to specific committees, updating complaint statuses, and communicating with complainants.

Performance Monitoring:

- Monitor the performance of the beta version of the complaint system under various conditions, including different levels of user activity and system load.
- Use performance monitoring tools to track key metrics such as response times, server resource utilization, and error rates.
- Identify any performance bottlenecks or scalability issues and optimize the system accordingly to ensure it can handle the expected workload.

5.4 Modifications and Improvements

After integrating there were various problems that came into notice so to overcome those problems more effort was taken like:

Validation and Data Integrity Updates: After test results various validation protocols are used to ensure data integrity throughout the system. This involved implementing stricter validation checks at various input points to prevent data entry.

Security: Here testing revealed users that logged in can get access to different files of the user so to protect sensitive data proper session management is done. Also fixed common threats such as SQL injection and cross-site scripting (XSS). Additionally, we enhanced authentication mechanisms and implemented encryption protocols to safeguard user credentials and confidential information.

User Interface Refinement: Feedback from respected guide applied to change user interface for better usability and accessibility. Redesigned layout and navigation of webpages, doing these it enriches the webpages as well as updated Manual page as these gives an idea of how to work on page.

5.5 Test Cases

Table 5.1: Test cases

Sr. No.	Test Case	Input	Output
1	Dynamic Login, Input Sanitization	Email Id: abc@thesiacollege.com Password:	Login Successful if "@thesiacollege.com" and proper password; then redirect to appropriate Dashboard.
2		Email Id: abc@gmail.com Password:	Login Unsuccessful if "@thesiacollege.com" isn't there and redirect to Google.com.
3	Captcha Generator	Enter Captcha:	It should be bounded in the image and its border.
4		Enter Captcha:	It should be valid if not reset form.
5		Enter Captcha:	It should generate new every time.
6	User Creation	Enter Email Id:	It should end with "@thesiacollege.com"
7		Select Role:	If Admin is selected, then no other Committee is to be assigned.
8		Select Role:	If any role other than Admin is Assigned, then Committee should be assigned.
9		Add User.	When submit is clicked then it should display password assigned to it.

10		Update User.	It should fetch details of user form database and display.
11		Update User.	It should be able to Update Password, Change User Roles.
12		Delete User.	It should delete user.
13	Complaint Management	View Graph	As user is signed display graph on Dashboard
14		View Complaints	User assigned with role of Committee Member and Committee Convener should be able to see complaints.
15		Remarks:	Remarks should be added with updating status on that complaint.
16	Report Generation	Generate Report:	Report should be displayed to Admin and for Committee Convener and Member it should be allowed to take print.
17			If date is not selected, then it should display "No record found" as reports should be displayed from certain period of time only.

18	Session Management	Enter Email Id: Password:	If Admin is signed in, then it should display contents of its folder only and not of other same for other users also.
19		Logout	If the user is logging out, then all its sessions should be destroyed. And if not, it should logout after 30 Minutes.
20	Complaint Form	Location of Incident:	Allowing to Fetch coordinates from map after pin pointing.
21		Upload Files:	It should select only single file that can be Image, Video or Document like .pdf,.docx.
22		Date of Incident:	It should not allow to enter future date.

Chapter 6: Results & Discussion

6.1 Test Reports

Dynamic Login: If User is not found then it will redirect to Google.com and if user entered wrong password, then it will give message as "*Invalid email or password*"; and if all details are correct then it will display Dashboard.

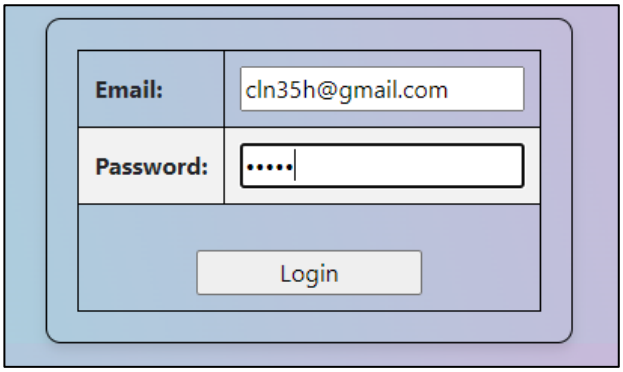


Fig. 6.1: Login Form

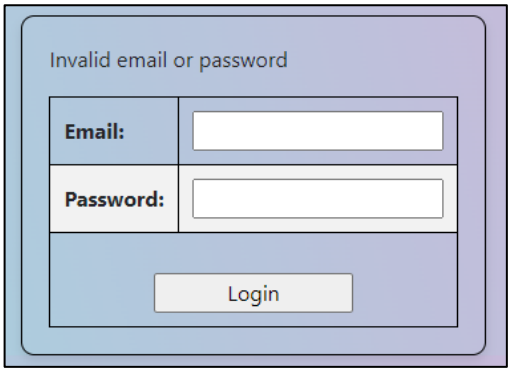


Fig. 6.2: Failed Login

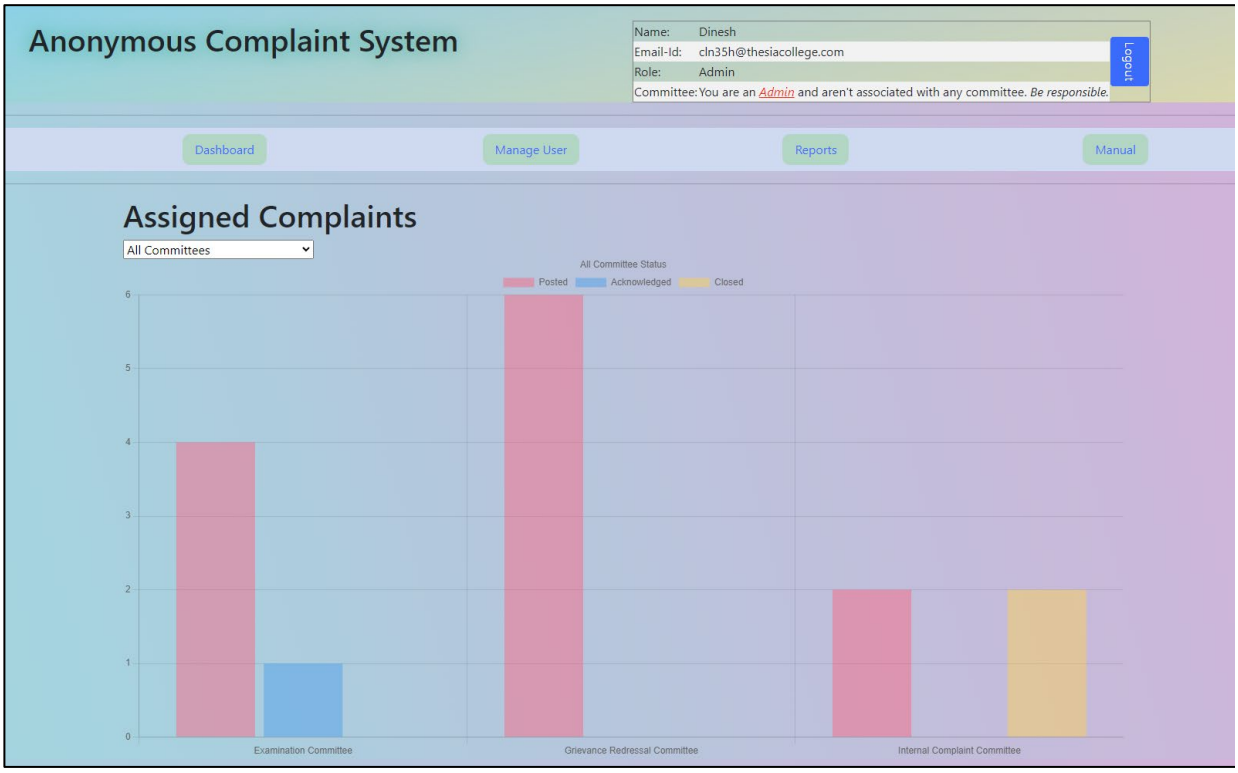


Fig. 6.3: Successful Login

Captcha Generator: Captcha is generated with the help of another file and then fetched in form.

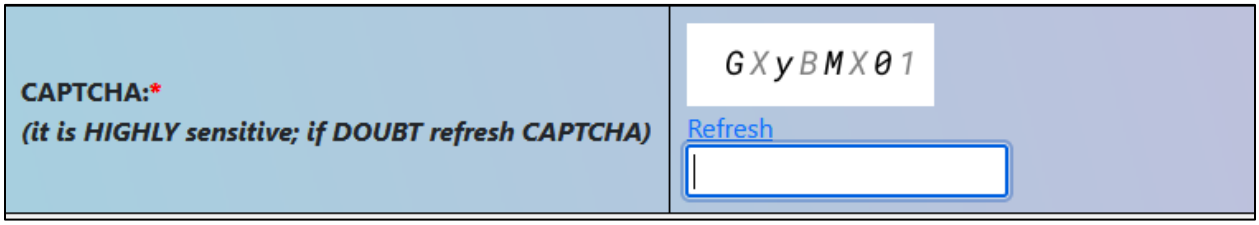


Fig. 6.4: CAPTCHA on Complaint form

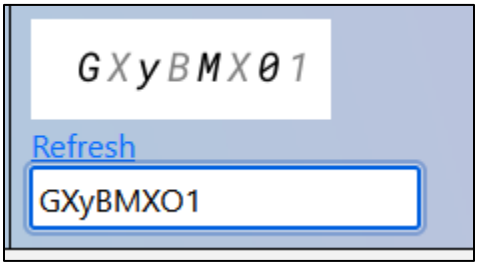


Fig. 6.5: Wrong CAPTCHA entered.

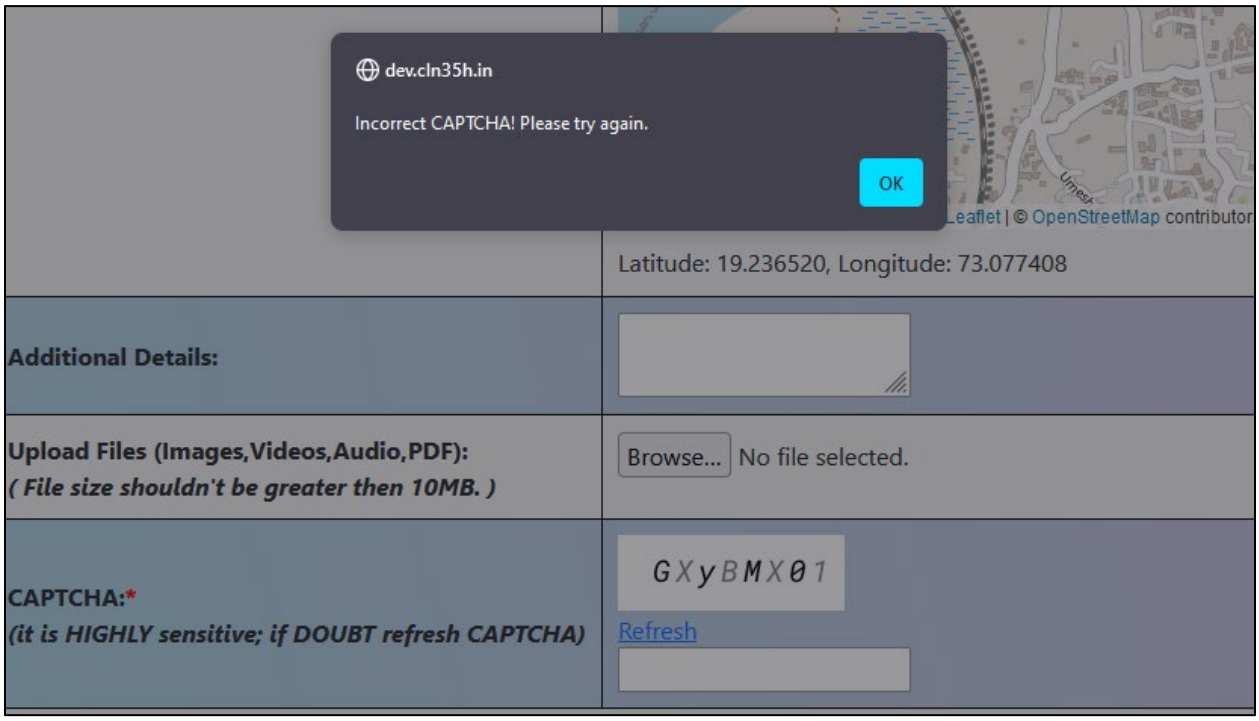


Fig. 6.6: Complaint Form reset due to Incorrect CAPTCHA.

Complaint Form: It is the most important part of this project, so various security measures are taken into consideration. For e.g. Some characters aren't allowed to be entered on input field as they can result in XSS attacks so to achieve that Regular Expression (ReGeX) is used on very input field. Another example is when user enters wrong CAPTCHA after filling a form it resets as this is consider as it is being filled by automation techniques. After successfully submitting a complaint, it would wait for 30 seconds and redirect to Google.com as well in mean time it would clear history from browser that Complaint form was opened.

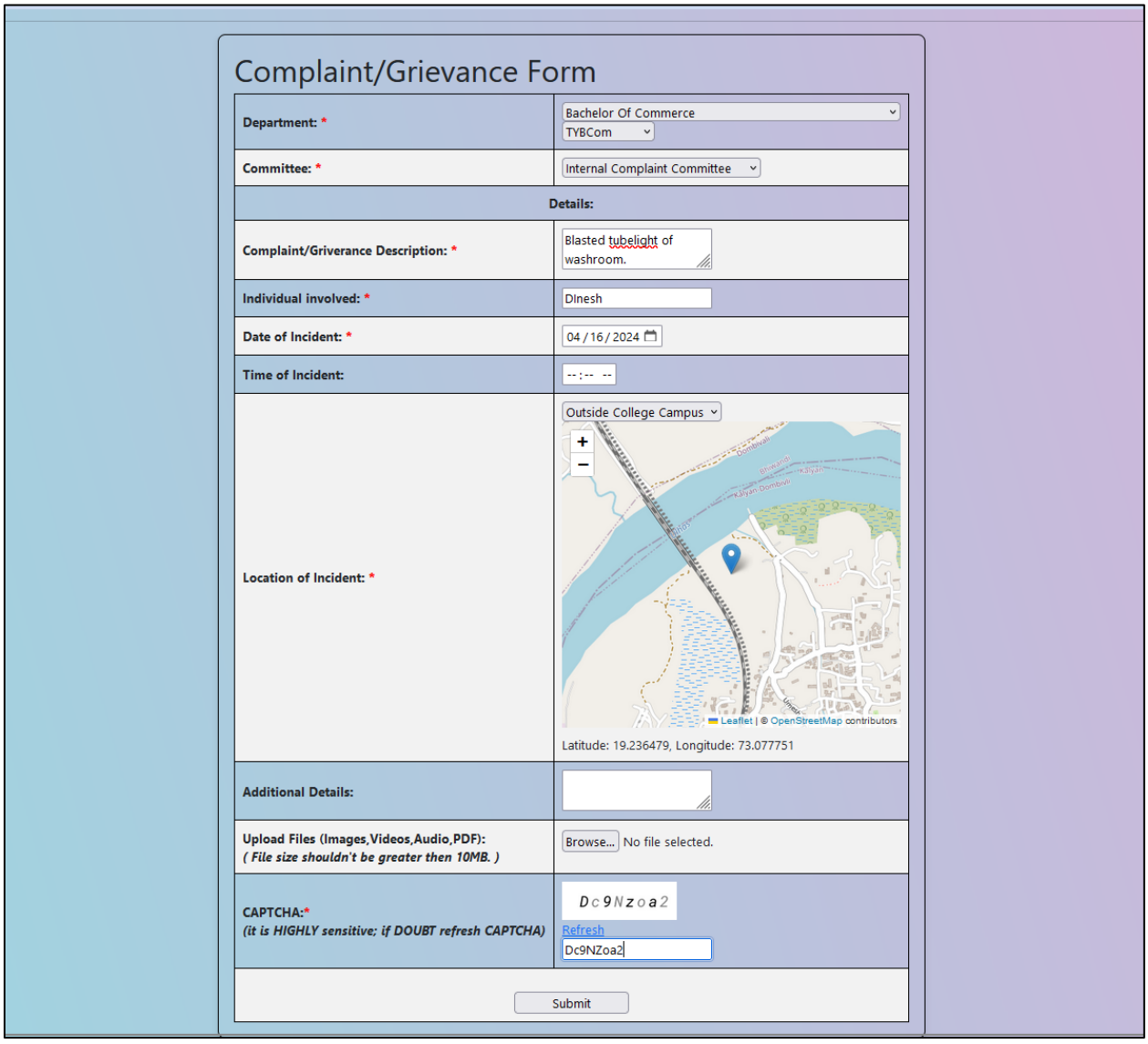


Fig. 6.7: Complaint form filled correctly but with wrong CAPTCHA.

