

SMART PARKING ASSISTANCE HUB

A PROJECT REPORT

Submitted by

DHARSHINI.V

(Reg. No: 24MCR019)

KANAGA DURGA.M

(Reg. No: 24MCR049)

KARTHIKEYAN.P

(Reg. No: 24MCR051)

in partial fulfilment of the requirements

for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS



KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DECEMBER 2024

DEPARTMENT OF COMPUTER APPLICATIONS**KONGU ENGINEERING COLLEGE****(Autonomous)****PERUNDURAI, ERODE – 638 060****DECEMBER 2024****BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “**SMART PARKING ASSISTANCE HUB**” is the bonafide record of project work done by **DHARSHINI V** (Reg.No: 24MCR019), **KANAGA DURGA M** (Reg.No: 24MCR049) and **KARTHIKEYAN P** (Reg.No:24MCR051) in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications of Anna University, Chennai during the year 2024-2025.

SUPERVISOR**HEAD OF THE DEPARTMENT****(Signature with seal)****Date:**

Submitted for the end semester viva-voice examination held on _____

INTERNAL EXAMINER**EXTERNAL EXAMINER**

DECLARATION

I affirm that the project report entitled “**SMART PARKING ASSISTANCE HUB**” being submitted in partial fulfilment of the requirements for the award of Master of Computer Applications is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

DATE:

DHARSHINI V

(Reg. No: 24MCR019)

KANAGA DURGA M

(Reg. No: 24MCR049)

KARTHIKEYAN P

(Reg. No. 24MCR051)

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date:

Name and Signature of the Supervisor
(Dr.P.VIJAYAKUMAR)

ABSTRACT

The "Smart Parking Assistance Hub" is an innovative solution designed to address the growing challenges of parking space management in urban areas. This project leverages cutting-edge technologies including React for the frontend, Python for backend development, MongoDB as the database system, and OpenCV for real-time parking space detection.

The system aims to provide users with an intuitive platform that allows them to find and reserve parking spots in real-time. Using computer vision techniques via OpenCV, the backend is capable of detecting available parking spaces in parking lots or on streets by processing live video feeds. The frontend, built with React, offers users an easy-to-navigate interface, displaying live data and providing features such as space reservation, navigation to available spots, and historical parking data.

MongoDB is employed to store and manage user data, parking space availability, and system logs efficiently, ensuring high scalability and responsiveness. This project aims to enhance urban mobility by optimizing parking space usage, reducing the time spent searching for parking, and minimizing traffic congestion.

The "Smart Parking Assistance Hub" is a step toward the future of smart cities, promoting efficient use of resources, improving the parking experience, and contributing to sustainable urban living.

ACKNOWLEDGEMENT

We respect and thank our correspondent THIRU.A.K.ILANGO, B.Com., MBA., LLB., and our Principal Dr.V.BALUSAMY B.E(Hons)., M.Tech., Ph.D., Kongu Engineering College, Perundurai for providing us with the facilities offered.

We convey our gratitude and heartfelt thanks to our Head of the Department Dr.A.TAMILARASI M.Sc., M.Phil., M.Tech., Ph.D., Department of Computer Applications, Kongu Engineering College, for her perfect guidance and support that made this work to be completed successfully.

We also wish to express my gratitude and sincere thanks to our project coordinators Dr.T.KAVITHA MCA., M.Phil., Ph.D., Assistant Professor(Sr.G) Department of Computer Applications, Kongu engineering College, who have motivated us in all aspects for completing the project in the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide Dr.P.VIJAYAKUMAR MCA., M.Phil., Ph.D., MBA., Associate Professor Department, Department of Computer Applications, Kongu Engineering College for giving her valuable guidance and suggestions which helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping overwhelm in all proceedings. We bow our heart with heartfelt thanks to all those who thought us their warm services to succeed and achieve our work.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	iv
	ACKNOWLEDGEMENT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	
	1.1 ABOUT THE PROJECT	1
	1.2 EXISTING SYSTEM	1
	1.3 DRAWBACKS OF EXISTING SYSTEM	1
	1.4 PROPOSED SYSTEM	3
	1.5 ADVANTAGES OF PROPOSED SYSTEM	4
2	SYSTEM ANALYSIS	
	2.1 IDENTIFICATION OF NEED	6
	2.2 FEASIBILITY STUDY	6
	2.2.1 Economic Feasibility	8
	2.2.2 Technical Feasibility	9
	2.2.3 Operation Feasibility	10
	2.3 SOFTWARE REQUIREMENTS SPECIFICATION	
	2.3.1 Hardware Requirements	12
	2.3.2 Software Requirements	12
3	SYSTEM DESIGN	
	3.1 MODULE DESCRIPTION	14

	3.2 DATA FLOW DIAGRAM	20
	3.3 USECASE DIAGRAM	22
4	IMPLEMNTATION	
	4.1 CODE DESCRIPTION	25
	4.2 STANDARDIZATION OF THE CODING	25
	4.3 ERROR HANDLING	26
5	TESTING AND RESULTS	
	5.1 TESTING	27
	5.2 UNIT TESTING	27
	5.3 INTEGRATION TESTING	28
	5.4 VALIDATION TESTING	28
6	CONCLUSION AND FUTURE ENHANCEMENT	
	6.1 CONCLUSION	29
	6.2 FUTURE ENHANCEMENT	29
	APPENDICES	
	A. SAMPLE CODING	31
	B. SCREENSHOTS	36
	REFERENCES	38

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.1	Level 0	20
3.2	Level 1	21
3.3	Use Case Diagram	22
B.1	Home Page	36
B.2	Admin Dashboard	36
B.3	User Dashboard	37

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

In today's fast-paced educational environment, students often encounter various challenges and issues during their time in college or university. These issues can range from academic concerns to administrative problems and can have a significant impact on the student's overall educational experience. In order to address these issues effectively, a comprehensive and effective student grievance redressal system is needed. A student grievance redressal system is a platform that provides students with a means of expressing their concerns and complaints regarding their educational experience. The primary objective of this system is to ensure that all student complaints are heard, addressed, and resolved in a timely and effective manner. The system is designed to provide students with a user-friendly interface that makes it easy to register their complaints and track their progress.

The student grievance redressal system is developed using Django (python framework), one of the most widely used programming languages for developing web-based applications. python is a versatile and flexible language that makes it well-suited for developing complex systems like a student grievance redressal platform. The system is designed to be user-friendly, making it easy for students to navigate and access all the necessary information. One of the key features of the student grievance redressal system is the ability to track and monitor the progress of each complaint. This helps to ensure that all complaints are addressed in a timely and efficient manner, and that students are kept informed of the progress of their complaints. The system is also equipped with reporting and analysis tools to help university administrators understand the nature and frequency of complaints and make informed decisions to improve the educational experience for students.

