

STUDENT GRIEVANCE SYSTEM



Submitted by:-

AVNI(20/49047),RUDRAKSH(20/49031),
HANSRAJ(20/49009)

Table of Contents

CHAPTER 1:INTRODUCTION

- 1.1 Problem statement
- 1.2 Limitations of current systems
- 1.3 Proposed solution
- 1.4 Advantages of proposed solution

CHAPTER 2:SRS

2.1 Introduction

- 2.1.1 Purpose
- 2.1.2 Document Conventions
- 2.1.3 Intended Audience and Reading Suggestions
- 2.1.4 Product Scope

2.2 Overall Description

- 2.2.1 Product functions
- 2.2.2 Product perspective
- 2.2.3 User classes and characteristics
- 2.2.4 Operating Environment
- 2.2.5 Design and Implementation Constraints

2.3 External Interface Requirements

- 2.3.1 User Interfaces
- 2.3.2 Hardware Interfaces
- 2.3.3 Software Interfaces

2.4 System Features

- 2.4.1 Registration

2.4.2 Raise Query

2.4.3 View query status

CHAPTER 3 : PROJECT MANAGEMENT

3.1 Gantt Chart

3.2 Risk Management and analysis

3.3 Functional Point Analysis

CHAPTER 4 : DESIGN IMPLEMENTATION

4.1 Interface Design (GUI)

4.2 Architectural Design

4.3 ER Diagram and Database Schema

CHAPTER 5 : TESTING

5.1 Test cases

5.2 Flow Graph

5.3 Cyclometric Complexity

CERTIFICATE

This is to certify that the project entitled,

“Student Grievance System”

Submitted by **“HANSRAJ SINWAR”**,

“Rudraksh” and **“Avni Sharma”** in the partial fulfillment of the requirements for the award in **“SOFTWARE ENGINEERING”** is an authentic work carried out by them under my supervision and guidance.

Certified by :

Mrs. Anita Goel

Mrs. Sheetal Taneja

Acknowledgement

It gives me great pleasure to express my gratitude towards our software engineering teacher **MRS. ANITA GOEL AND MRS. SHEETAL TANEJA** for their guidance, support and encouragement throughout the duration of the project. Without his motivation and help the successful completion of the project would not have been possible.

NAME : Hansraj Sinwar

Rudraksh

Avni Sharma

CHAPTER 1 : INTRODUCTION

1.1 Problem Statement:

This grievance system software will be developed for managing all the complaints related to the students' respective fields. Our aim is to digitize the Management system of the grievance system. It consists of overall management of student grievance system through software.

The purpose of this project is to provide optimised solutions for the student grievances. The proposed model for the student complaint management system will have ability to minimize students dissatisfaction we try to improve the relationship between student and university by presenting the model of e-complaint web based system This system will give solution to the students grievances. The existing system has manual processing through committee, principal and concerned departments . The proposed system had capable to complete the process automatically by using our application.

We implement an efficient and user friendly web application. Hence the application provides a solution through a simple interface which helps to overcome the time consuming process.

The utility and main objective of the project is to add automation to the process in an institution. This is an online based application so it can provide efficiency to acquire, store and process. Each individual student will be provided with their respective details. This web application involves five types of individuals, student, Committee, Principal, Institution and University. Each of them has to register with

their respective employee id and password out of which the password can be changed by the particular user.

1.2 Limitations of current system:

a) The existing system is completely manual. In order to write the complaint, the student either

- Visits the related department and registers his complaint in the respective complaint register, which is monitored by the respective Department heads.
- Existing system requires manual process (i.e sending grievance from lower level to critical level requires manual process.)

b) Students not able to find the complaints status.

c) After receiving a complaint through complaint form, Commission staff will review the submitted materials and contact the submitter for any required additional information or clarifications.

d) Commission staff will have to inform manually to students about the complaint.

e) This manual(physical) system having difficulty in saving the records.

f) The Complaint box would not be opened on Saturday, Sunday and any other college holidays.

g) The Committee shall not meet frequently to the decided on the basis of number of grievances registered.