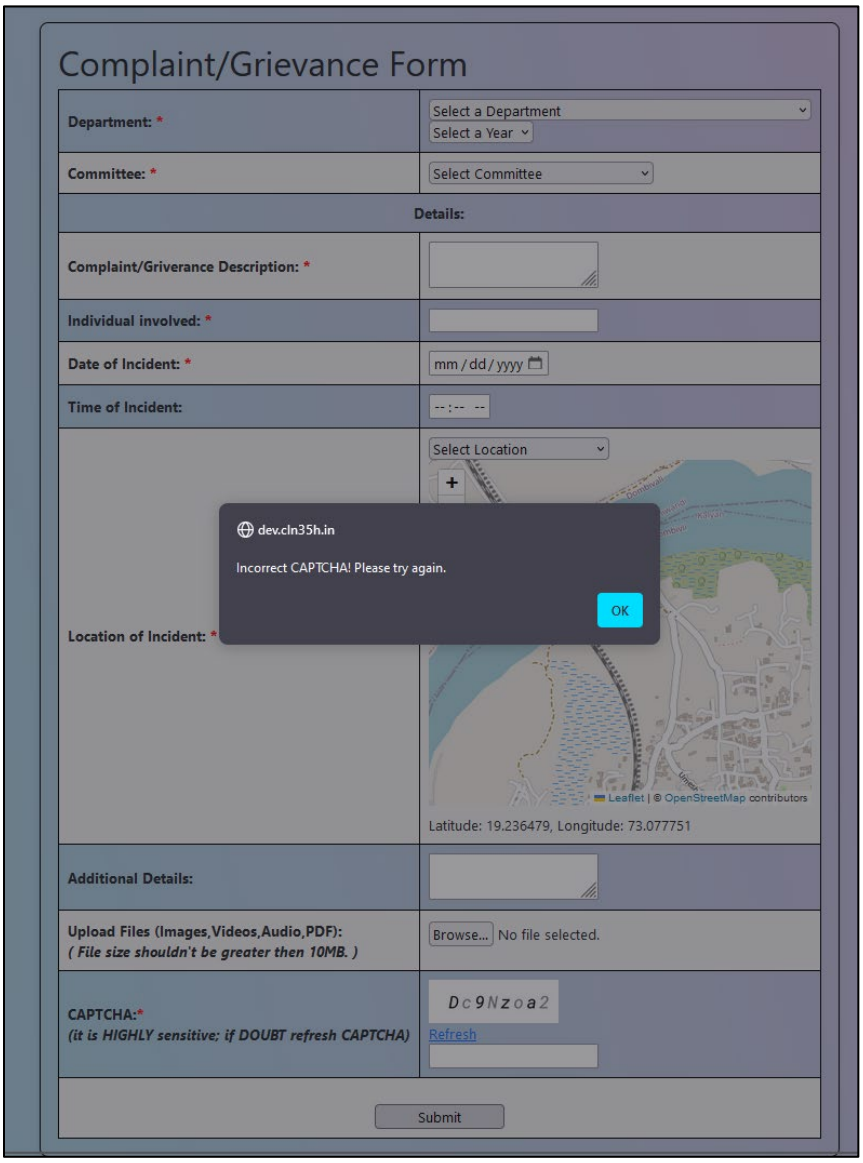


Fig. 6.8: Complaint form reset due to incorrect CAPTCHA.

Fig. 6.9: Complaint form filled correctly but with right CAPTCHA.

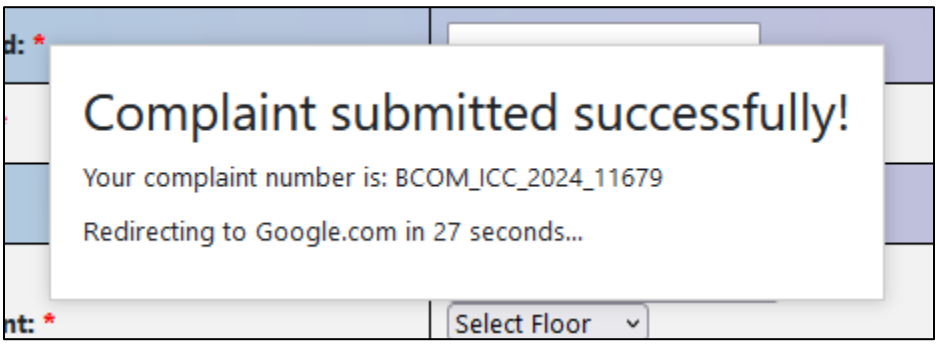


Fig. 6.10: Complaint Number generated.

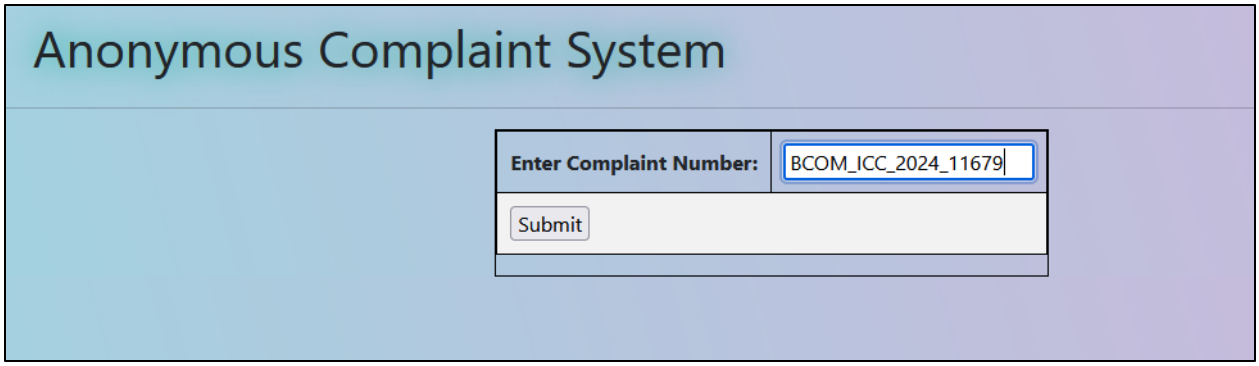


Fig. 6.11: Complaint Status form.

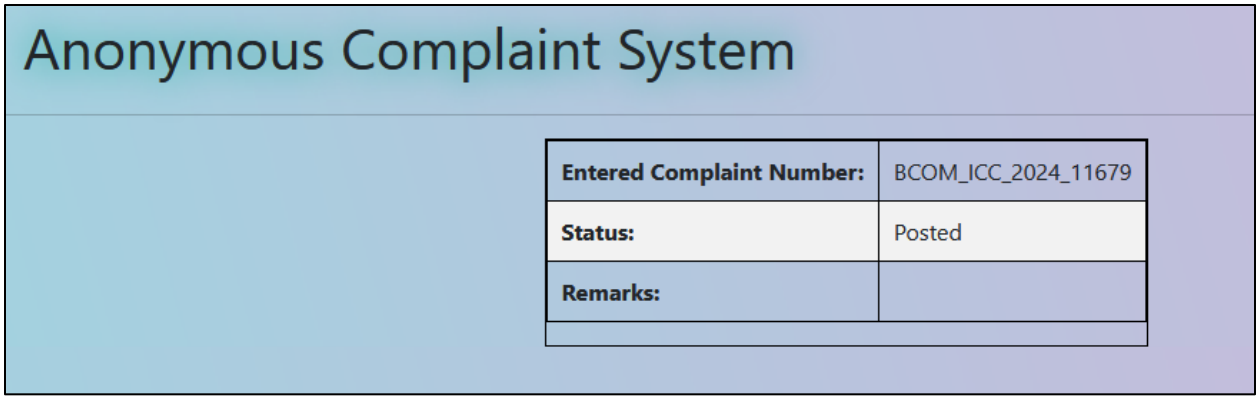


Fig. 6.12: Response of Complaint Number.

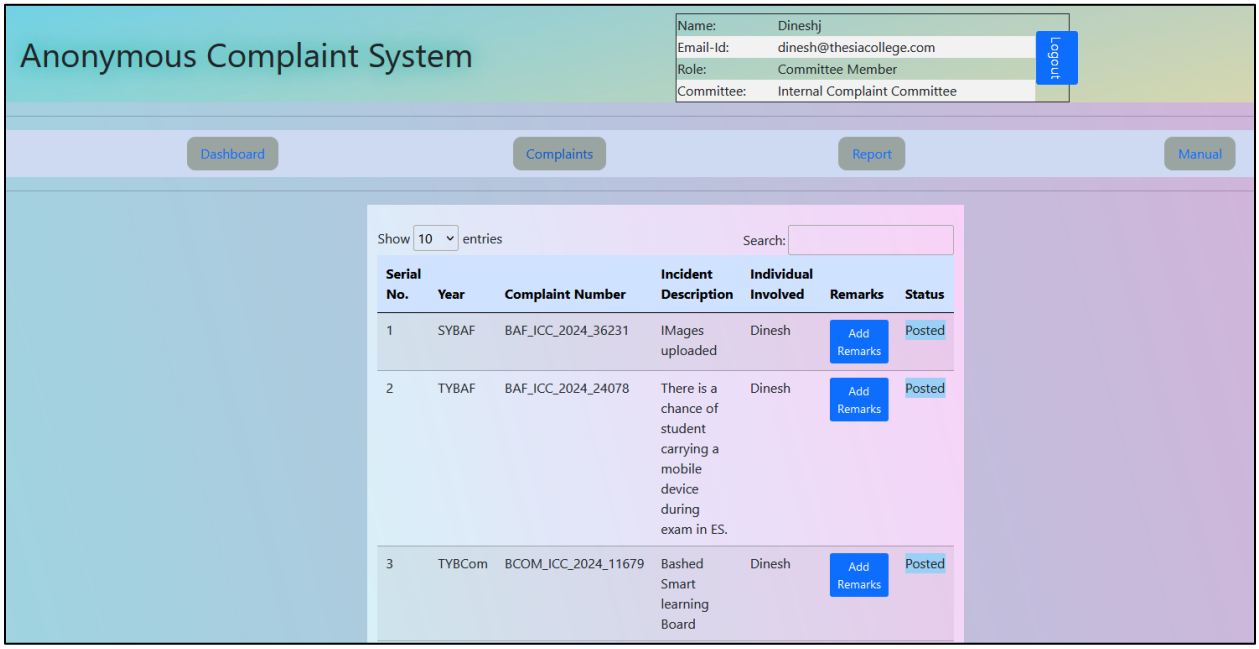


Fig. 6.13: Complaint in process of resolving by Committee Member.

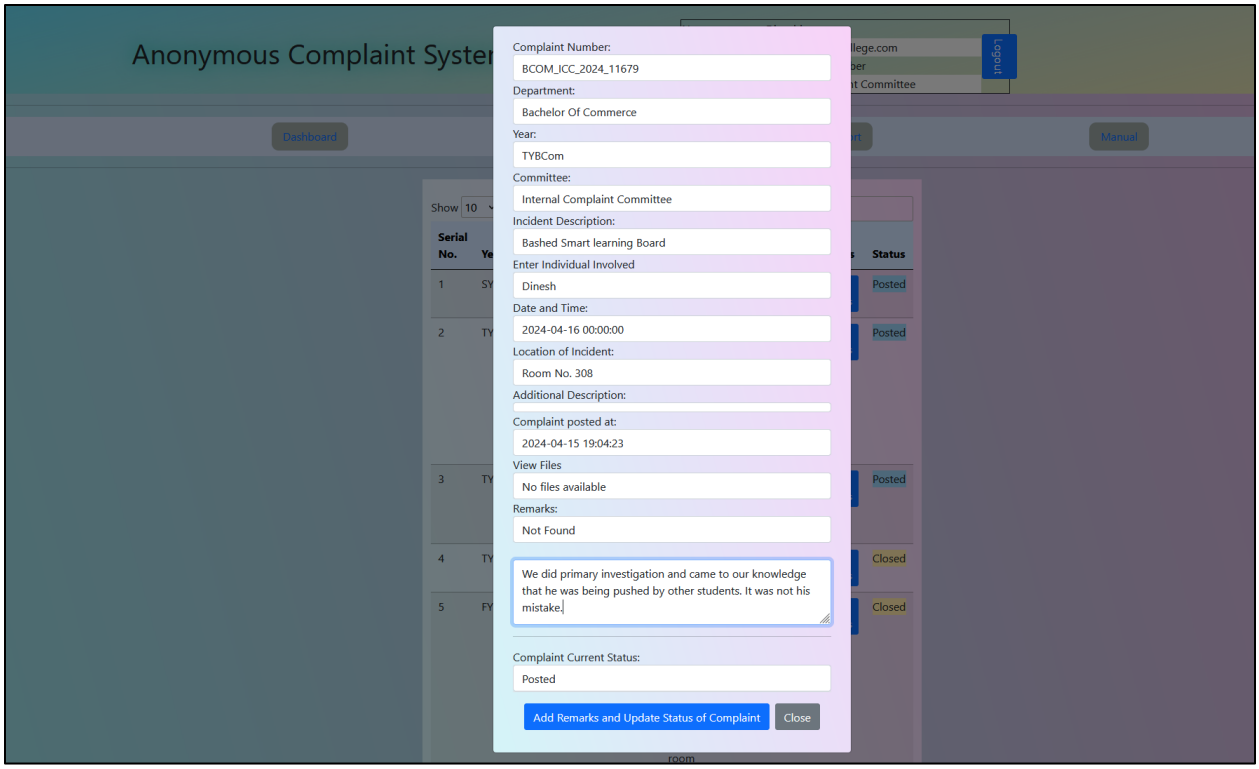


Fig. 6.14: Complete Complaint details on Committee Member dashboard.

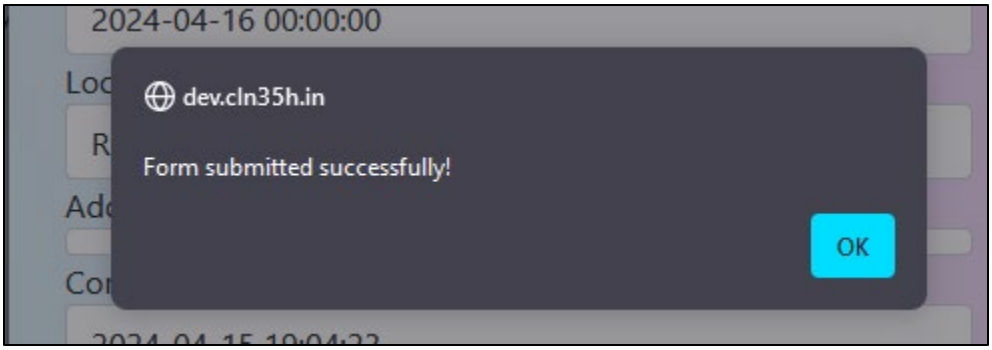


Fig. 6.15: Alert of successfully remarks added.