1.2 EXISTING SYSTEM

The current system for addressing students' grievances is entirely pen and paper based. The following steps show us the general process a student would typically take to file a complaint or grievance:

1. The student needs to visit the appropriate department in their college that deals with their particular concern, for example: The Fees and Accounting Department deals with fee related concerns, Library Department deals with library membership related concerns or availability of books, the Examination Department deals with exam paper or exam timetable related concerns, so each has their own specific role and set of procedures to follow.
2. The next step for the student is to write an application stating their grievances and submit it to the office of the departmental admin of the respective department.
3. After submitting a grievance, there is a delay period where the admin investigates the problem and devises an appropriate course of action.
4. Then the student is required to visit the department again for a response to their grievances.

1.3 DRAWBACKS OF EXISTING SYSTEM

Following are some disadvantages of the old system:

- Improper recording of complaints.
- Delays in processing and resolving grievances.
- Difficulties in Implementing Updates.
- Limited Accessibility Outside Office Hours.
- Difficulty in tracking.
- Risk of Misplacement or Loss.

1.4 PROPOSED SYSTEM

To overcome the drawbacks in the existing system, the proposed system is designed. The proposed system is fully website based. By this proposed system user can arise their grievances via the web-based application and track the status of the grievances. It reduces the manual paper work and it provides a platform for students to submit their complaints

about academic, personal or administrative issues that have an impact on the learning environment. It encourages students to raise concerns without fear and reprisal.

1.5 ADVANTAGES OF PROPOSED SYSTEM

- Simple and quite easy to use.
- Improved student's satisfaction.
- Flexibility for updates.
- Enforce standardized processes.
- Positive Institutional image.

CHAPTER 2

SYSTEM ANALYSIS

2.1 IDENTIFICATION OF NEED

In olden days, the student in the college want to go to the particular head authority if they have any query or grievances. They have to spend time to write a letter and wait for the solution. Hence, Web application for student's grievance redressal system was created. Using this website, the student can easily register their complaints through online and they can able to check their status of the complaints or suggestions as they registered.

2.2 FEASIBILITY STUDY

The feasibility study deals with all analysis that takes up in developing the project. Each structure has to be thought of in the developing of the project, as it has to serve the end user in friendly manner. One must know the type of information to be gathered and the system analysis consists of collecting, organizing and evaluating facts about a system and its environment. Three considerations involved in feasibility are

- Economic Feasibility
- Operational Feasibility
- Technical Feasibility

2.2.1 Economic Feasibility

The cost and benefit analysis may be conducted to know whether, the computerized system is favorable in today's fast-moving world. This application is developed with one of the open-source languages and HTML, CSS for building user interface or for frontend development. The economic feasibility is carried out to check the economic impact that the system will have an organization. In the proposed system many students can make use of it by register the complaints. The students can access the application through online

2.2.2 Operational Feasibility

The proposed device having access to system to solve issues what happened in existing device. The current day-to-day operations of the organization can be fit into the system. Mainly operational feasibility should include on analysis of how the proposed system will affect the organizational structure and procedures. The developed website is user-friendly to the students because they can easily access the application and register the complaints. The purpose of the operational feasibility is to determine whether the new system will be used if it is developed and implemented, also include the level of acceptance of the system by the users.

2.2.3 Technical Feasibility

Technical feasibility evaluates the technical complexity of the expert system and often involves determining whether the expert system can be implemented with state-of-the-art techniques and tools. It is an assessment of whether a proposed project, product, or service can be successfully implemented using current or available technology. The web application is developed using HTML, CSS with the help of visual studio code to implement, which have several developing tools for developing the web application. This application allows any number of users to register complaints, extending the application's usage to greater number of users. Thus, the students can easily access the web application and it is user-friendly.

2.3 SOFTWARE REQUIREMENTS SPECIFICATION

System requirements is a statement that identifies the functionality that is needed by a system in order to satisfy the customer's requirements. System requirements are the most efficient way to address user needs while lowering implementation costs.

- Hardware Requirements
- Software Requirements

2.3.1 Hardware Requirements

The hardware for the system is selected considering the factors such as CPU processing speed, memory access, peripheral channel access speed, printed speed; seek time & relational data of hard disk and communication speed etc. Below is the minimum hardware requirement of the project.

System	: I5 Processor
Hard disk	: 1 TB
RAM	: 8 GB
Monitor	: 15 VGA color

2.3.2 Software Requirements

The software for the project is selected considering the factors such as working front end environment, flexibility in the coding language, database knowledge of enhances in backend technology etc.

Frontend	: HTML, CSS, with Django.
Backend	: Django (python framework).
Database	: Postgresql
Tool	: Visual Studio code

FRONT END

- HTML is the standard markup language used to create the structure and content of web pages. It consists of a series of elements that define different parts of a webpage, such as headings, paragraphs, links, images, and more. HTML provides the basic structure for web content.
- CSS is used for styling the appearance of HTML elements on a webpage. It allows you to define colors, layouts, fonts, and other visual aspects of a webpage. With CSS, you can control the presentation and layout of multiple web pages at once.

BACKEND AND DATABASE

- Django is a Python-based web application framework that is free and open source. A framework is simply a collection of modules that facilitate development. They're grouped together and allow you to build apps or websites from scratch rather than starting from scratch.
- PostgreSQL is a powerful, open-source relational database management system (RDBMS) known for its robustness, extensibility, and compliance with SQL standards. Here's an overview of its key features and characteristics:

Some of the key features of PostgreSQL include:

1. **Relational Database:** PostgreSQL follows the relational model and stores data in tables with rows and columns, enabling efficient data storage and retrieval.
2. **ACID Compliance:** It ensures ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring data consistency and reliability even in the event of system failures.
3. **Advanced SQL Support:** PostgreSQL supports a wide range of SQL features, including complex queries, subqueries, triggers, views, stored procedures, and window functions, allowing for sophisticated data manipulation.
4. **Data Types:** It offers a rich set of data types, including numeric, string, date/time, JSON, arrays, GIS (Geographic Information System), and user-defined types, providing flexibility in data storage.
5. **Extensibility:** PostgreSQL can be extended through custom functions, data types, and procedural languages like PL/pgSQL, PL/Python, PL/Perl, and more. This extensibility allows users to tailor the database to specific needs.
6. **Concurrency Control:** Utilizes MVCC (Multi-Version Concurrency Control) to manage simultaneous transactions, ensuring efficient handling of concurrent database access without blocking or compromising data integrity.