1.3 Proposed solution/system:

- The idea is to automate the entire complaint process.
- Sending grievance from lower level to critical level is done automatically.
- Students can able to track the grievance once the complaint has been registered.
- It must be an easy access application, accessible to students, members of Student Grievance Redressal Committees, respective heads.
- Students should be able to post complaints under different categories, Department Level, Institute/College Level and University Level. Again these categories would be sub divided among sub categories such as Admission, Finance, Examination, Lecture Timetable/Learning, Paper Re-Evaluation, etc.
- Members of Students Grievance Redressal Committee should be able to sort complaints.
- The Portal should link students with respective Department/Institutions/College and University Students Grievance Redressal Committees.

1.4 Advantages of proposed solution:

1.4.1 Cumbersome in offline

Traditionally, In college if anyone wants to file a complaint one has to write on a paper and has to wait for the response. Using this System, Student and College Admin can come together to solve the issues related to the college.

1.4.2 Priority Based

Student can register complaint by specifying ***Department level, Institute Level or University level*** according to priority of complaint.

1.4.3 Higher efficiency

Students and teachers both can manage the grievances in ordered manner. Student-teacher interaction or student-student interaction through cell will result in better communication.

1.4.4 Complaint categories

Student can categorize the complaint as for ***Admission, Finance, Exams, Lecture, Timetable, Paper Re-evaluation, Others.***

1.4.5 BOT system

BOT interacts at the base level to solve queries that are frequently asked.

1.4.6 Complaint status

In- Process Complaints, Pending Complaints, Closed Complaints on either side of the User.

1.4.7 Transparency

Greater confidentiality and transparency in grievance dealing procedure.

1.4.8 Automates entire complaint process right from registration to closure.

1.5 Requirement Gathering from Users Involved through Google Form Creation

Google form link:

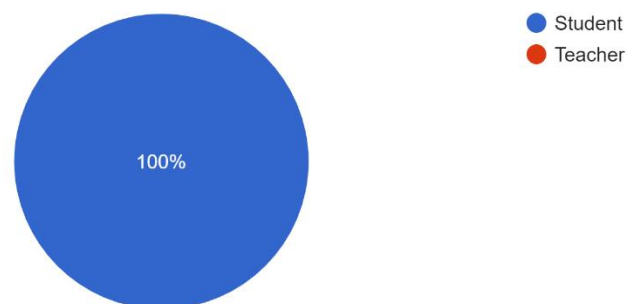
<https://forms.gle/yNrAx2A6cKNbZPWv7>

Requirements gathering from users sheet link:

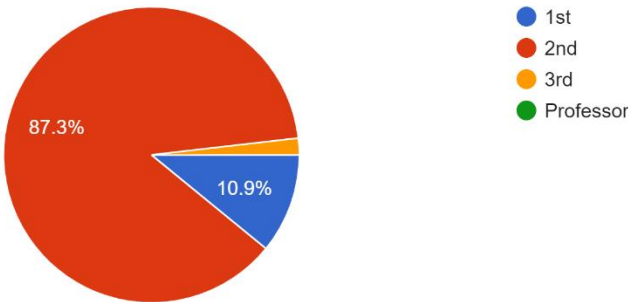
https://docs.google.com/spreadsheets/d/1u9-TBvDV9OgYalDXe1Nz4mF4jWVcYppmqTnm2gdeoHM/edit?usp=drive_sdk

Google form responses summary:

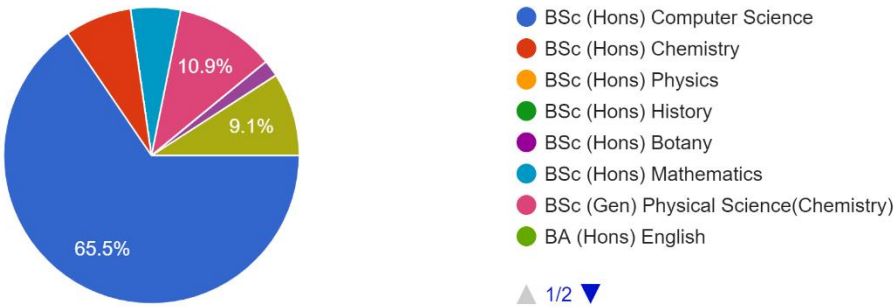
Are you a student or a teacher?
55 responses