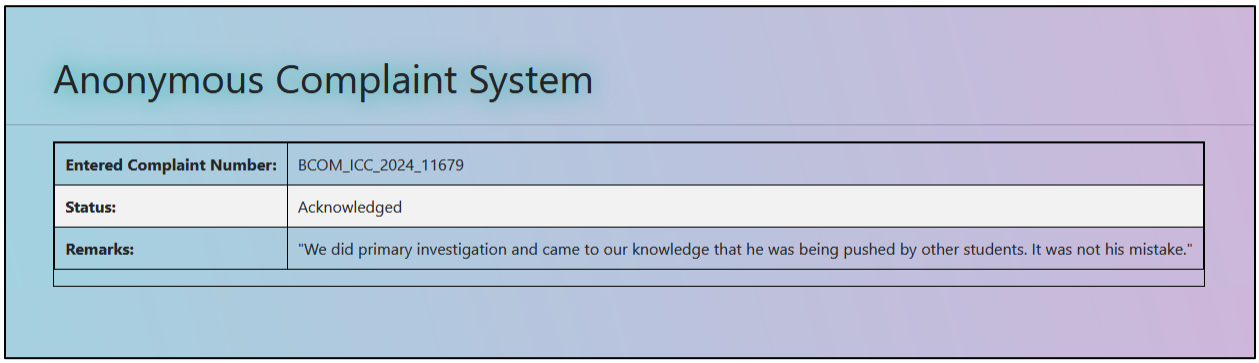


Fig. 6.16: Complaint Status after update of remarks.

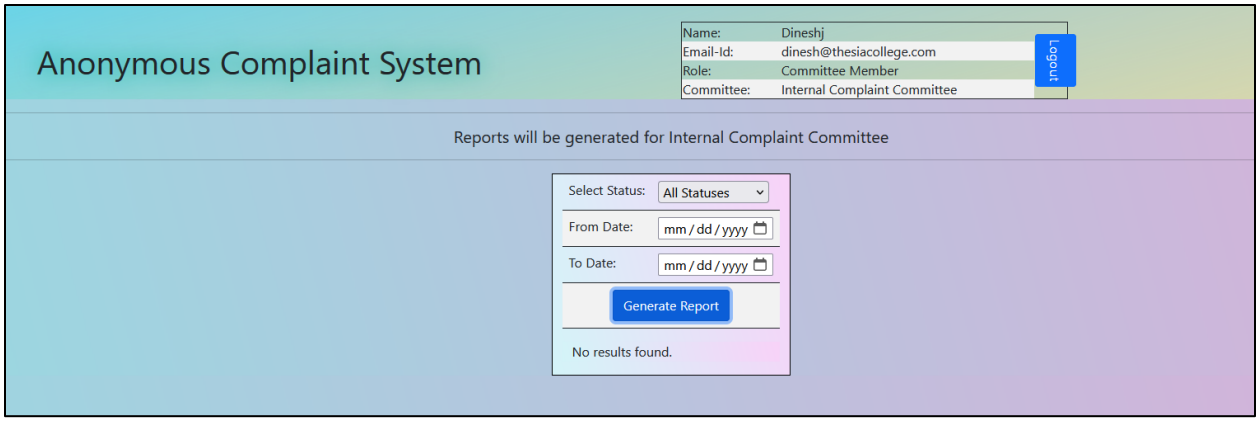


Fig. 6.17: Error on Report Generation as no proper date selected.

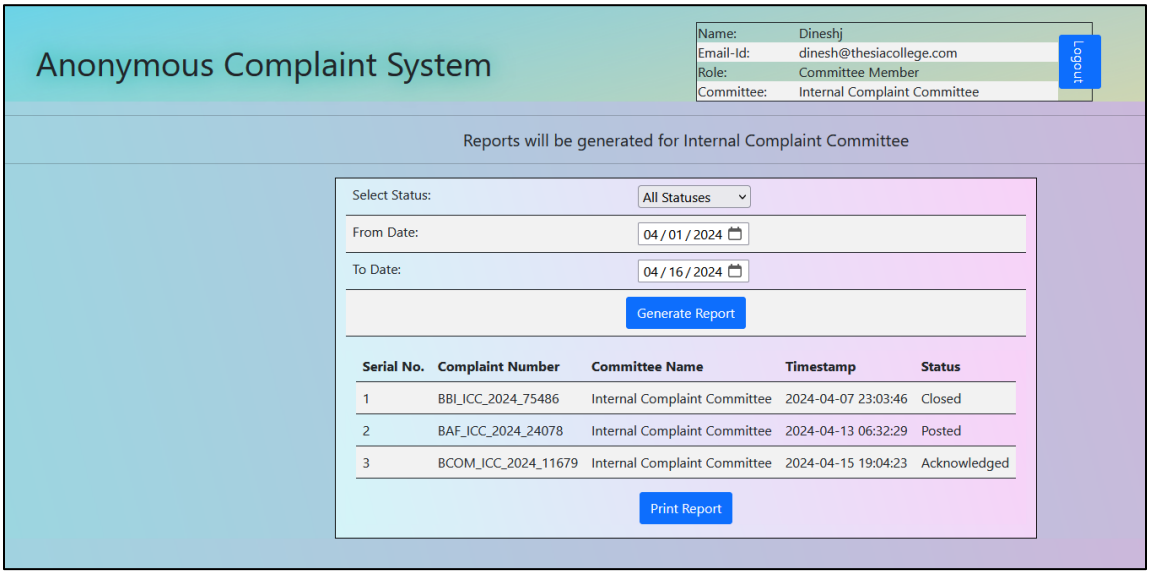


Fig. 6.18: Report displayed properly.

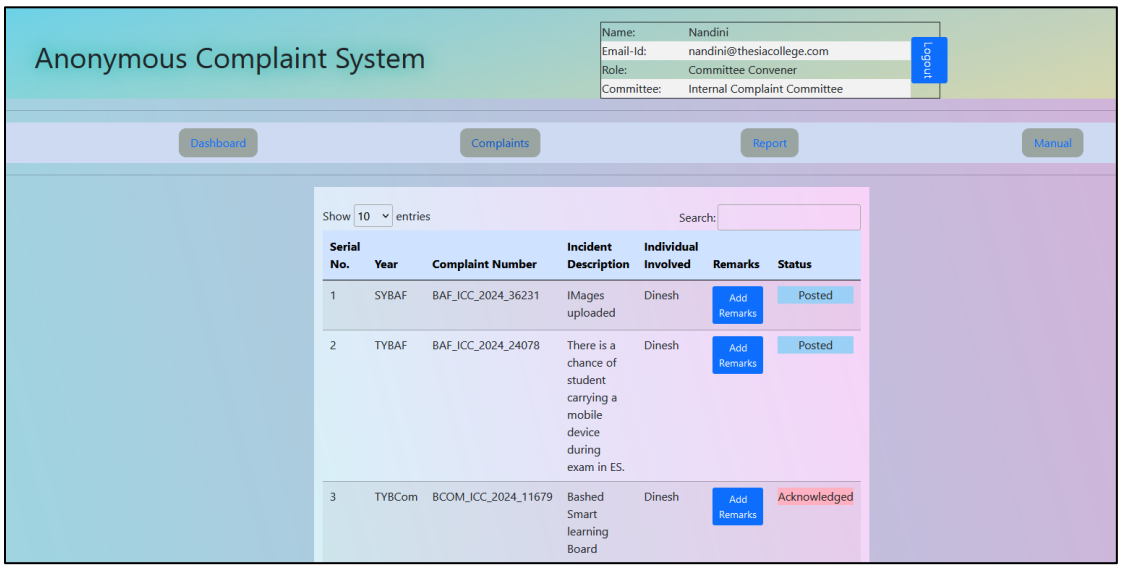


Fig. 6.19: Complaint in process of resolving by Committee Convener.

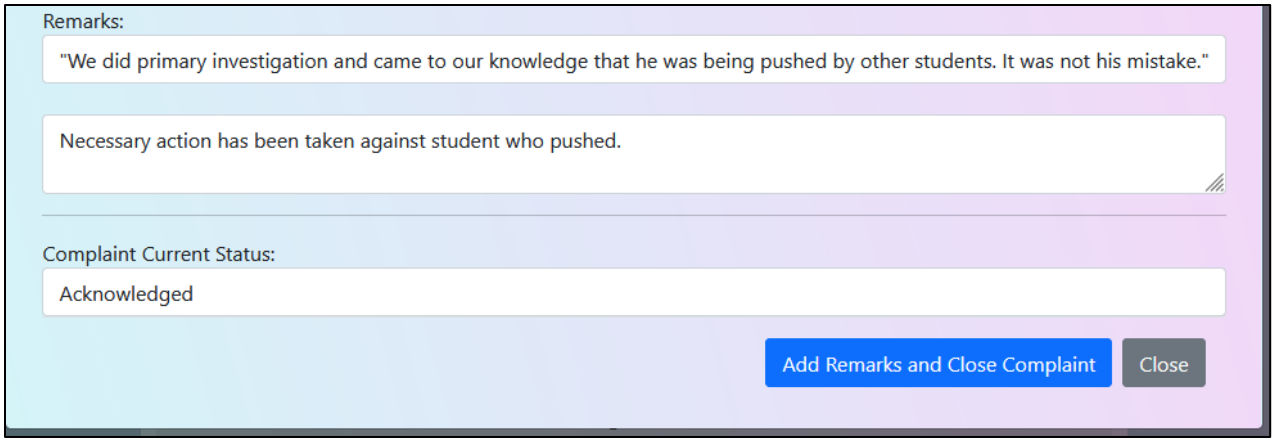


Fig. 6.20: Remarks added by Committee Convener.

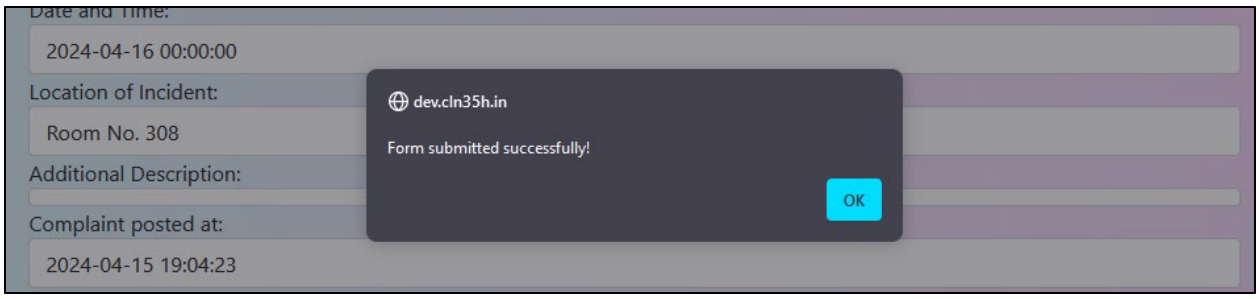


Fig. 6.21: Complaint has been updated successful message.

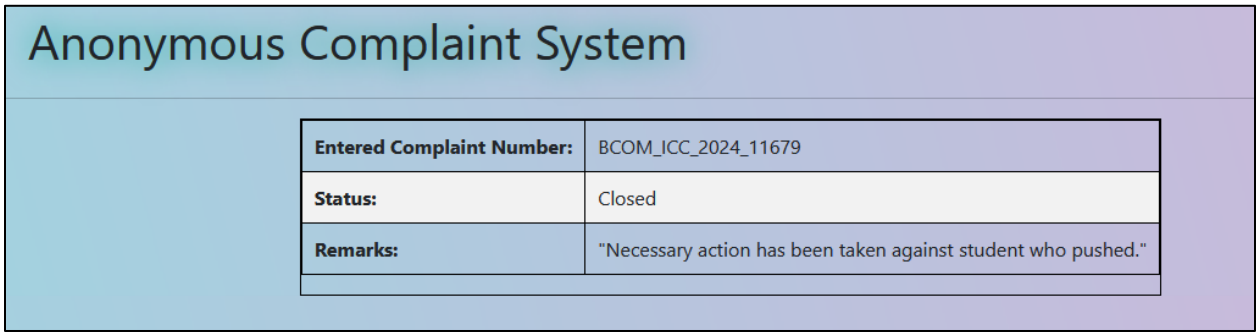


Fig. 6.22: Complaint Status after closing remarks.

6.2 User Documentation

6.2.1 Documentation for user

Visit link **https://dev.cln35h.in/**

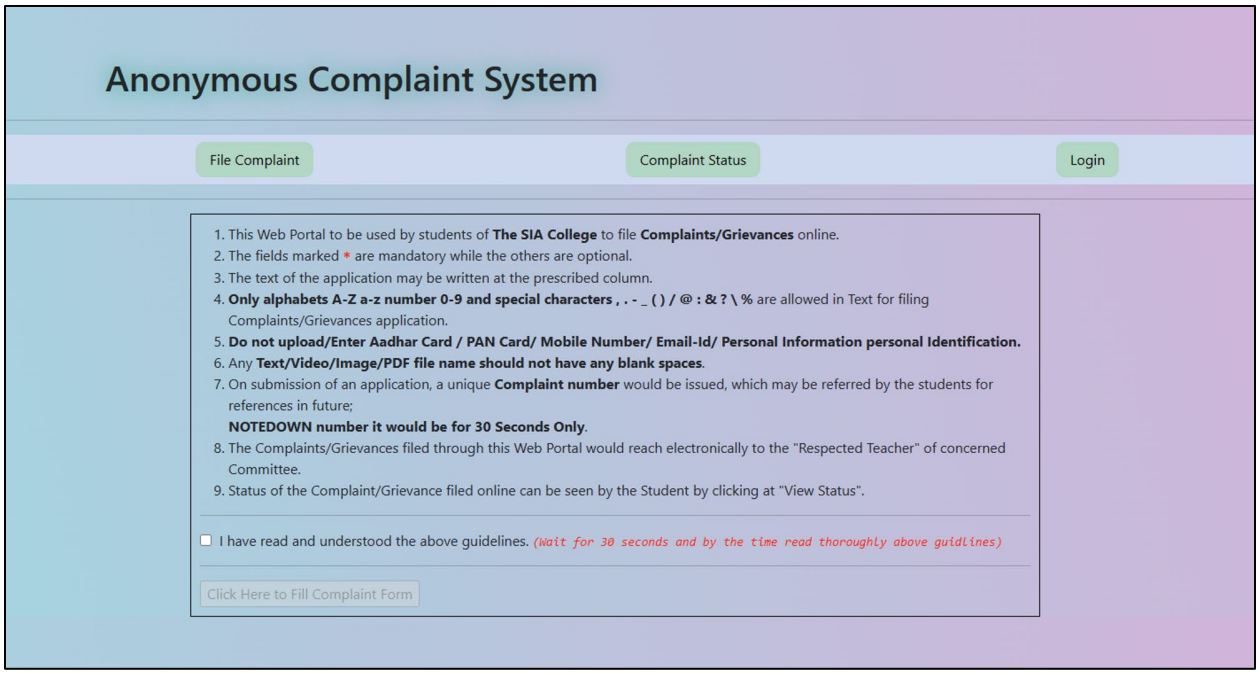


Fig. 6.23: Index Page

To file a complaint read guidelines and wait for 30 Seconds and then click on "*Click Here to Fill Complaint Form*" then the complaint form will be displayed.