CHAPTER 3

SYSTEM DESIGN

3.1 MODULE DESCRIPTION

A module description offers comprehensive information that is accessible in a variety of ways about the module and the components it supports. This application has two primary parts and numerous supporting modules.

The project contains following modules:

- Register
- User (students/faculty)
- Complaints
- Solved complaints
- Unsolved complaints

Register

This is the most basic and the vital module where the new users get their chance to create their account or sign-in in this web application. Proper validations will be provided to keep only authenticated users i.e. those users who will provide correct information. All the data supplied by the users will be stored in the database and it will be used for further validation and authentication. During registration, user has to give login and password of their choice.

User (students/faculty)

The user may be student/faculty can register by providing their user name and password. Users will be able to login into the system using their user name and password. In this the user has to give out the login details i.e. user id and password and then only he/she can be logged on. In this module students must login to make complaints and faculties must login to check the grievance and reply for the grievance. The user id and password given by

the users are checked from the data stored in the database. The details of the user are stored in the PostgreSQL. User have some login constraints.

complaints

Complaints module refers to the structured form with fields for students to detail their complaints including description, category (academic, administrative, harassment, etc.), and any supporting documents or evidence they might have. Students should be able to check the status of their complaints - whether it's pending, being reviewed, or resolved. Automated notifications or emails to inform students about the progress or resolution of their complaints. The result of the complaints are easily addressed by the students by email. Complaints can be assigned to specific administrators or departments responsible for handling different types of grievances.

Solved complaints

The module consists of the complaints that are addressed and solved by the admin of the particular concern. Administrators can update the status of a complaint once the issue has been addressed and solved satisfactorily. Store all details related to the resolved complaints, including the original complaint, actions taken, resolution details, and feedback received. Ensure that records of resolved complaints are easily viewable for reference by both students and administrators. Maintain transparency in the resolution process by keeping students informed about the progress and final resolution of their complaints.

Unsolved complaints

It consists of the unsolved complaints that yet to be solved. complaints remain unresolved due to poor communication between the students and the grievance redressal authorities. This might include delays in responses, unclear communication channels, or inadequate feedback on the status of the complaint. Some grievances might involve complex issues that are challenging to resolve swiftly. These could involve academic matters, discrimination, harassment, or administrative problems that require in-depth investigation and careful handling. The absence of clear policies or procedures for handling specific types of complaints can lead to unresolved issues. Addressing unsolved complaints within a student grievance redressal system often requires a comprehensive approach.

3.2 DATA FLOW DIAGRAM

The Data Flow Diagram provides information about the inputs and outputs of each entity and process itself.

LEVEL 0

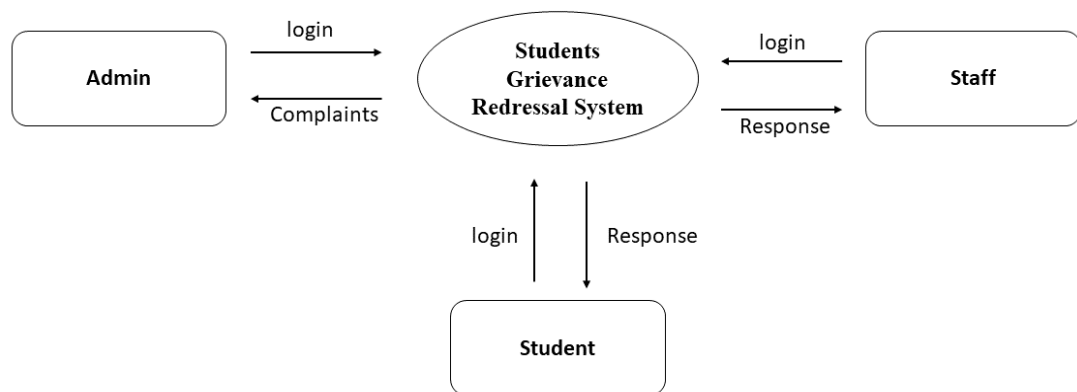
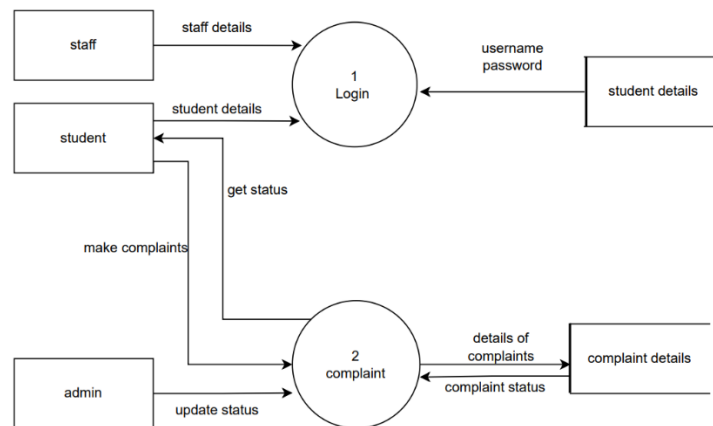


Figure 3.1 Level 0

LEVEL 1**Figure 3.2 Level 1**

3.3 DATABASE DESIGN

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. Database management system manages the data accordingly. The term database design can be used to describe many different parts of the design of an overall database system.

Principally, and most correctly, it can be thought of as the logical design of the base data structure used to store the data. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structure, but also the forms and queries used as part of the overall database application within the database management system.

Database Integration

Database integration for the Student Grievance Redressal System involves the seamless incorporation of multiple databases to enhance efficiency and data management. By integrating diverse databases, such as those containing student information, grievance records, and administrative details, the system achieves a unified and comprehensive data structure. This integration ensures quick access to relevant information, streamlining the grievance resolution process. It enables administrators to efficiently handle and analyze grievances, track student records, and implement timely resolutions. Overall, database integration optimizes the functionality of the grievance redressal system, fostering a cohesive and well-organized approach to managing student concerns within the academic institution.

Data Independence

Data independence is a critical feature of the Student Grievance Redressal System, ensuring adaptability and longevity. In this context, data independence means that modifications to the database, whether in its physical storage or logical structure, can be made without affecting the system's core functionality. Physical data independence allows for changes in the storage infrastructure without impacting the user experience, while logical data independence facilitates alterations to the database schema without disrupting

the grievance submission and tracking processes. This capability ensures that the system can evolve to meet new requirements, technological advancements, or improved data management practices without compromising its reliability and effectiveness.