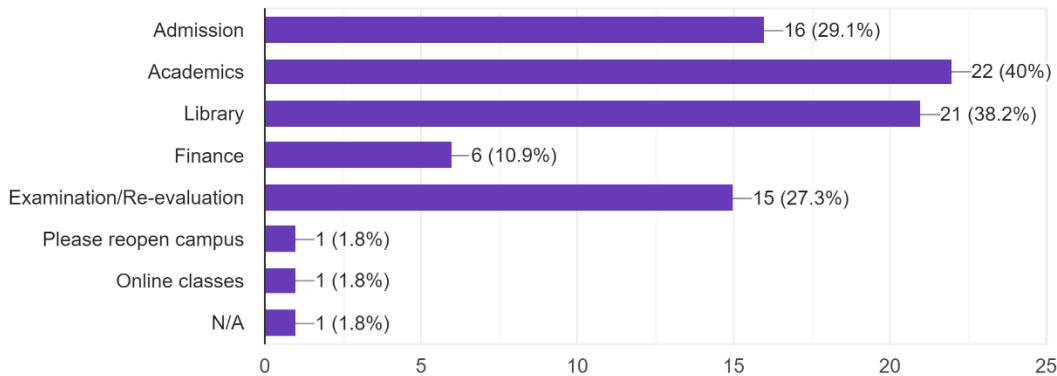
In which year are you?
55 responses



Choose department(if student)
55 responses

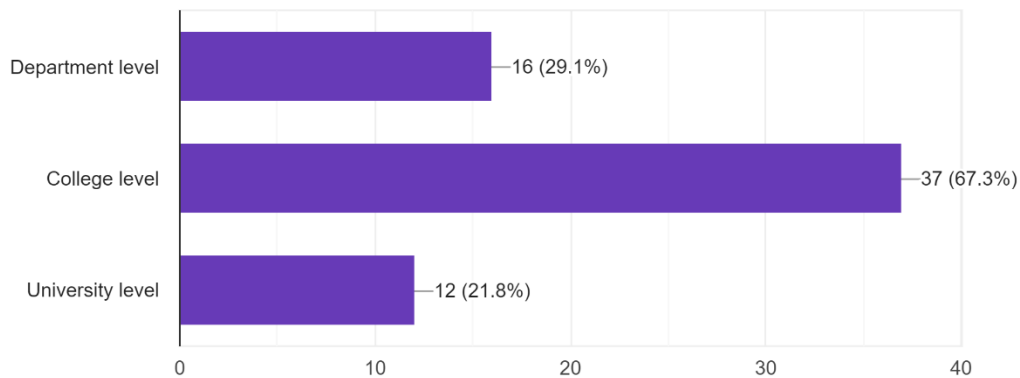


What type of grievance(s) have you faced/been approached for?
55 responses



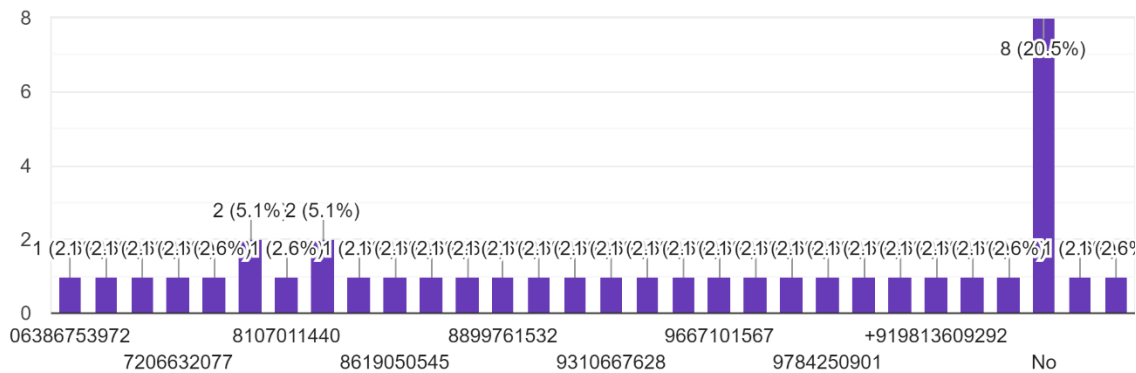
What was the level your query was raised to?