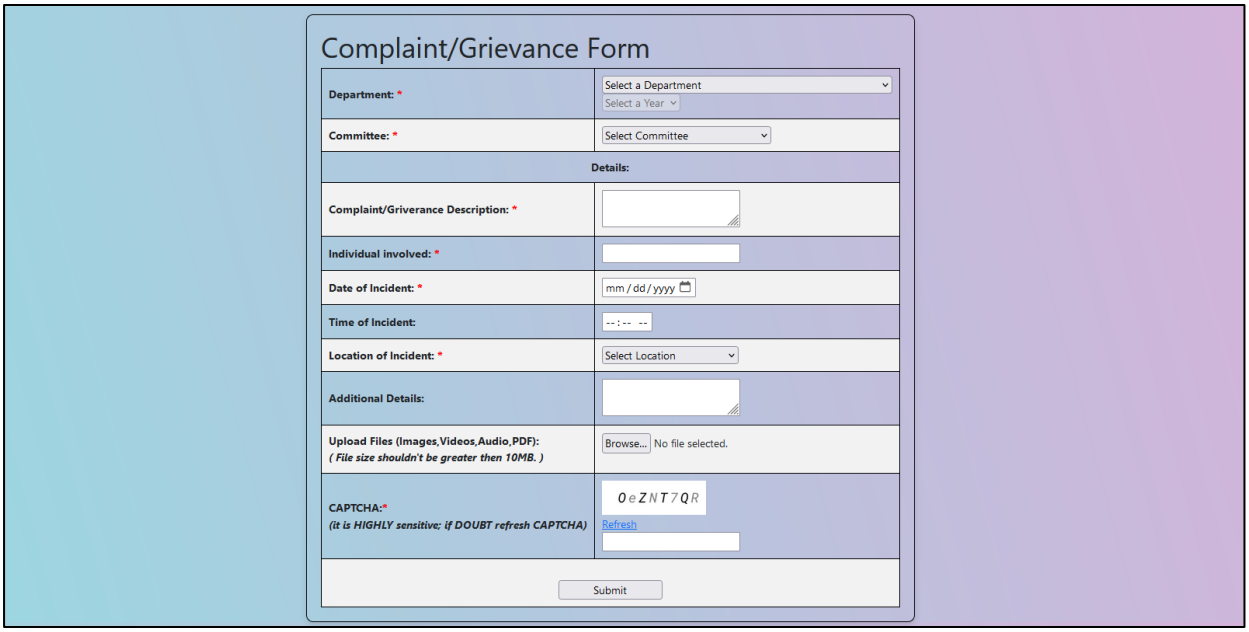


Fig. 6.24: Complaint form

After filing details properly and entering "*CAPTCHA*" click on "*Submit*" button. It will then display the complaint number and the website will be redirected to google.com after 30 seconds so note the complaint number and do not share it with anybody.

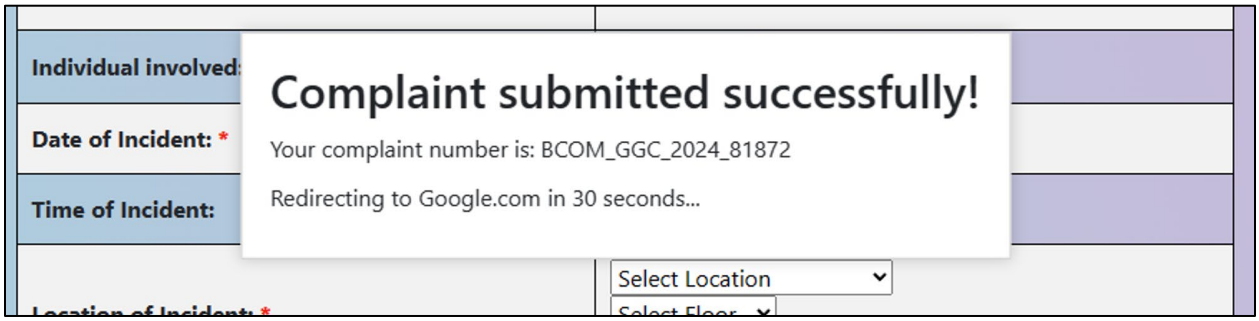


Fig. 6.25: Complaint Submitted successfully.

To check status of complaint, click on "*Complaint Status*" from navbar. It will open a new link, enter your "*Generated Complaint Number*".

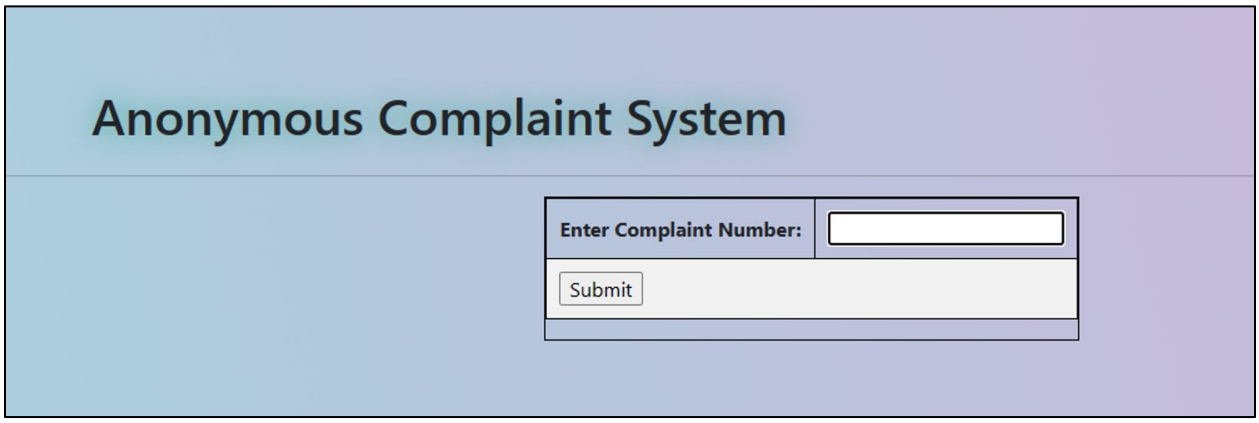


Fig. 6.26: Complaint Status Form

After entering "*Generated Complaint Number*" click on "*Submit*" button. It will display the status of complaint.

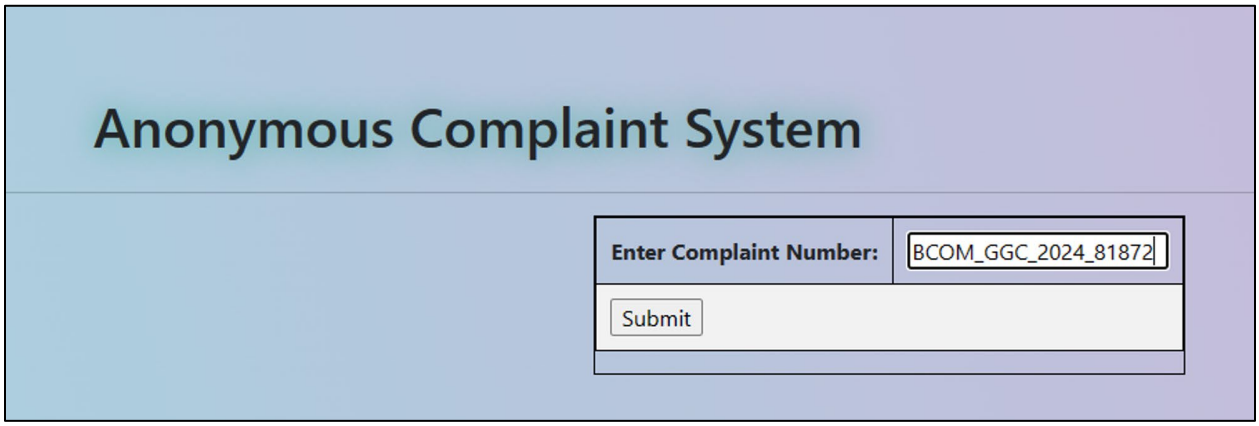


Fig. 6.27: Complaint Number entered.

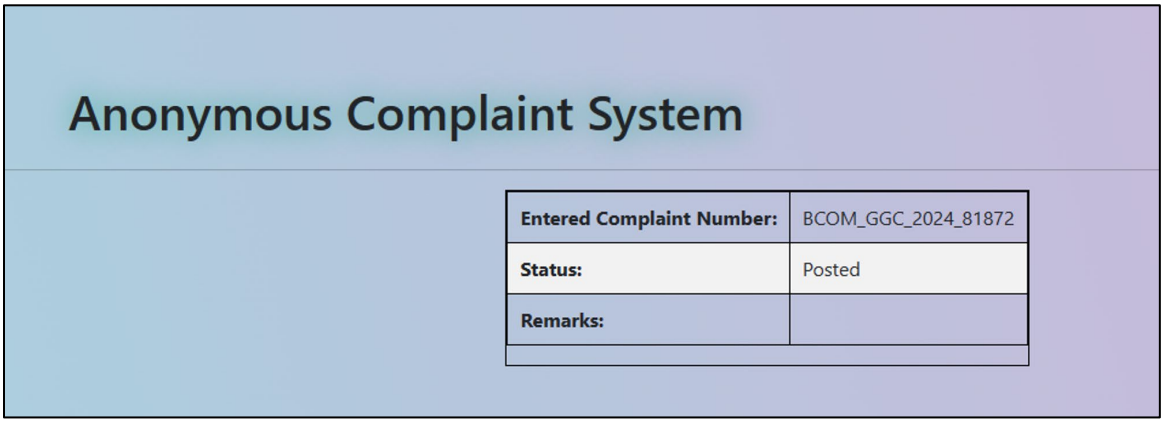


Fig. 6.28: Displaying Status of Complaint

6.2.2 Documentation for Admin

Visit link **https://dev.cln35h.in/**

For logging in system click on "*Login*" from navbar then login form will be displayed. Enter Email Id. and Password allotted to you. After entering click on "*Login*" button. As per the role assigned to the user your Dashboard will be displayed.

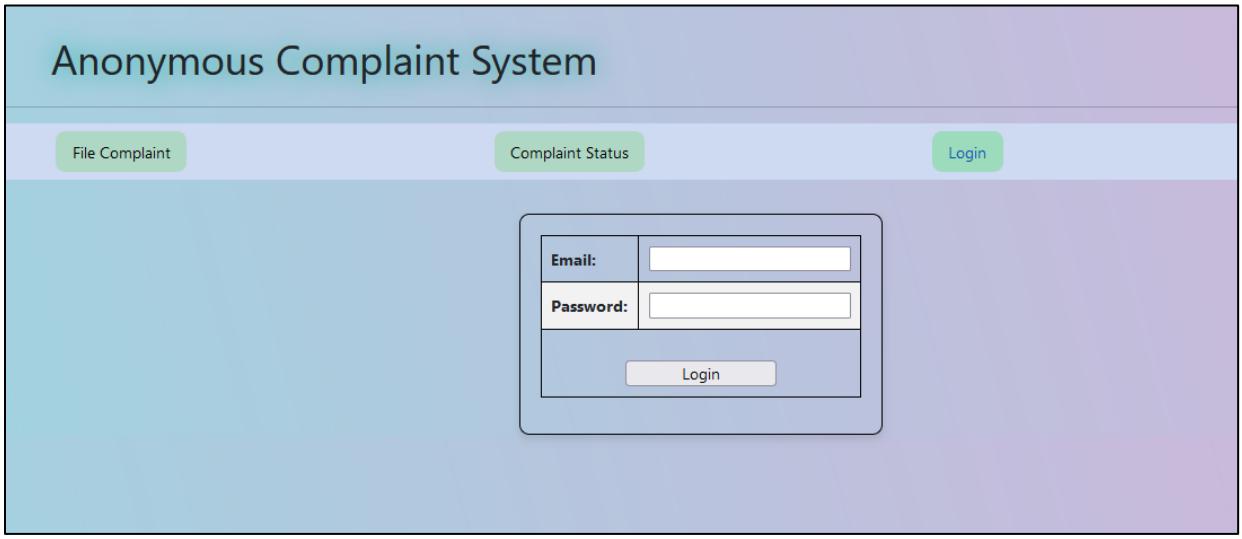


Fig. 6.29: User login form

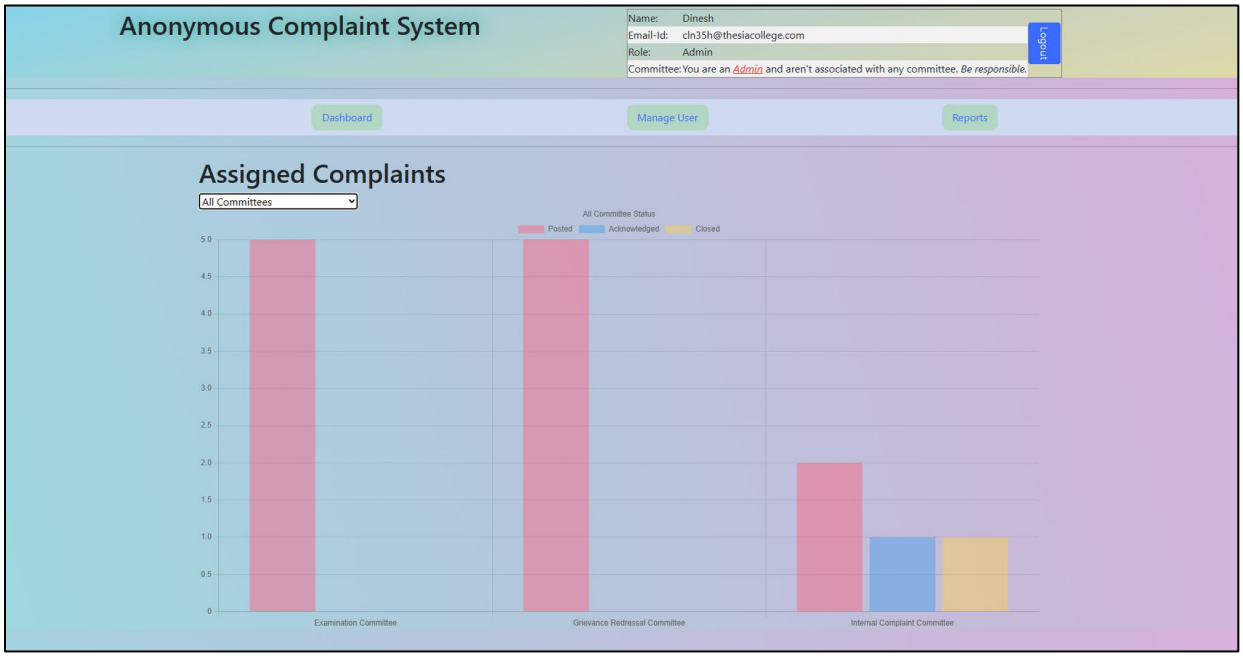


Fig. 6.30: Admin Dashboard

To add, update, delete user or change role of user click on "*Manage User*".