Data Security

Data security is a paramount consideration in the Student Grievance Redressal System to safeguard sensitive information and ensure the privacy of student data. Robust encryption methods are employed to protect personal details, grievance records, and administrative information stored in the database. Access controls and authentication mechanisms are implemented to restrict unauthorized entry to the system. Regular security audits and monitoring are conducted to identify and address potential vulnerabilities promptly. The use of secure communication protocols further fortifies data integrity during information exchange. By prioritizing data security, the system ensures that students' grievances are handled confidentially, instilling trust in the overall process and fostering a secure environment for managing sensitive academic concerns.

3.4 ER DIAGRAM

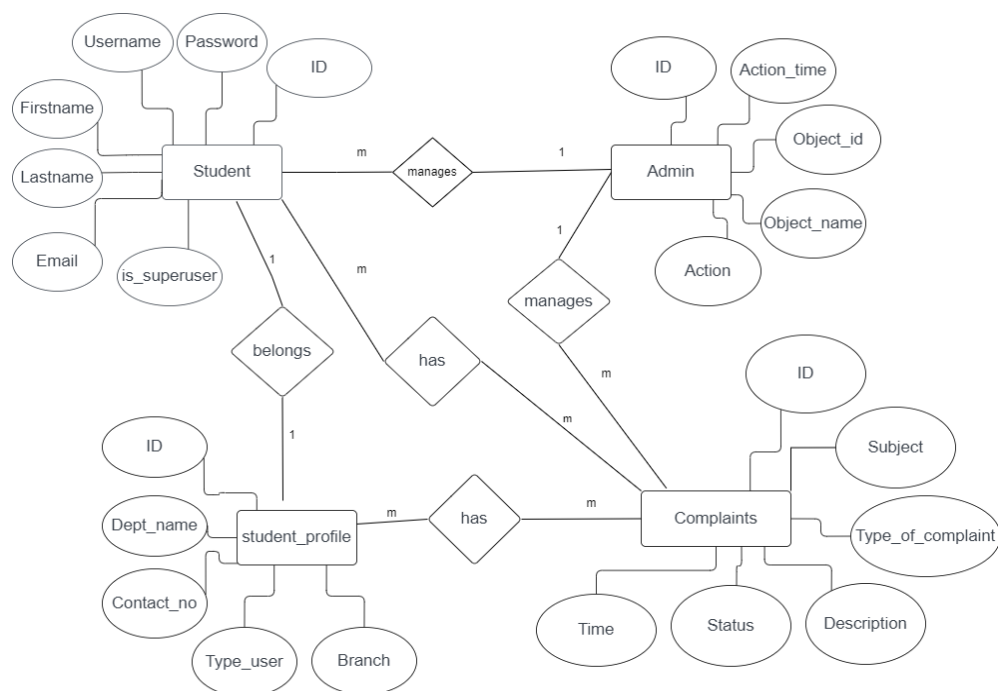


Figure 3.3 ER Diagram

3.5 TABLE DESIGN

Table 3.1 Complaints

S.no	Field name	Type	size	Constraints
1	Id	Int	10	Primary key
2	Subject	Varchar	200	Not null
3	Type_of_complaint	Varchar	200	Not null
4	Description	Varchar	250	Not null
5	Time	Datetime		Not null
6	Status	Int	5	Not null

The relation complaint contains six attribute fields namely Id, subject, type of complaint, description, time, status. Id is of an integer type and is of size 10 with primary key constraint. The size of the id is 10 as the id number will not exceed more than 10, and there should not be any duplications so it is with primary key.

Subject attribute and type of complaint attribute is a varchar type with max size of 200 as we can choose the subject of the grievance we had and also with not null constraint because it cannot be left empty. Description attribute is a varchar type with max size of 250 as we can write complete description of the complaint as we wish and also with not null constraint because it cannot be left empty. Time attribute describes about when the complaint has been registered and it is of type datetime and also with not null constraint as it cannot be left empty.

Table 3.2 Student

S.No	Field name	Type	Size	Constraints
1	Id	Int	10	Primary key
2	Password	Varchar	50	Not null
3	Username	Varchar	250	Not null
4	First_name	Varchar	250	Not null
5	Last_name	Varchar	250	Not null
6	Email	Varchar	254	Not null
7	Is_superuser	Boolean		Not null

The relation student contains seven attributes such as Id, password, username, first_name, last_name, email, is_superuser. Id is of type int and of size 10 as it need not exceed than that and with primary key constraint to avoid duplication. Password is of type varchar and of size 50 as the password can contain only 50 characters and with some constraint as it cannot be empty. Username is of type varchar and of size 250 characters as it can contain maximum of charcters with not null constraint.

First_name is of type varchar and of size 250 characters as it can contain maximum of charcters with not null constraint. Last_name is of type varchar and of size 250 characters as it can contain maximum of charcters with not null constraint. Email is of type varchar and of size 254 characters as it can contain maximum of charcters with not null constraint. Is_superuser is of type boolean datatype with not null constraint.

Table 3.3 Student_Profile

S.No	Feid name	Type	Size	Constraints
1	Id	Int	10	Primary key
2	Department	Varchar	30	Not null
3	Contact_number	Int	10	Not null
4	Type_user	Varchar	20	Not null
5	Branch	Varchar	30	Not null

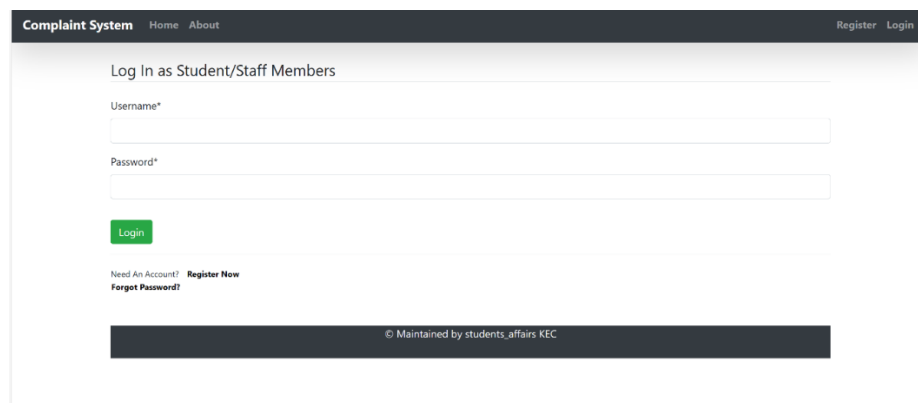
The relation student_profile contains five attributes such as id, department, contact_number, type_user, branch. Id is of type integer and of size 10 with constraint primary key as there cannot be duplicate values. Department and Branch is of type varchar and is of size 30 with not null constraint. Contact_number is of type int and is of size 10 with not null constraint. Type_user is of type varchar and is of size 30 as it can contain upto 30 characters and the field cannot not be empty.