55 responses



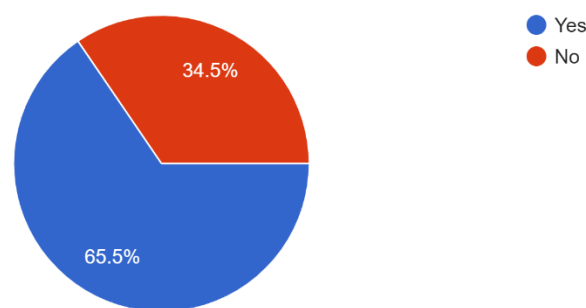
Are you willing to share your contact number(if yes then mention)?

39 responses



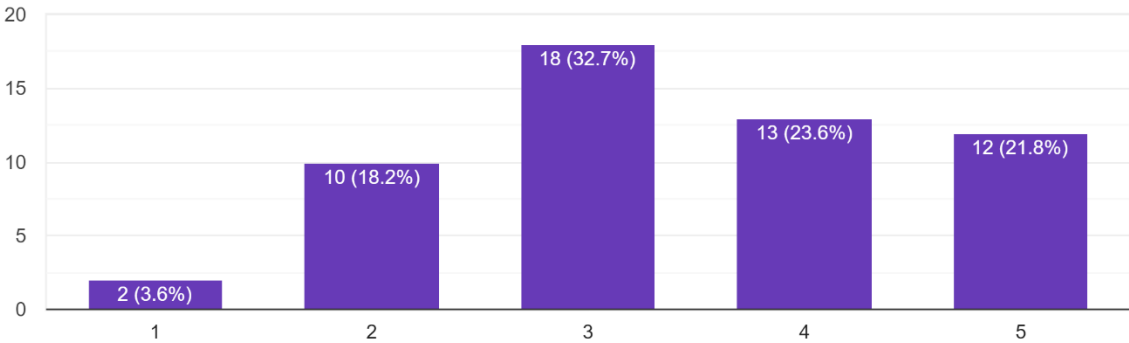
Would you like to become a part of student Grievance cell?

55 responses



How satisfied are you with current mode of grievance handling?

55 responses



CHAPTER 2 : SRS

2.1 Introduction :

The SRS is produced at the culmination of the analysis task. The function and performance allocated to software as part of the system engineering and refined establishing a complete information description, a detailed functional description, a representation of system behaviour, indication of performance requirements and design constraints, appropriate validation criteria and the information related to requirements. The SRS is a technical specification of requirements of the Student Grievance System.

2.1.1 Purpose:

The main purpose of our system is to make Student's Grievances easier and solve in minimum time limit. To establish the process for the management of Student Grievances that is a systematic and fair.

2.1.2 Document conventions:

This document follows MLA Format. Bold-faced text has been used to emphasize section and sub-section headings.

2.1.3 Intended audience and Reading suggestion:

The intended audience of the system are the users of the app. The users of the app will be the Students, Grievance cell members(may be student or teacher)and Admin of app.

2.1.4 Product scope:

In scope:

- Student can interact with bot (general queries).
- Student can raise query.
- Grievance cell members can resolve and manage queries.

- User can check history of queries.
- Admin can maintain and improve bot and manage database.

Out of scope:

- Student can't raise more than one query at a time.
- Management of members.

2.2 Overall Description:

2.2.1 Product Functions:

Use case 1: Sign-up

Brief: This use case describes how user can sign-up to their account.

Primary actor: App user

Pre-condition: Sign-up page is available

Post-condition: Redirected to login page

Success scenario:

- App shows the sign-up page
- User fill the required details (email, password, college roll no)
- User submit all the details
- System shows notification of successfully registered

Exception scenario:

- Invalid Email
- User already exists
- All information not filled correctly

Basic flow:

Sign-up page -> enter required information -> submit the details -> successfully registered

Alternate flow:

Sign-up page -> enter required information -> submit the details -> user already exists

Sign-up page -> enter required information -> submit the details -> provided information is incorrect

Use case 2: Log-in

Brief: This use case describes how user can log-in into the account.