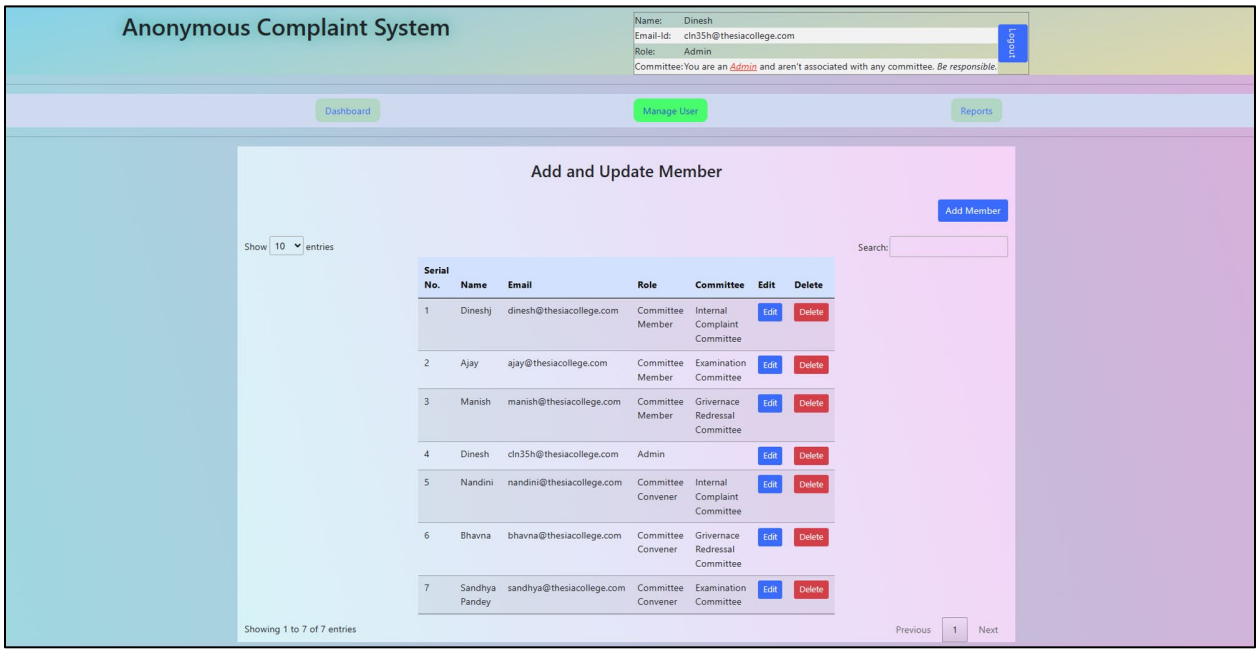


Fig. 6.31: Manage User

To add a new user, click on "Add Member"; it will open a modal and fill details.

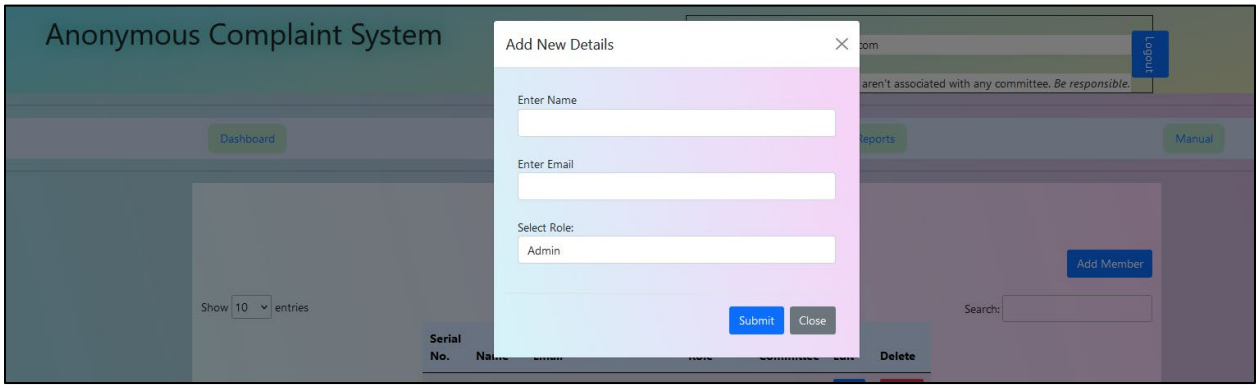


Fig. 6.32: Modal for Adding User

Enter Name, Email Id. and Role. Note that Email Id. should end with "*@thesiacollege.com*" and User who is assigned Admin can't have any Committee assigned to it.

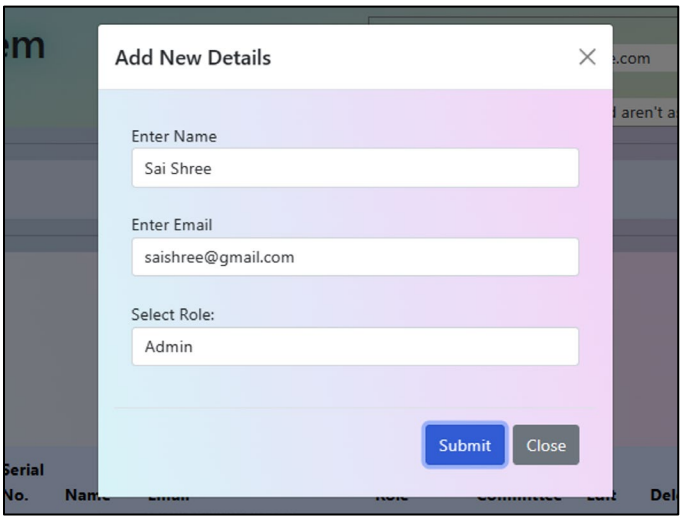


Fig. 6.33: Added User details in modal.

After entering details click on "*Submit*" button. Then it will display successfully User added with password and will be displayed with other users.

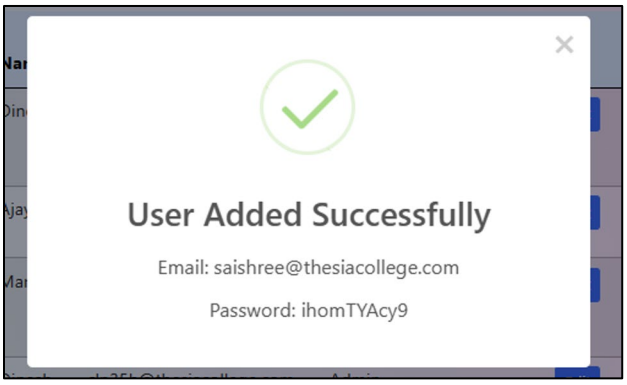


Fig. 6.34: User Added Modal

7	Sandhya Pandey	sandhya@thesiacollege.com	Committee Convener	Examination Committee	Edit	Delete
8	Sai Shree	saishree@thesiacollege.com	Admin		Edit	Delete

Fig. 6.35: User displaying in table.

In future if any changes are to be made for User like change of Role, forgot password. It can be done by clicking on "*Edit*" Button beside the User Details.

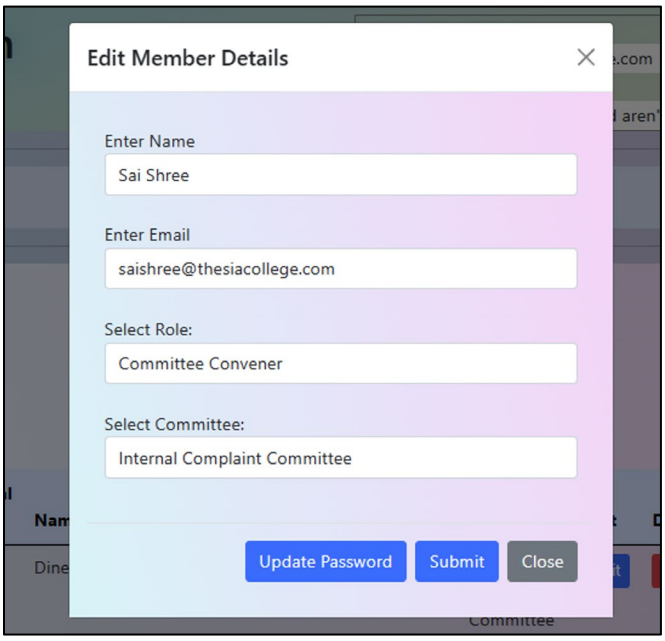


Fig. 6.36: Edit Modal

You can make necessary changes from here and click on "*Submit*" button. It will update user details and if you want to update password to the user click on "*Update Password*" it will display modal of User updated successfully.

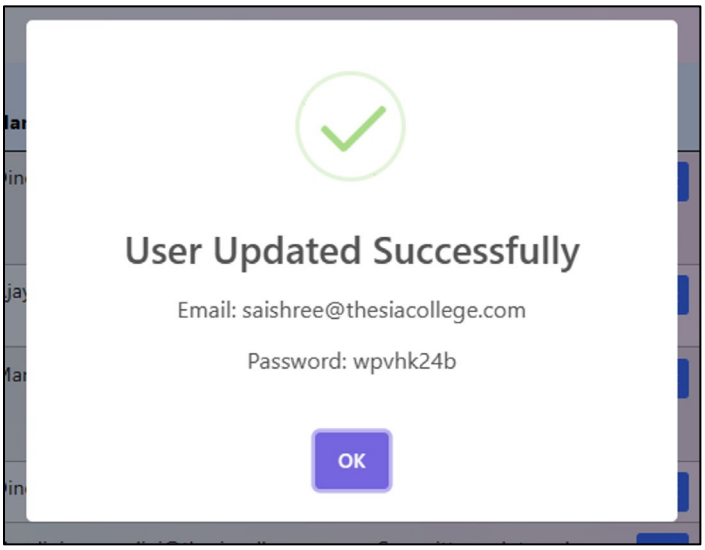


Fig. 6.37: Update Modal

To delete user there is "*Delete*" button beside "*Edit*" button click on it; then it will it display alert asking should user be deleted.

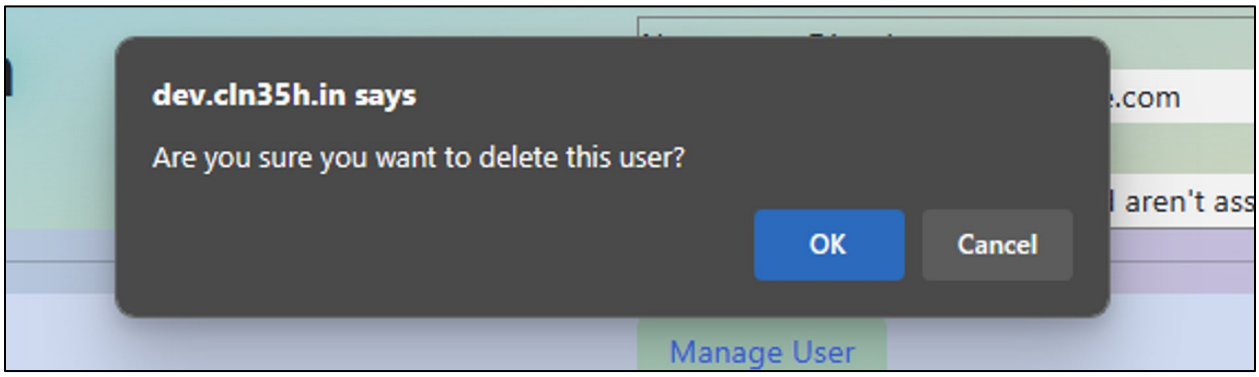


Fig. 6.38: Confirmation alert to delete User.

If press "OK" it will delete the user.

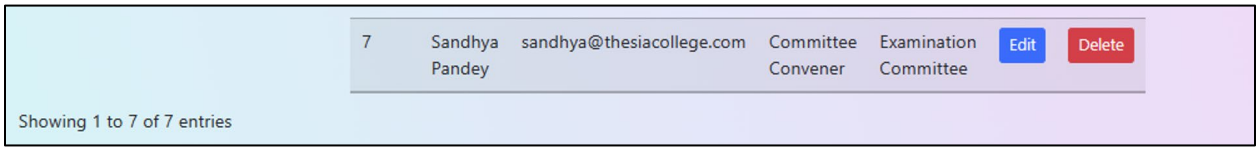


Fig. 6.39: User deleted successfully.

To view the report, click on "Reports" from navbar.

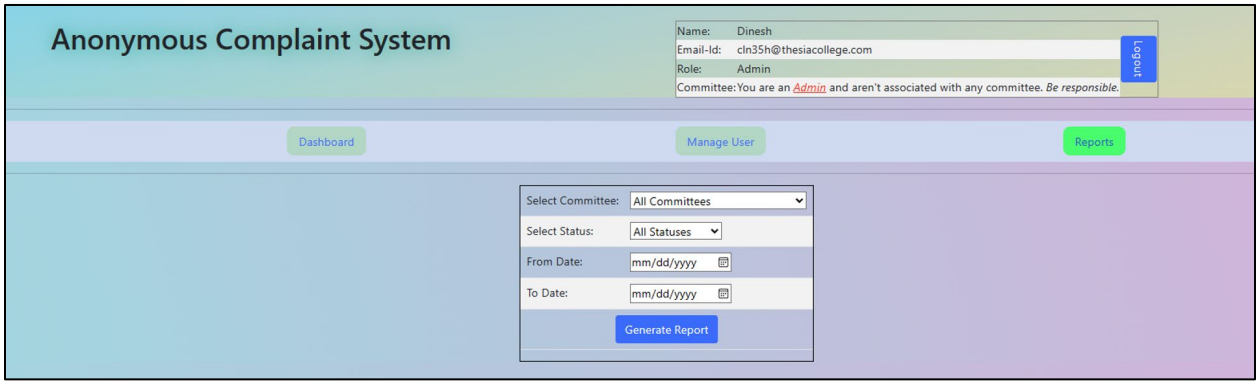


Fig. 6.40: Report module for admin

Select proper dates from particular period of time. As per the use you can select Committee as well as their Status of complaints assigned to them.

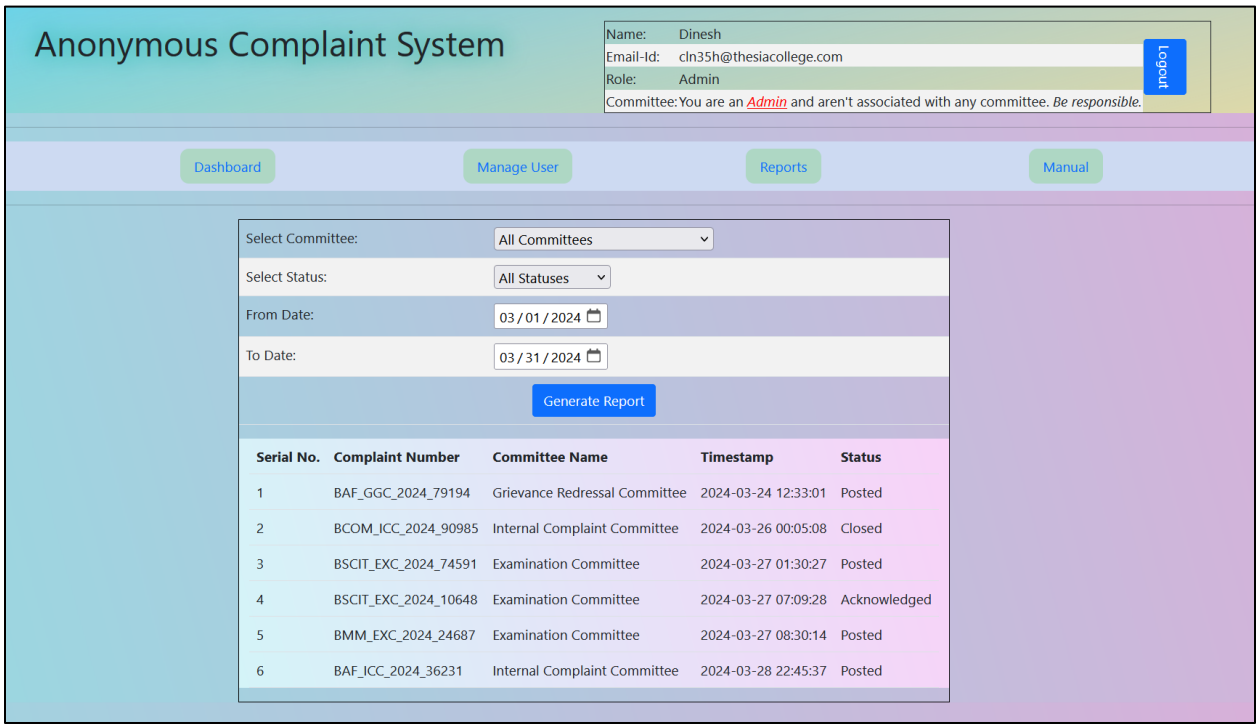


Fig. 6.41: Report displayed for Admin

To logout click on "*Logout*", it will then give alert asking "*Are you sure you want to logout*" click on "*OK*" it will then logout and display Index page.