3.6 INPUT DESIGN

Efficient input design is crucial for the Student Grievance Redressal System to facilitate a user-friendly and streamlined experience. The input forms should be intuitively designed, allowing students to easily submit their grievances with clear and concise fields for relevant information. Mandatory fields ensure essential details are captured, while dropdown menus and checkboxes help standardize input and reduce errors. Error-checking mechanisms are implemented to validate the accuracy of entered data, providing instant feedback to users. Capturing timestamps automatically ensures accurate record-keeping. The input design is aimed at simplifying the grievance submission process, enhancing data accuracy, and promoting a positive user experience within the system.

Login form

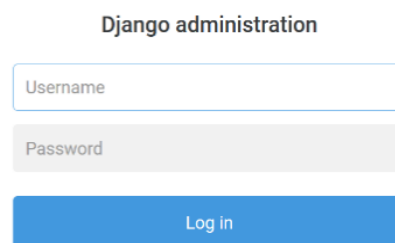
The login form for the Student Grievance Redressal System serves as a secure gateway for authorized access. It typically includes fields for entering a unique student ID or username along with a password. This ensures that only authenticated users can access the system. The form may also incorporate additional security measures such as CAPTCHA or two-factor authentication for an added layer of protection. Intuitive design and clear instructions contribute to a user-friendly experience. Through this login mechanism, students gain personalized access to submit and track their grievances, ensuring confidentiality and enhancing the overall security of the grievance redressal system.



The screenshot shows a web application titled "Complaint System" with a dark header. The header contains links for "Home" and "About" on the left, and "Register" and "Login" on the right. The main content area is titled "Log In as Student/Staff Members". It features two input fields: "Username*" and "Password*", both with asterisks indicating they are required. Below the password field is a green "Login" button. At the bottom of the form, there are links for "Need An Account? Register Now" and "Forgot Password?". A footer bar at the bottom of the page states "© Maintained by students_affairs KEC".

Figure 3.5 Students login

Figure 3.3 shows the student/ staff login page, where the students can login to students grievance redressal system using username and password.



The screenshot shows the Django administration login interface. It has a title "Django administration" at the top. Below the title are two input fields: "Username" and "Password". Below these fields is a blue "Log in" button.

Figure 3.6 Admin login

The admin can login with their username and password. After logging in, the responses to all the grievances received. Admin can also add and deleted the complaints.

3.7 OUTPUT DESIGN

The output design for the Student Grievance Redressal System focuses on presenting information in a clear and organized manner. Once students submit their grievances, the system generates well-structured outputs displaying the status, resolution updates, and any relevant feedback. Visual elements, such as progress indicators or color-coded notifications, enhance the user's understanding of their grievance's current status. Additionally, the system may provide downloadable reports or summaries for administrators, offering valuable insights into overall grievance trends and resolutions. A user-friendly and visually intuitive output design contributes to effective communication and transparency, ensuring that both students and administrators can easily comprehend and navigate the grievance resolution process.

	id [PK] integer	Subject character varying (200)	Type_of_complaint character varying (200)	Description text	Time date	status integer	user_id integer
1	2	classroom	ClassRoom	not clean	2023-11-16	3	3
2	5	classroom	ClassRoom	not clean	2023-11-26	2	8
3	6	exams reg..	Teacher	will be changed	2023-11-26	3	3
4	7	exams reg..	Teacher	updated	2023-11-26	1	3
5	8	grievance	ClassRoom	Not clean	2023-12-09	3	10
6	9	exams reg..	Teacher	Concept not understood	2023-12-09	3	10

Figure 3.7 Data storing in database

	is_superuser boolean	username character varying (150)	first_name character varying (150)	last_name character varying (150)	email character varying (254)	is_staff boolean	is_active boolean	date_joined timestamp w
1	true	subashini			subashini@gmail.com	true	true	2023-11-03 1
2	false	sneha	I	sneha	sneha@gmail.com	false	true	2023-11-16 1
3	false	suba	subashini	R	suba@gmail.com	false	true	2023-11-17 1
4	false	harini	harini	priyaah	harini12@gmail.com	false	true	2023-11-18 1
5	false	anu	anu	priya	anu@gmail.com	false	true	2023-11-26 1
6	false	kavi	kavi	priya	kavi@gmail.com	false	true	2023-11-26 1
7	false	suvi	suivitha	s	suvi07@gmail.com	false	true	2023-12-03 1
8	false	subars	suba	shini	subashinir05@gmail.com	false	true	2023-12-09 1

Figure 3.8 User's data in database

CHAPTER 4

IMPLEMENTATION

4.1 CODE DESCRIPTION

Code description can be used to summarize code or to explain the programmer's intent. Good comments don't repeat the code or explain it. They clarify its intent. Comments are sometimes processed in various ways to generate documentation external to the source code itself by document generator or used for integration with systems and other kinds of external programming tools. Developer have chosen HTML, CSS as front end and Django as backend.

4.2 STANDARDIZATION OF THE CODING

Coding standards define a programming style. A coding standard does not usually concern itself with wrong or right in a more abstract sense. It is simply a set of rules and guidelines for the formatting of source code. The other common type of coding standard is the one used in or between development teams. Professional code performs a job in such a way that is easy to maintain and debug. All the coding standards are followed while creating this project. Coding standards become easier, the earlier you start. It is better to do a neat job than cleaning up after all is done. Every coder will have a unique pattern than others to. Such a style might include the conventions he uses to name variables and functions and how he comments his work. When the said pattern and style is standardized, it pays off the effort well in the long.

4.3 ERROR HANDLING

Exception handling is a process or method used for handling the abnormal statements in the code and executing them. It also enables to handle the flow control of the code/program. For handling the code, various handlers are used that process the exception and execute the code. Mainly if-else block is used to handle errors using condition checking, if-else catch errors as a conditional statement. In many cases there are many corner cases

which must be checking during an execution but if-else can only handle the defined conditions. In if-else, conditions are manually generated based on the task.

An error is a serious problem than an application doesn't usually get pass without incident. Errors cause an application to crash, and ideally send an error message offering some suggestion to resolve the problem and return to a normal operating state, there no way to deal with errors live or in production the only solution is to detect them via error monitoring and bug tracking and dispatch a developer or two to sort out the code.

Exceptions, on the other hand are exceptional conditions an application should reasonably be accepted to handle. Programming language allow developers to include try. Catch statements to handle exceptions and apply a sequence of logic to deal with the situation instead of crashing. And when an application encounters an exception that there is no work around for, that is called an unhandled exception, which is the same thing as an error.