Primary actor: App user

Pre-condition: Actor should be an existing user

Post-condition: Redirected to Home/dashboard

Success scenario:

- App shows the log-in page
- User fill the required details (email, password)
- User submit all the details
- System redirected to dashboard

Exception scenario:

- Invalid Email or password
- Email not exists

Basic flow:

Log-in page -> enter required information -> submit the details -> Dashboard

Alternate flow:

Log-in page -> enter required information -> submit the details ->

Log-in page -> enter required information -> submit the details -> email provided doesn't exist

Use case 3: Forget-password

Brief: This use case describes how user can reset their password

Primary actor: App user

Pre-condition: Actor should be an existing user

Post-condition: Redirected to login page

Success scenario:

- App shows the forget password page
- User fill the required details (email and college roll no)
- User submit all the details
- System send mail to reset password

Exception scenario:

- User not exists

Basic flow:

Log-in page -> click on forget password -> send E-mail -> option to reset

Alternate flow:

Log-in page -> click on forget password -> user not exists

Use case 4: Raise query

Brief: User can raise query according to their problems.

Primary actor: Students

Pre-condition: User must logged-in

-User should not have any query pending

Post-condition: Query raised

Success scenario:

- App shows the dashboard page
- User clicks on the raise query (button)
- User will fill the details about the query
- System send the query to the respective cell member.

Exception scenario:

- User not provided the whole details.

Basic flow:

Log-in -> raise query tab ->choose level of query ->choose type ->raise query -> query raised successfully.

Alternate flow:

Log-in ->raise query tab ->choose level of query ->choose type ->raise query -> Query doesn't satisfying the conditions.

Use case 5: Respond to query

Brief: This use case describes how respective cell member (or bot) will respond to the queries.

Primary actor: Grievance cell, bot

Pre-condition: Actor should be an existing member of grievance cell

Success scenario:

- Bot responds to query
- if not, passes it to grievance cell
- Cell member resolve the query
- Solution of query sent to the student

Exception scenario:

- The level doesn't able to resolve the query

Basic flow:

Bot responds to the query

Cell member logs in -> view pending queries -> choose a query -> give relevant response

Alternate flow:

Cell member logs in -> view pending queries -> choose a query -> forward query to next level

Use case 6: Check History

Brief: This use case helps user to see the past queries and their solutions.

Primary actor: Students, cell member

Pre-condition: User must logged-in

Post-condition: No query raised

Success scenario:

- App shows the history tab
- User can see the queries that he/she raise
- User click on any query

- Solution of that query displayed

Exception scenario:

- Not query raised yet

Basic flow:

Log-in -> history tab ->select query ->see resolution to query

Alternate flow:

Log-in ->history tab ->no query raised

Use case 7: Maintain and improve bot

Brief: This use case describe how user can update the student-bot interaction

Primary actor: Admin

Pre-condition: User should be an admin, logged-in

Post-condition: Data updated

Success scenario:

- App shows the dashboard page
- User clicks on relevant tab
- Changes done accordingly

Exception scenario:

- This change already done

Basic flow:

Log-in -> click relevant tab ->do the changes ->successfully done

Alternate flow:

Log-in ->click relevant tab ->do the changes ->already done

Use case 8: Receive resolution to the current query.

Brief: This use case describes the solution to the current query.

Primary actor: Students

Pre-condition: User must logged-in

Post-condition: Home page

Success scenario:

- App shows the dashboard page
- User clicks on the Current Query (button)
- See the answer of that query

Exception scenario:

- Solution yet not received

Basic flow:

Log-in -> current query tab ->resolution ->home-page

Alternate flow:

Log-in ->current query tab ->resolution yet not received

Use case 9: Check Status of query

Brief: This use case describes how user can able to see the status of current query.