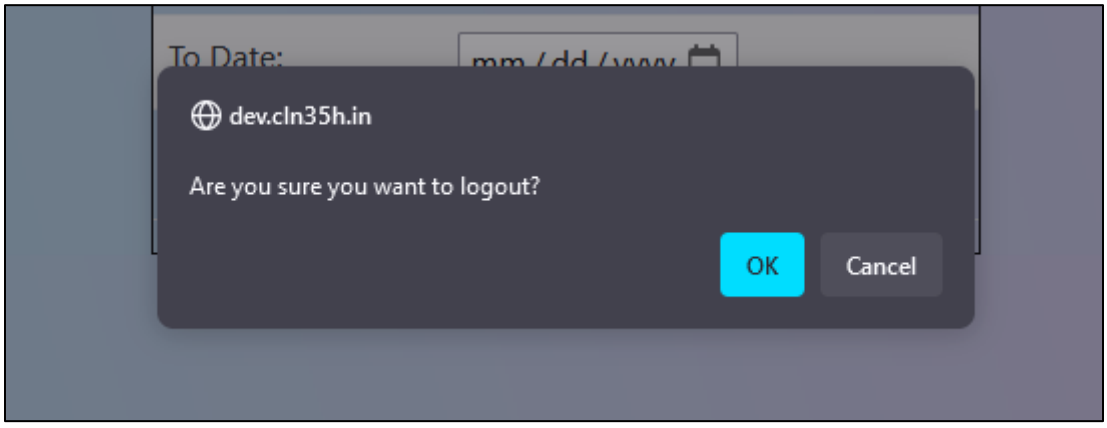


Fig. 6.42: Alert for logging out.

6.2.3 Documentation for Committee Convener and Committee Member

Visit link **https://dev.cln35h.in/**

For logging in system click on "*Login*" from navbar then login form will be displayed. Enter Email Id. and Password allotted to you. After entering click on "*Login*" button. As per the role assigned to you your Dashboard will be Displayed.

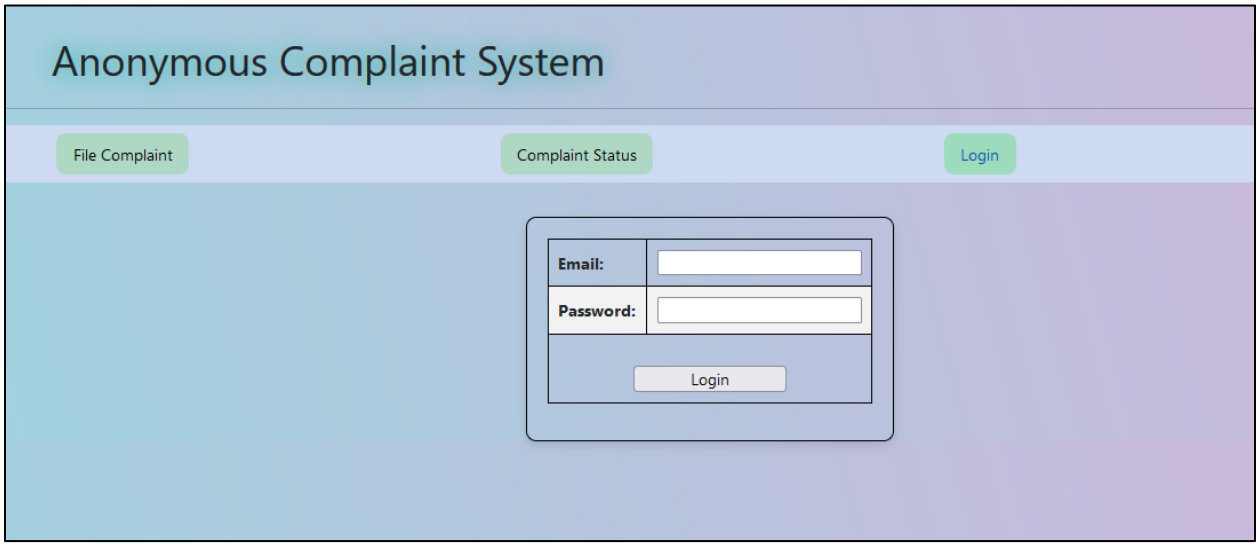


Fig. 6.43: User login form

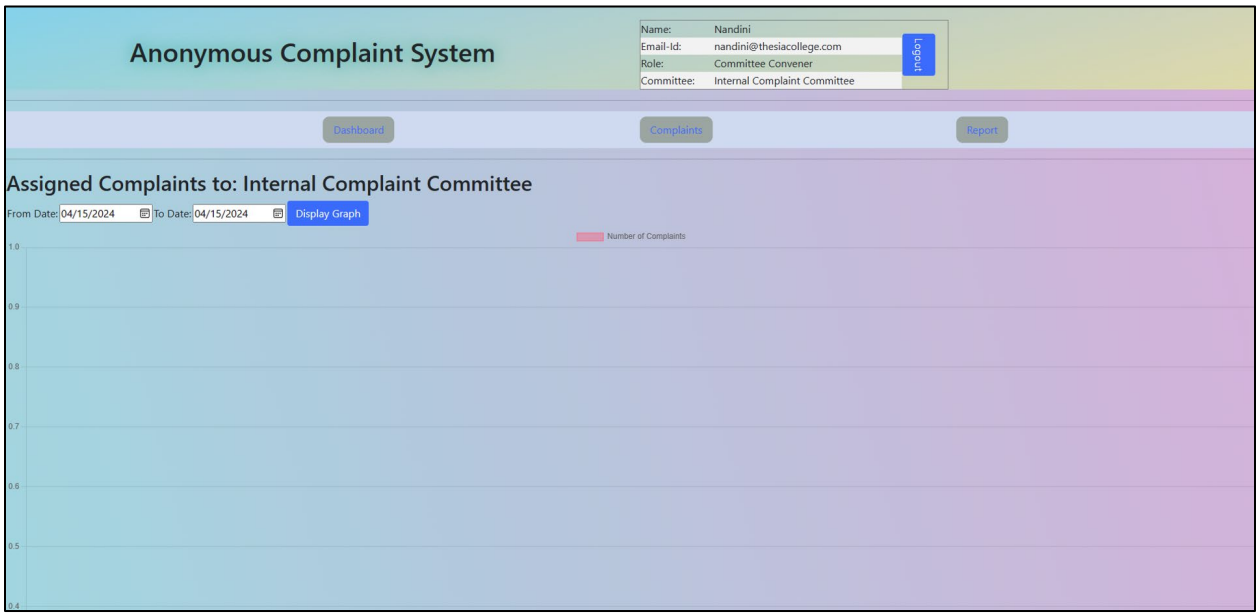


Fig. 6.44: Dashboard for Committee Convener

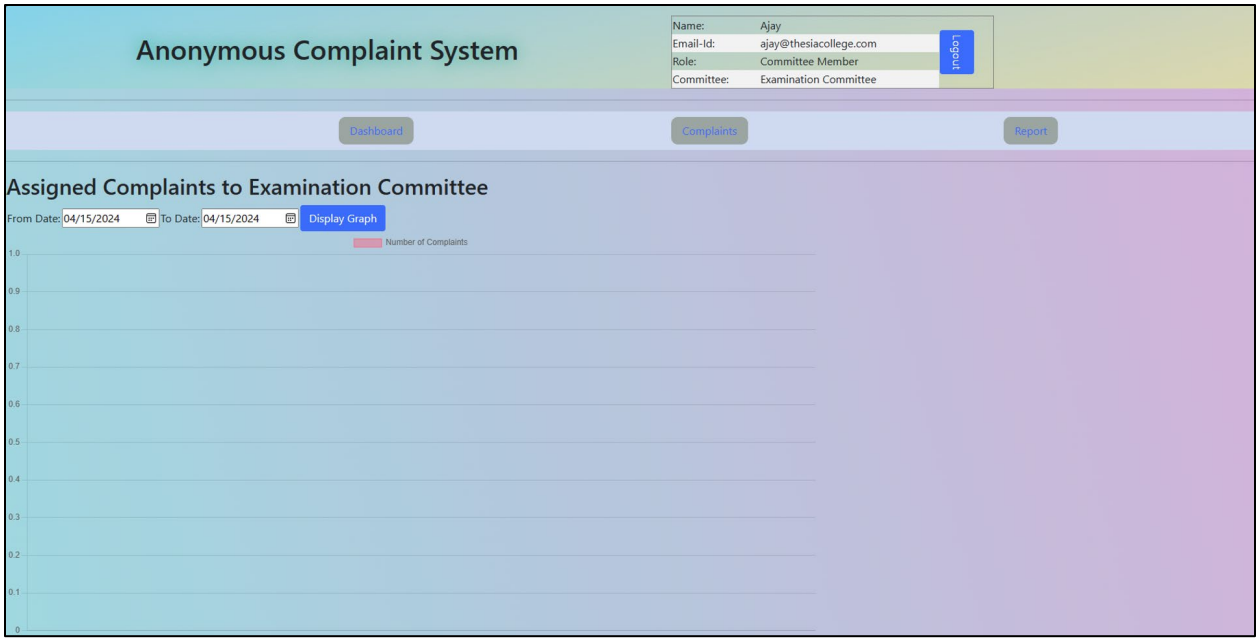


Fig. 6.45: Dashboard for Committee Member

Display Graph of that particular committee from certain period by selectin dates.

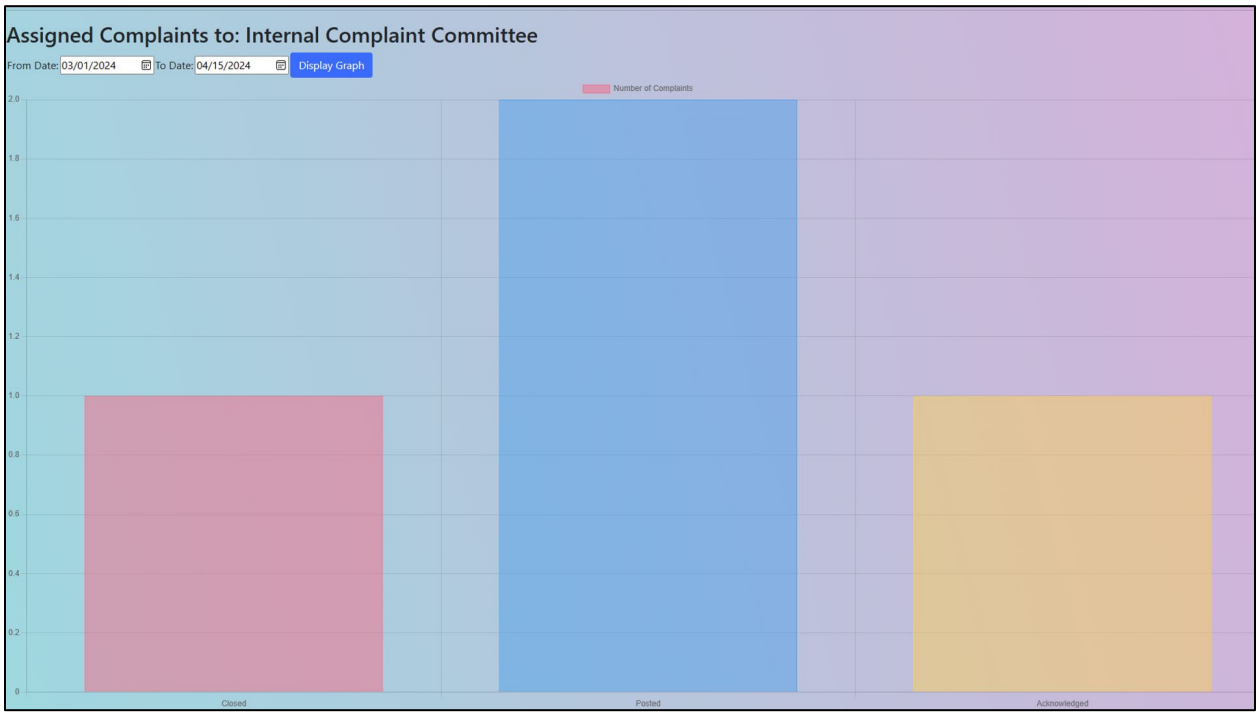


Fig. 6.46: Graph on Dashboard for Committee Convener and Member

Click on "*Complaints*" from navbar to display all complaints.

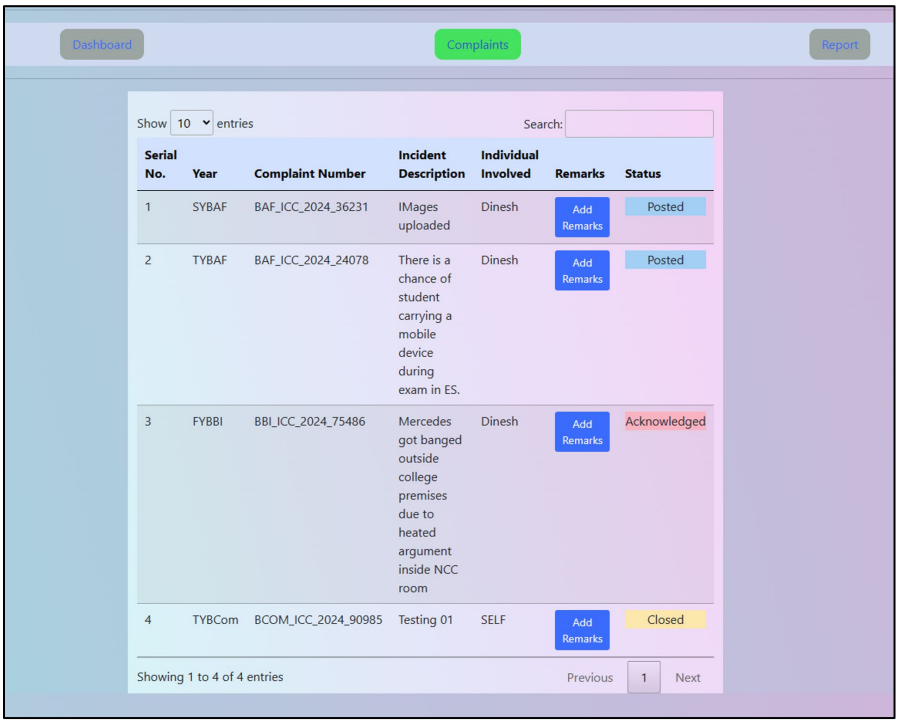


Fig. 6.47: Complaints table for Committee Convener

Fig. 6.48: Complaints table for Committee Member

Search for particular complaint associated with that committee.