CHAPTER 5

TESTING AND RESULTS

5.1 TESTING

Once the source code has been finalized and documented, including associated statistical structures, the next phase involves testing and validation. During this stage, there is a dedicated focus on thoroughly testing the program, actively seeking errors and discrepancies.

The developer approaches testing with caution, conducting detailed design and execution tests to showcase the program's functionality and identify any potential issues. While the aim is to demonstrate the program's effectiveness, it is acknowledged that errors may still be present. If the developer fails to uncover these errors, it is likely that end-users will discover them.

Responsibility for testing extends to individual program components or modules, with the developer consistently conducting tests to ensure their proper functioning. Additionally, in many cases, the developer undertakes integration testing, a crucial step in validating the overall application structure.

5.1.1 Unit Testing

In the context of unit testing, the examination focuses on individual components that constitute the system. Unit testing is sometimes referred to as program testing because it involves evaluating the software within a unit, which comprises modules and routines assembled to perform specific functions. Unit testing is primarily directed at assessing the modules independently of each other, aiming to identify errors.

This approach is instrumental in detecting coding and logic errors confined within the module itself. The testing process occurs during the programming stage, allowing for early identification and resolution of issues.

Example Test Case 1

Module: User Login

Test Type: Loading of appropriate form for administrator

Input: Username and Password

Output: Display user pages

Sample Test Output: Redirect to the main page and display the dashboard pages.

Analysis: In this scenario, the test ensures that the username and password are in the correct format. If there is a mismatch between the entered credentials and the data stored in the database, it may lead to a data mismatch error. This highlights the importance of thorough testing to catch such issues early in the development process.

5.1.2 Integration Testing

Integration testing is conducted to verify whether individual modules seamlessly operate as a cohesive unit. During integration testing, the specific modules earmarked for integration are subjected to examination. In this phase, manual test data previously employed to assess interfaces is replaced with data automatically generated from diverse modules.

This process serves the purpose of assessing how a module would interact within the proposed system. The integration and subsequent testing of modules aim to expose any issues related to interfaces within the system. The focus is on identifying and addressing potential problems that may arise when modules interact with each other.

5.1.3 Validation Testing

Verification and validation testing are essential assessments conducted prior to delivering the product to the customer. This ensures the early commencement of the software testing life cycle. The primary objective of both verification and validation is to confirm that the product aligns with the customer's requirements and effectively fulfils its intended purpose.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

In conclusion, our Student Grievance Redressal System is designed to provide an effective and transparent platform for students to address and resolve their concerns. By offering a user-friendly interface, the system empowers students to freely and conveniently submit their grievances. The application serves as a bridge between students and the administration, aiming to eliminate communication barriers and ensure a seamless grievance resolution process.

Through this system, students can expect timely notifications and updates on the status of their complaints. Overall, our Student Grievance Redressal System is committed to enhancing transparency, accessibility, and efficiency in handling student grievances, thereby contributing to a more harmonious and supportive academic environment.

6.2 FUTURE ENHANCEMENT

The future enhancements for the Student Grievance Redressal System include the integration of a mobile application for on-the-go accessibility, multilingual support for inclusivity, and an anonymous grievance submission option to encourage transparency. The incorporation of data analytics tools enables administrators to make evidence-based decisions, while an automated escalation mechanism, integration with student portals, and a robust feedback system enhance efficiency and user satisfaction.

Exploring artificial intelligence for grievance triage, implementing real-time communication features, and fostering collaborative resolution aim to modernize and optimize the overall grievance redressal process. These collective improvements are designed to create a more responsive, user-friendly, and inclusive platform for addressing student concerns in academic institutions.

APPENDICES

A. SAMPLE CODING

Form.py

```
from django import forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm

from django.forms.widgets import DateInput

from django.shortcuts import render, redirect

from .models import Profile, Complaint

from crispy_forms.helper import FormHelper

from crispy_forms.layout import Submit

import requests

class ComplaintForm(forms.ModelForm):

    class Meta:

        model=Complaint

        fields=('Subject','Type_of_complaint','Description')

class UserProfileForm(forms.ModelForm):

    class Meta:

        model=Profile

        fields=('collegename','contactnumber','Branch')

class UserRegisterForm(UserCreationForm):

    first_name = forms.CharField(max_length=30, required=True)

    last_name = forms.CharField(max_length=30, required=True)
```



```

class Meta:

    model = User

    fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2', )

def clean_email(self):

    email = self.cleaned_data.get('email')

    # Check to see if any users already exist with this email as a username.

    try:

        match = User.objects.get(email=email)

    except User.DoesNotExist:

        # Unable to find a user, this is fine

        return email

    # A user was found with this as a username, raise an error.

    raise forms.ValidationError('This email address is already in use.')

'''class ProfileForm(forms.ModelForm):

    class Meta:

        model = Profile

        fields = ('bio', 'location',)'''

class ProfileUpdateForm(forms.ModelForm):

    email =forms.EmailField(widget = forms.TextInput(attrs={'readonly':'readonly'}))

    first_name=forms.CharField( max_length=30, required=True)

    last_name=forms.CharField( max_length=30, required=True)

    class Meta:

        model = User

        fields = ['username','email','first_name','last_name']

```

```

def clean_email(self):

    # Get the email

    username = self.cleaned_data.get('email')

    # Check to see if any users already exist with this email as a username.

    try:

        match = User.objects.exclude(pk=self.instance.pk).get(username=username)

    except User.DoesNotExist:

        # Unable to find a user, this is fine

        return username

    # A user was found with this as a username, raise an error.

    raise forms.ValidationError('This email address is already in use.')

class UserProfileUpdateform(forms.ModelForm):

    collegename =forms.CharField(widget = forms.TextInput(attrs={'readonly':'readonly'}))

    Branch=forms.CharField(widget = forms.TextInput(attrs={'readonly':'readonly'}))

    class Meta:

        model=Profile

        fields=('collegename','contactnumber','Branch')

class statusupdate(forms.ModelForm):

    class Meta:

        model=Complaint

        fields=('status',)

        help_texts = {

            'status': None,

        }