Primary actor: Students

Pre-condition: User must logged-in

-User should have any pending query

Post-condition: Home-page

Success scenario:

- App shows the dashboard page
- User clicks the status of query tab
- Status (pending, completed)

Exception scenario:

- No query raised

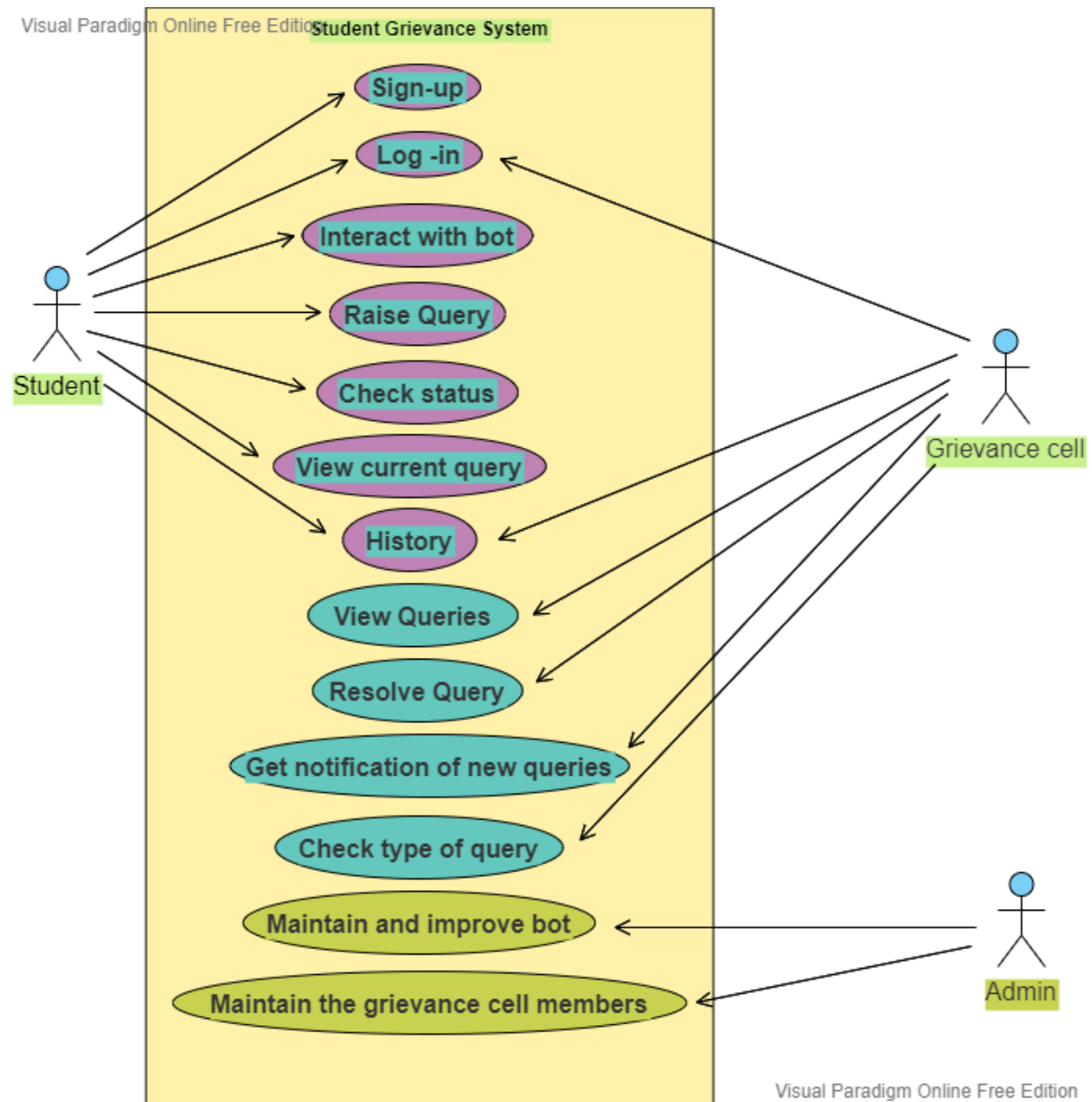
Basic flow:

Log-in -> status of query -> show status ->back to home

Alternate flow:

Log-in ->status of query ->no query raised yet.