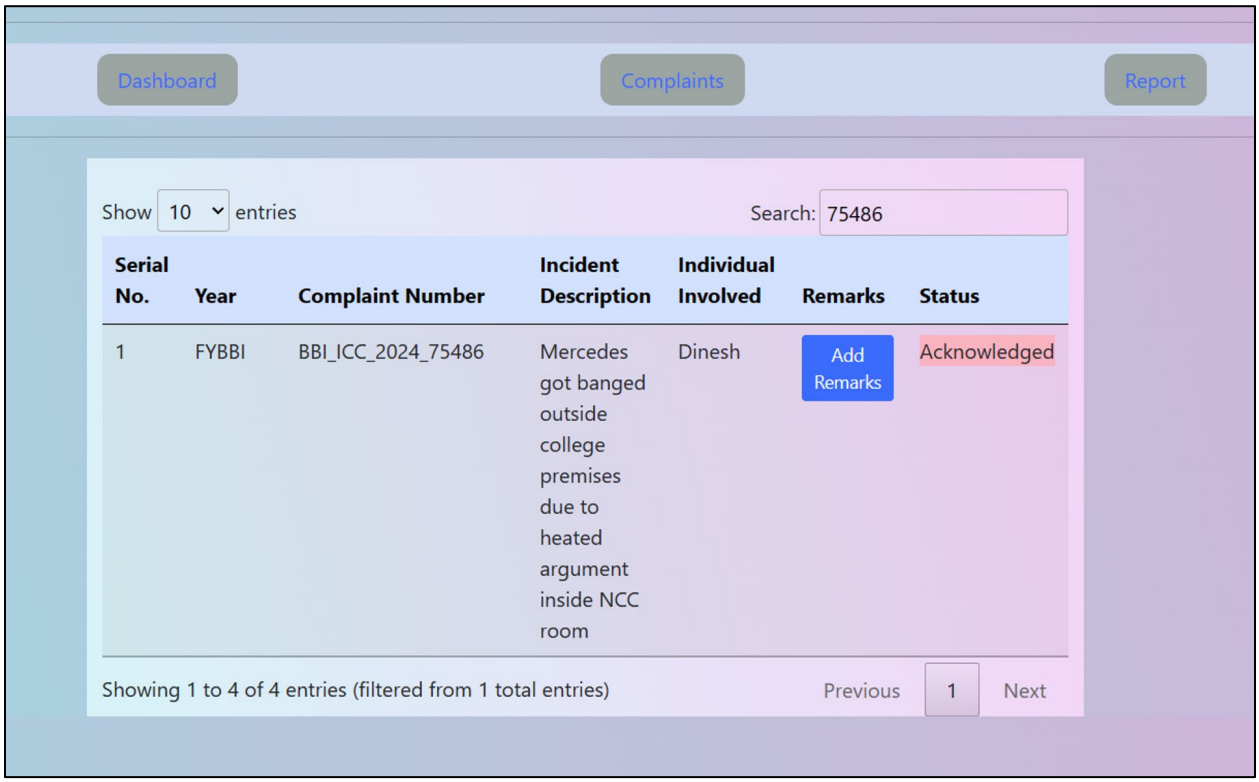


Fig. 6.49: Searching in Complaints table from Committee Convener

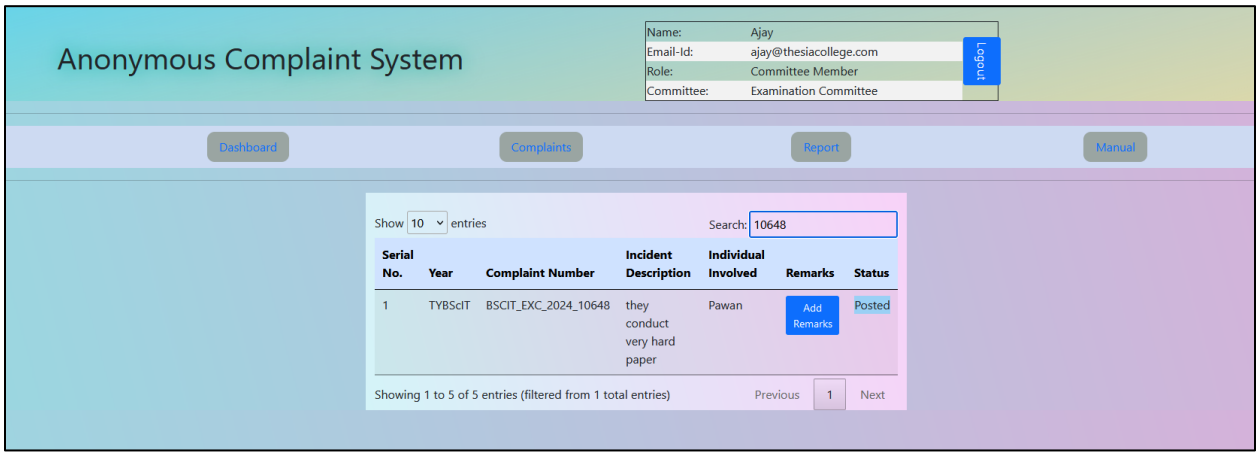


Fig. 6.50: Searching in Complaints table from Committee Member

To take action against complaint click on "*Add Remarks*" modal will be displayed having complaint details of that particular complaint.

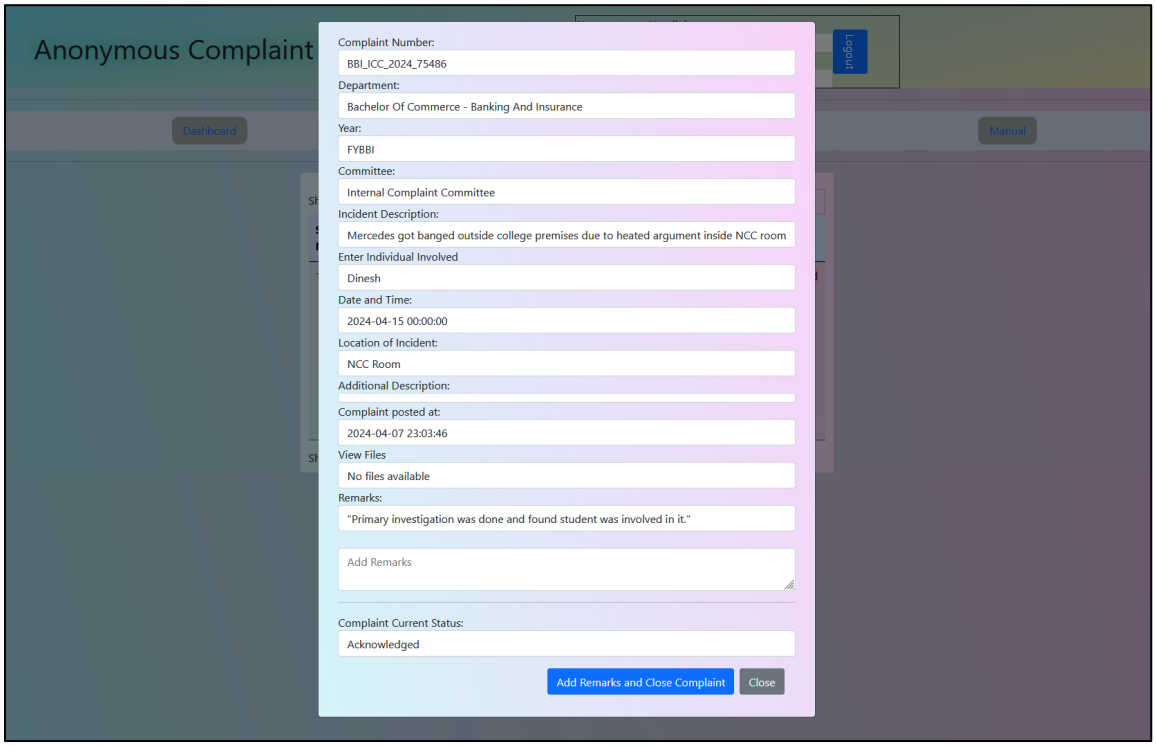


Fig. 6.51: Complaint details Modal for Committee Convener

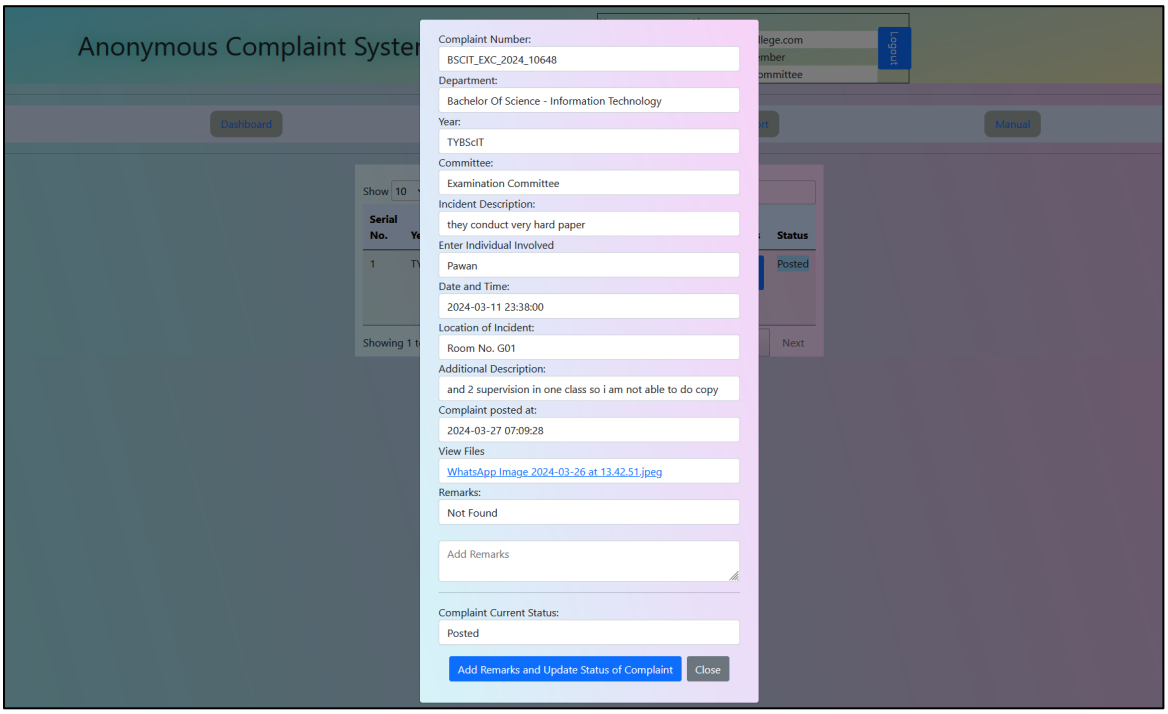


Fig. 6.52: Complaint details Modal for Committee Member

Give proper response by adding remarks; if Committee Convener is logged in then click on "*Add Remarks and Close Complaint*" as doing this will update status of complaint to "*Closed*" and "*Remarks*" will be added, for another user that is Committee Member is logged in then click on "*Add Remarks and Update Status of Complaint*" doing this will update status to "*Acknowledged*" and "*Remarks*" will be added for the same, also alert will be displayed as "*Form submitted uccessfully!*"

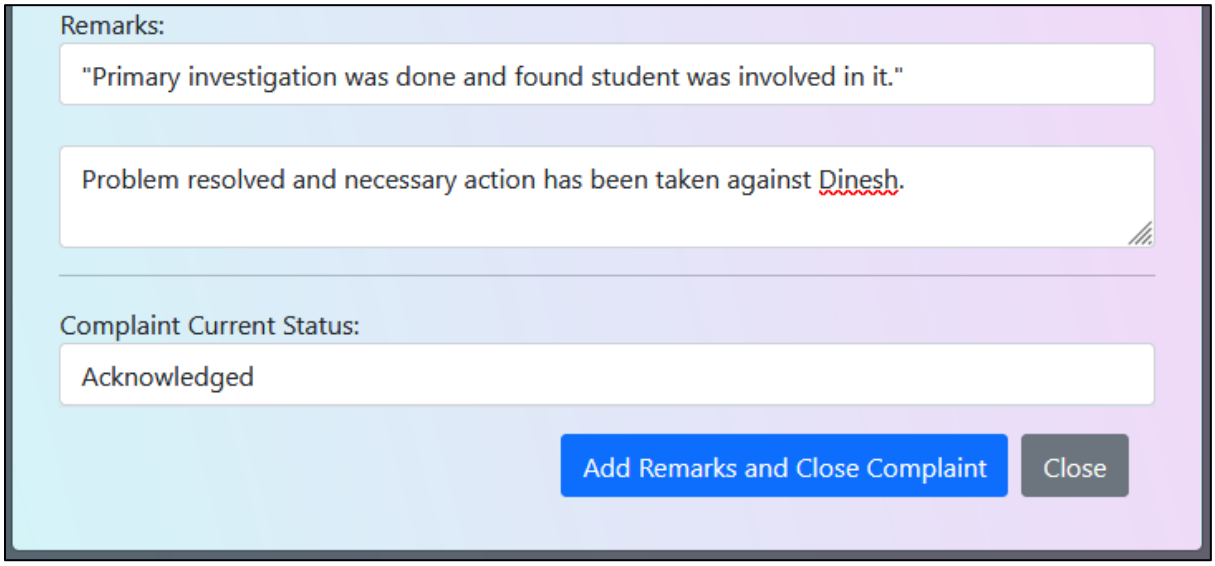


Fig. 6.53: Adding remarks and updating Status for Committee Convener

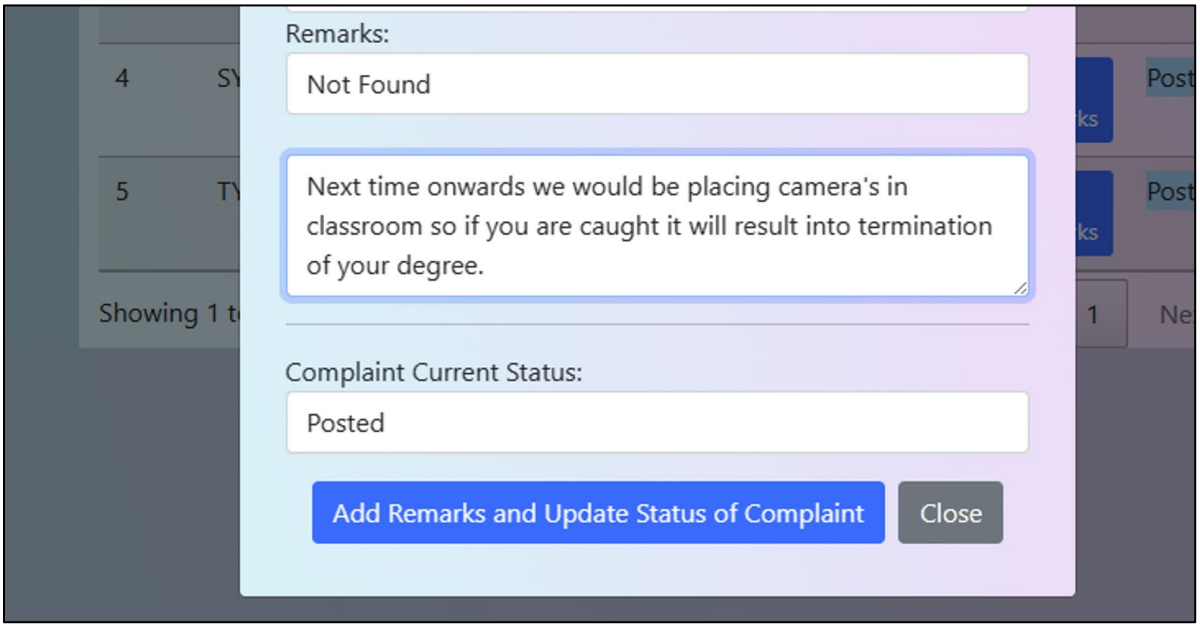


Fig. 6.54: Adding remarks and updating Status for Committee Member

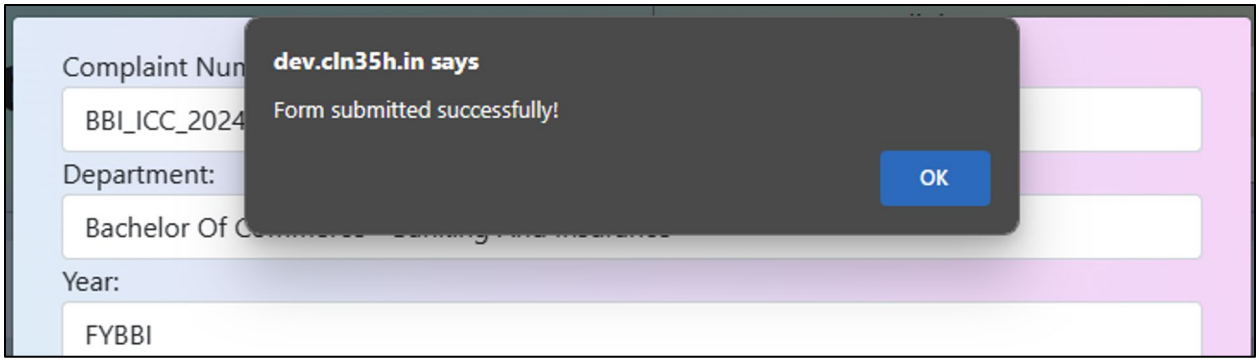


Fig. 6.55: Alert for remarks successfully added.