```

Model.py

```

from django.db import models

from django import forms

from django.contrib.auth.models import User

from django.dispatch import receiver

from django.db.models.signals import post_save

from django.core.validators import RegexValidator

from datetime import datetime

class Meta:

    app_label = 'ComplaintMS'

class Profile(models.Model):

    typeuser = (('student','student'),('grievance', 'grievance'))

    COL=(('MCA','MCA'),('MBA','MBA'),('BE','BE'),('BSC','BSC')) #change college names

    user =models.OneToOneField(User, on_delete=models.CASCADE,primary_key=True)

    collegename=models.CharField(max_length=29,choices=COL,blank=False)

    phone_regex =RegexValidator(regex=r'^\d{10,10}$', message="Phone number must be
entered in the format:Up to 10 digits allowed.")

    contactnumber    =    models.CharField(validators=[phone_regex],    max_length=10,
blank=True)

    type_user=models.CharField(max_length=20,default='student',choices=typeuser)

    CB=(('Computer Application',"Computer
Application"),('InformationTechnology',"InformationTechnology"),('ComputerScience',"C
omputerScience"),('InformationSystem',"InformationSystem"),('SoftwareSystem',"Softwar
eSystem"),('Electronics and Communication',"Electronics and
Communication"),('Mechanical',"Mechanical"),('Civil',"civil"),('Automobile',"Automobile"
))

```

```

Branch=models.CharField(choices=CB, max_length=29,default='Computer application')

def __str__(self):

    return self.collegename

def __str__(self):

    return self.user.username

@receiver(post_save, sender=User)

def create_user_profile(sender, instance, created, **kwargs):

    if created:

        Profile.objects.create(user=instance)

"@receiver(post_save, sender=User)

def save_user_profile(sender, instance, **kwargs):

    instance.profile.save()

class Complaint(models.Model):

    STATUS =((1,'Solved'),(2, 'InProgress'),(3,'Pending'))

    TYPE=((('ClassRoom','ClassRoom'),('Teacher','Teacher'),('Management','Management'),
('College','College'),('Other','Other')))

    Subject=models.CharField(max_length=200,blank=False,null=True)

    user=models.ForeignKey(User, on_delete=models.CASCADE,default=None)

    Type_of_complaint=models.CharField(choices=TYPE,null=True,max_length=200)

    Description=models.TextField(max_length=4000,blank=False,null=True)

    Time = models.DateField(auto_now=True)

    status=models.IntegerField(choices=STATUS,default=3)

    def __init__(self, *args, **kwargs):

        super(Complaint, self).__init__(*args, **kwargs)

```

```

        self.__status = self.status

    def save(self, *args, **kwargs):

        if self.status and not self.__status:

            self.active_from = datetime.now()

            super(Complaint, self).save(*args, **kwargs)

    def __str__(self):

        return self.get_Type_of_complaint_display()

    def __str__(self):

        return str(self.user)

class Grievance(models.Model):

    guser=models.OneToOneField(User,on_delete=models.CASCADE,default=None)

    def __str__(self):

        return self.guser

```

Settings.py

"""Django settings for web project.

Generated by 'django-admin startproject' using Django 2.0.7.

For more information on this file, see

<https://docs.djangoproject.com/en/2.0/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/2.0/ref/settings/>

"""

import os

Build paths inside the project like this: os.path.join(BASE_DIR, ...)

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

```

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = '$!@%0u0@m&x85($hsavn*qrcg54)=4a+f8!n#sgs1ko-8q+&mt'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [

    'ComplaintMS',

    'ComplaintMS.apps.SuitConfig',

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'crispy_forms',

    'django.contrib.sites',

    'allauth', # new

    'allauth.account', # new

    'allauth.socialaccount',

    'reportlab',

    'phonenumbers_field',

```

```

'django_extensions',
]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT_URLCONF = 'web.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'APP_DIRS': True,

'OPTIONS': {

'context_processors': [

'django.template.context_processors.debug',

'django.template.context_processors.request',

'django.contrib.auth.context_processors.auth',

```

```

        'django.contrib.messages.context_processors.messages',

    ],

},

],

WSGI_APPLICATION = 'web.wsgi.application'

# Database

# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql_psycopg2',

        'NAME': 'complaintmsdjango',

        'USER': 'postgres',

        'PASSWORD': '123456',

        'HOST': '127.0.0.1',

        'PORT': '5432',

    }

}

# Password validation

# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

    },

```



```

{
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
},
{
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]

# Internationalization

# https://docs.djangoproject.com/en/2.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Kolkata'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.0/howto/static-files/

STATIC_URL = '/static/'

STATIC_ROOT = os.path.join(BASE_DIR, 'static')

CRISPY_TEMPLATE_PACK = 'bootstrap4'

LOGIN_REDIRECT_URL = '/login_redirect/'

LOGIN_URL = 'signin'

```

```

LOGOUT_URL='logout'

SESSION_EXPIRE_AT_BROWSER_CLOSE= True

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL_HOST='smtp.gmail.com'

EMAIL_PORT=587

EMAIL_HOST_USER = "# add email address here

EMAIL_HOST_PASSWORD = " #email password

DEFAULT_FROM_EMAIL = "# add email address here

EMAIL_USE_TLS = True

#EMAIL_BACKEND = "django.core.mail.backends.filebased.EmailBackend"

#EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'

#EMAIL_FILE_PATH = os.path.join(BASE_DIR, "sent_emails")

AUTHENTICATION_BACKENDS = (

    # Needed to login by username in Django admin, regardless of `allauth`

    "django.contrib.auth.backends.ModelBackend",

    # `allauth` specific authentication methods, such as login by e-mail

    "allauth.account.auth_backends.AuthenticationBackend",

)

SITE_ID=3

ACCOUNT_EMAIL_REQUIRED = True

ACCOUNT_USERNAME_REQUIRED = False

ACCOUNT_SIGNUP_PASSWORD_ENTER_TWICE = False

ACCOUNT_AUTHENTICATION_METHOD = 'email'

ACCOUNT_UNIQUE_EMAIL = True

```

B. SCREENSHOTS

HOME PAGE



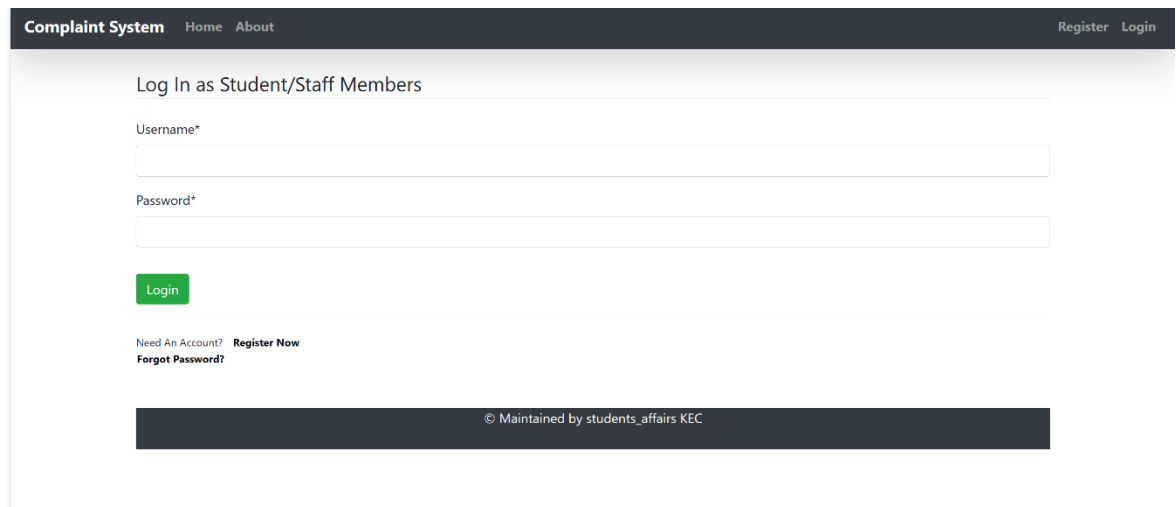
Figure B.1 Students grievance redressal system