For report Generation click on "*Report*" from navbar it will open new link. Select proper dates from particular period of time, as per the use you can "S*elect Status*" that you want to view.

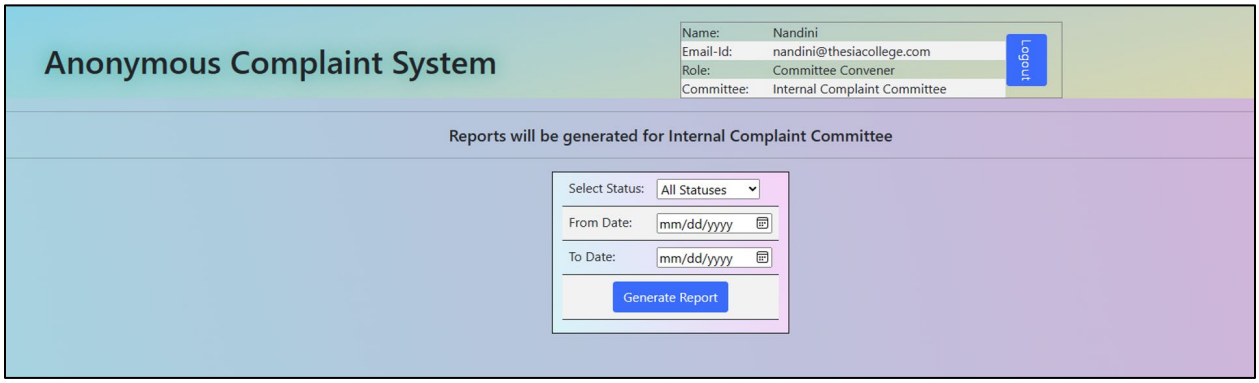


Fig. 6.56: Options for selection of Status for making report.

After fetching it will display "Print report" option to take print click on it.

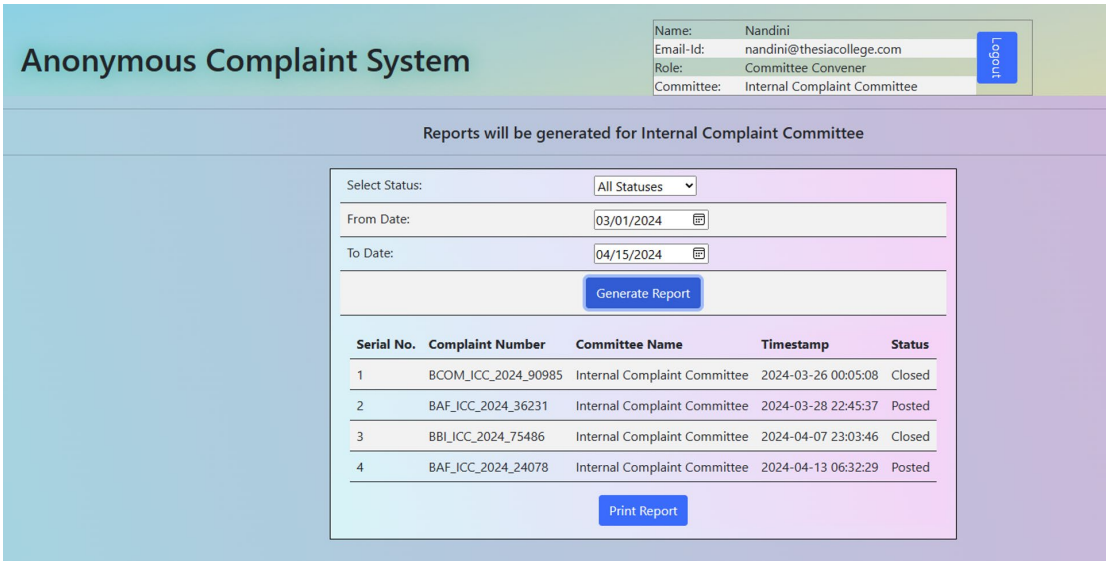


Fig. 6.57: Report table generated.

Chapter 7: Conclusion

7.1 Conclusion

This project is made towards a new approach of filing complaints online that can result in some discipline among students and appropriate use of technology. And bring forward thinking of taking action/ acting against anybody who is in authority of power or not. Having multiple options for individuals to express their viewpoints can help streamline processes and mitigate unnecessary workflow. To maintain complaints and their details, this project uses various rich features but for the period they are limited as maintaining complaints there are various guidelines and necessary framework/regulations to be maintained.

In this project various methodologies are used to maintain secrecy of complaints. For e.g. When a complaint is being successfully registered it generates a complaint number that is unique in itself as it helps in displaying the Committee Name it is being selected and handling of it. To maintain the secrecy of Complainant after filing a complaint from Browser history it would remove/delete the link that was used.

For handling complaints by the respective committee, a mechanism is used to just display complaints as per the role assigned to them. For e.g. Committee Convener can add remark and make status as close the complaint; Committee Member can add remark but can change their status to Acknowledged because Committee Convener is the Head of Committee, and it has rights to close the complaints. This would help in giving proper authority over each other to not misuse their power to acknowledge / close complaint without proper response/Remarks.

Admin in this project can add/ remove users and assign roles to them. As this gives Admin a centralized power to manage users by doing these there is less chance of complaints to be made inattentive or treated as vague or uninformative.

7.1.1 Significance of the System

The most significant part of the project is that every complaint should be taken with utmost serious priority and not ignore or take lenient, to do that whichever complaint is being posted it displays old complaints on top of others. As there are various complaints received it gets assigned to the committee that is selected as it helps that committee in managing / handling those complaints; with these Committee Member can take action and later on Committee Convener can close complaints with remarks.

As Committee Member and Committee Convener are bound to add remarks, students who have filed complaints can later check the status of their complaints and remarks. Most importantly it does not display the name of any respected Committee Member/ Committee Convener but only remarks given by them. By doing so it helps in maintaining anonymity at all stages of the complaint management process.

7.2 Limitations of the System

Due to the advancement of technology, there are some limitations in this project as well. This project is made for a limited number of people or for small organizations like college and school. For big organizations these functionalities won't be enough and maintain hierarchical integrity as well as anonymity. For e.g.

- Storing log entry of login, logout, and handling of complaints like timing/log of doing activity on complaints.
- Updating Remarks is another problem where complaints get overwritten from other committee members when it updates status.
- Transferring complaints among committees as some complaints are not for that committee it should be able to transfer complaints to another respective committee.
- After checking the status of complaints by students if they feel unsatisfactory it should allow them to make a review on complaints filed by them.

7.3 Future Scope of the Project

The future scope for this project is, as we seen earlier that it is for small organization making use it for big organization and to do so it must cover all the limitations by adding more functional, security features and more transparent management of complaints are to be made. For e.g. Storing logs in SQL DB are not that optimum instead we can use Apache Druid, Prometheus or Time-Series Database. Same for displaying latest updated Remarks we can store in NoSQL Database and if there is a Remark given on that Complaint earlier also can be stored without the help of denormalization. For security concerns we can utilize WEB3 technologies like Smart contracts so that contents of complaints or any can't be overwritten or deleted.

Plagiarism Report

Chapter 1: Introduction

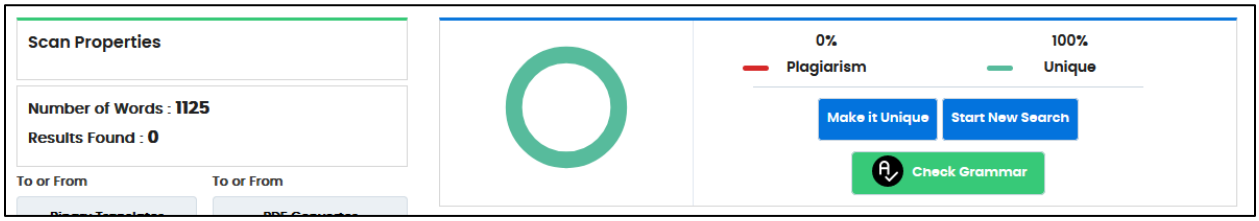


Fig. P.1: Plagiarism Report for Chapter 1

Chapter 2: System Analysis Existing System

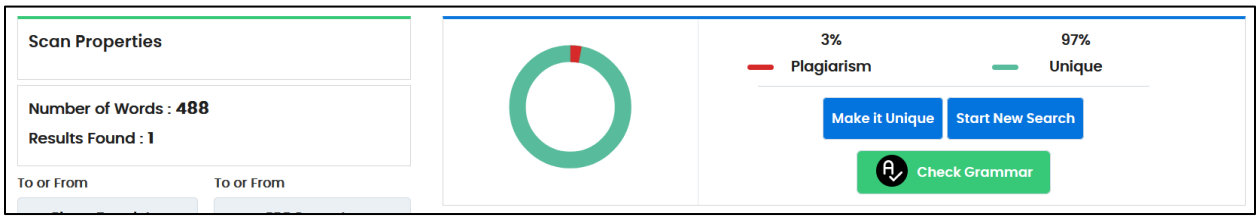


Fig. P.2: Plagiarism Report for Chapter 2

Chapter 3: Requirement & Analysis

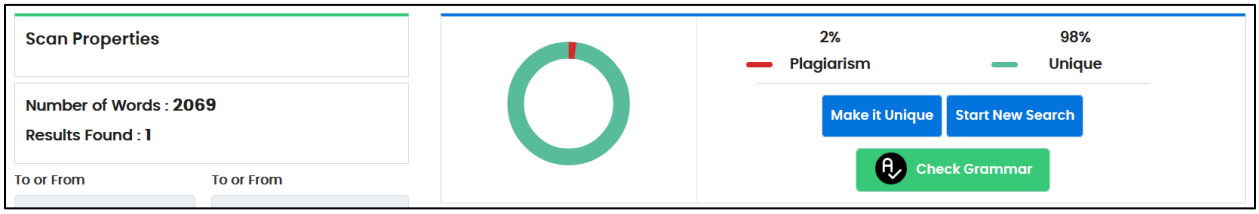


Fig. P.3: Plagiarism Report for Chapter 3

Chapter 4: System Design

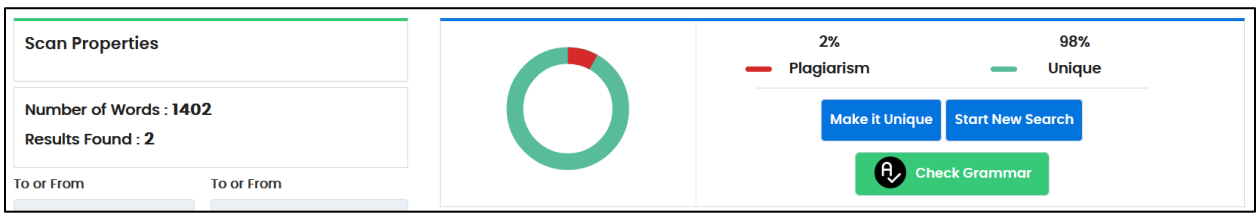


Fig. P.4: Plagiarism Report for Chapter 4

Chapter 5: Implementation & Testing

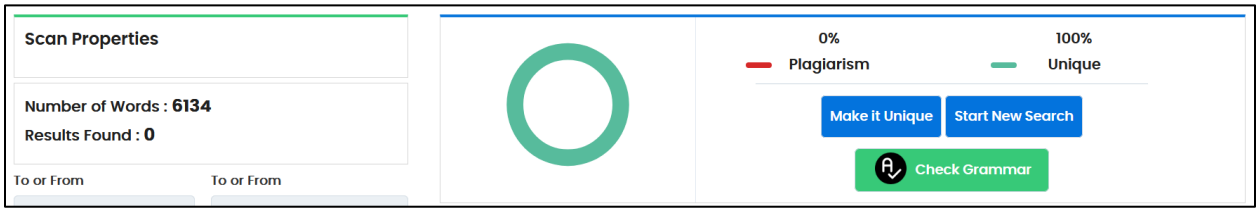


Fig. P.5: Plagiarism Report for Chapter 5

Chapter 6: Results & Discussion

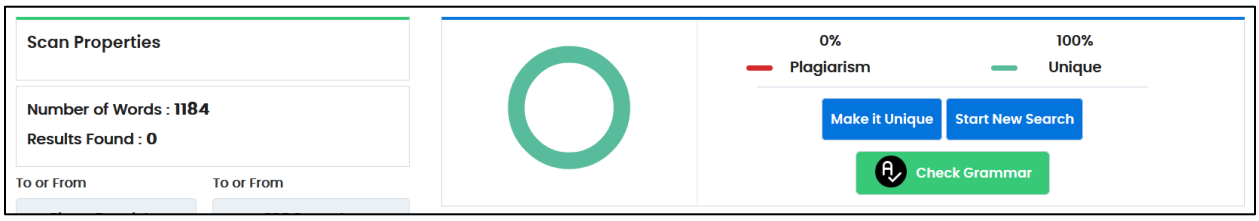


Fig. P.6: Plagiarism Report for Chapter 6

Chapter 7: Conclusion

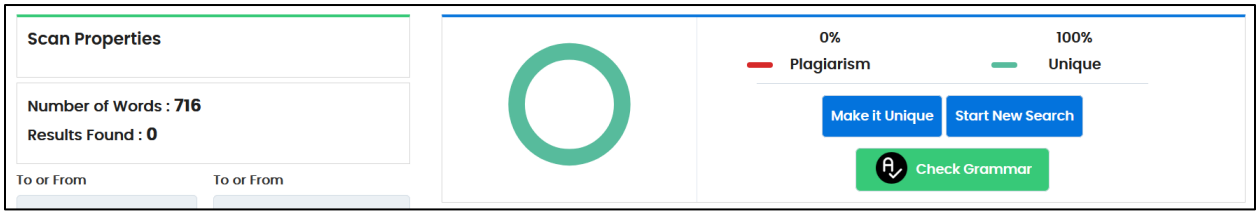


Fig. P.7: Plagiarism Report for Chapter 7

REFERENCES

- PHP, MySQL, & JavaScript All-in-One For Dummies by Richard Blum
- Expert PHP and MySQL by Andrew Curioso, Ronald Bradford, Patrick Galbraith
- Introduction to Software Testing by Paul Ammann, Jeff Offutt
- https://www.php.net/manual/en/password.constants.php
- https://tutorial101.blogspot.com/2023/02/php-mysql-pdo-crud-server-side-ajax.html