Figure B.1 shows the home page of grievance redressal system. The students can register with their id, name, email and they can choose their respective department. Only the registered students can raise their grievances.

REGISTRATION PAGE

Figure B.2 Students registration page

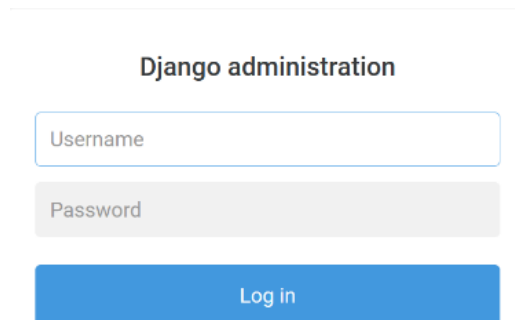
LOGIN PAGE



The screenshot shows a web application titled "Complaint System" in the top left corner of a dark header bar. To the right of the title are links for "Home" and "About". Further right in the same bar are links for "Register" and "Login". Below the header, the main content area has a light gray background. It starts with the heading "Log In as Student/Staff Members". Below this heading are two input fields: "Username*" and "Password*", each with a light gray border. A green "Login" button is positioned below the password field. At the bottom of the login section, there are two links: "Need An Account? Register Now" and "Forgot Password?". A dark gray footer bar at the very bottom contains the text "© Maintained by students_affairs KEC".

Figure B.3 Students login

Figure B.3 shows the student/ staff login page, where the students can login to students grievance redressal system using username and password.



The screenshot shows a simple login form for "Django administration". The title "Django administration" is centered at the top. Below it are two input fields: "Username" and "Password", both with light gray borders. A blue "Log in" button is centered below the password field.

Figure B.4 Admin login

The admin can login with their username and password. After logging in, the responses to all the grievances received. Admin can also add and deleted the complaints.

ADD COMPLAINTS

Add Complaints :

Subject*

This field is required.

Type of complaint*

This field is required.

Description*

This field is required.

Submit

Figure B.5 Adding grievances

In figure B.5, the students can add the complaint and can submit the complete description of the grievances.

Complaint System Home About Profile Logout

ComplaintMS Hide Sidebar

Welcome : subars

Profile

Password Reset

Add Complaints

UnSolved Complaints

Solved Complaints

UnSolved Complaints

ID	User	Subject	Complaint Type	Issued date	Desc	Status
1	subars	grievance	ClassRoom	Dec. 9, 2023	Details	

© Maintained by students_affairs KEC

Figure B.6 Added grievances

The figure B.6 depicts the addition of the complaints and details to the database. It displays that your complaints have been registered. It will be notified by the admin and the response will be updated in the status field.

DELETION OF RECORDS

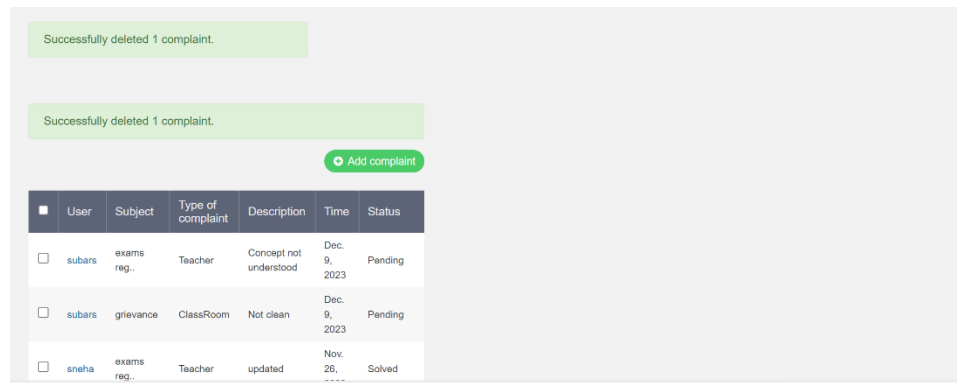


Figure B.7 Deletion of complaint

The figure 6.6 depicts that the record has been successfully deleted from the database and it displays message as such that it is successfully deleted.

COMPLAINT DETAILS IN POSTGRESQL

	id [PK] integer	Subject character varying (200)	Type_of_complaint character varying (200)	Description text	Time date	status integer	user_id integer
1	2	classroom	ClassRoom	not clean	2023-11-16	3	3
2	5	classroom	ClassRoom	not clean	2023-11-26	2	8
3	6	exams reg..	Teacher	will be changed	2023-11-26	3	3
4	7	exams reg..	Teacher	updated	2023-11-26	1	3
5	8	grievance	ClassRoom	Not clean	2023-12-09	3	10
6	9	exams reg..	Teacher	Concept not understood	2023-12-09	3	10

Figure B.8 Table view

The relation complaint contains six attribute fields namely Id, subject, type of complaint, description, time, status. Id is of an integer type and is of size 10 with primary key constraint. The size of the id is 10 as the id number will not exceed more than 10, and there should not be any duplications so it is with primary key.

Subject attribute and type of complaint attribute is a varchar type with max size of 200 as we can choose the subject of the grievance we had and also with not null constraint because it cannot be left empty. Description attribute is a varchar type with max size of 250 as we can write complete description of the complaint as we wish and also with not null constraint because it cannot be left empty. Time attribute describes about when the complaint has been registered and it is of type datetime and also with not null constraint as it cannot be left empty.

REFERENCES

- [1] <https://www.w3schools.com/django/>
- [2] <https://www.geeksforgeeks.org/django-tutorial/>
- [3] <https://www.geeksforgeeks.org/how-to-use-postgresql-database-in-django/>
- [4] <https://hevodata.com/learn/django-postgresql/>