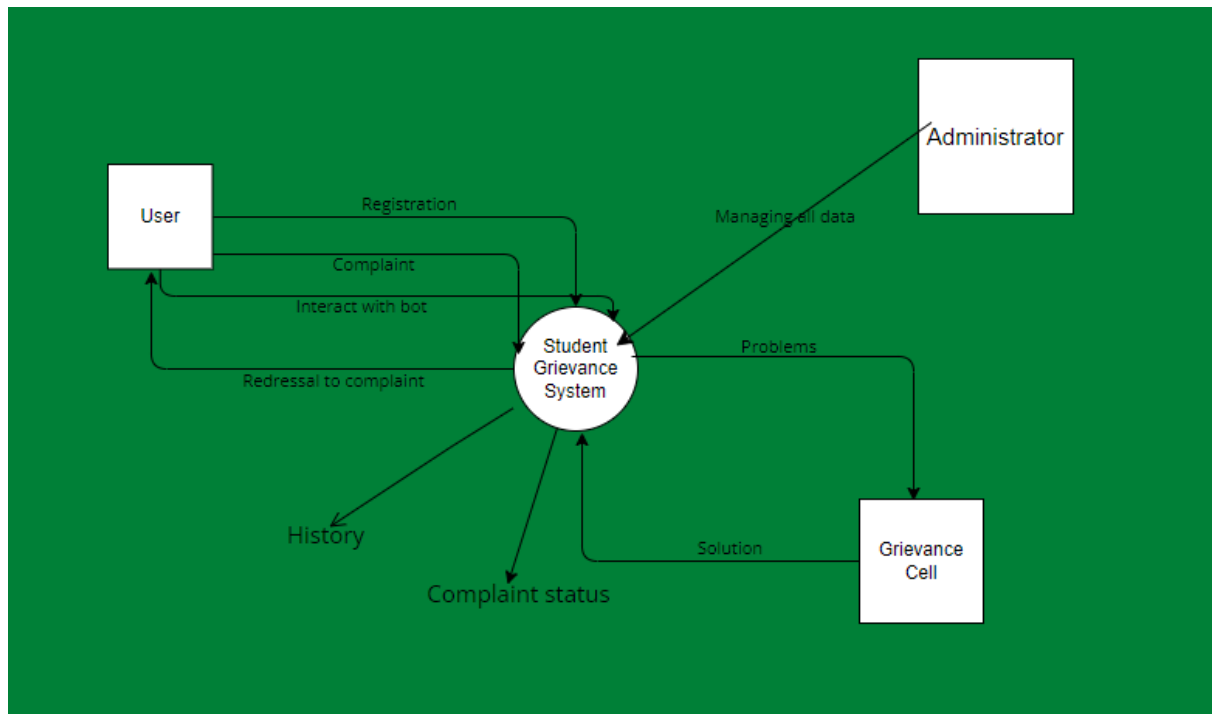
Use Case Diagram:



2.2.2 Product perspective:

Data flow diagrams

- 0-level DFD



- 1-level DFD

- a) User will be able to use the app to raise query
- b) View queries raised by user
- 2. Grievance cell members:
 - a) User will be able to resolve queries
 - b) User will view and manage queries
- 3. Admin:
 - a) User will be able to maintain the bot
 - b) User will be able to manage members

2.2.4 Operating Environment:

This app will operate on mobile devices for both android and ios. An internet connection is a must.

2.2.5 Design and Implementation Constraints:

- 1) Only English language is supported.
- 2) User must have android or ios devices.
- 3) User must have correct id and password to login.
- 4) Level of query depends on the type of query.

2.3 External Interface Requirements:

2.3.1 User Interfaces:

GUI

2.3.2 Hardware Interfaces:

- 1) Operating System-Android or IOS
 - Android version 7 and up
 - IOS version 8.0 and up
- 2) Storage-256 MB
- 3) RAM-512 MB

2.3.3 Software Interfaces:

- 1) Front end - HTML, CSS, Javascript
- 2) Back end – PHP

3) Database – SQL

2.4 System Features:

2.4.1 Registration

2.4.1.1 Functional requirements

- Internet connectivity is a must
- User must be associated to the college(student/professor)
- User must have signed up if logging in to the system

2.4.2 Raise Query

2.4.2.1 Functional requirements

- Internet connectivity is a must
- No previous query should be pending

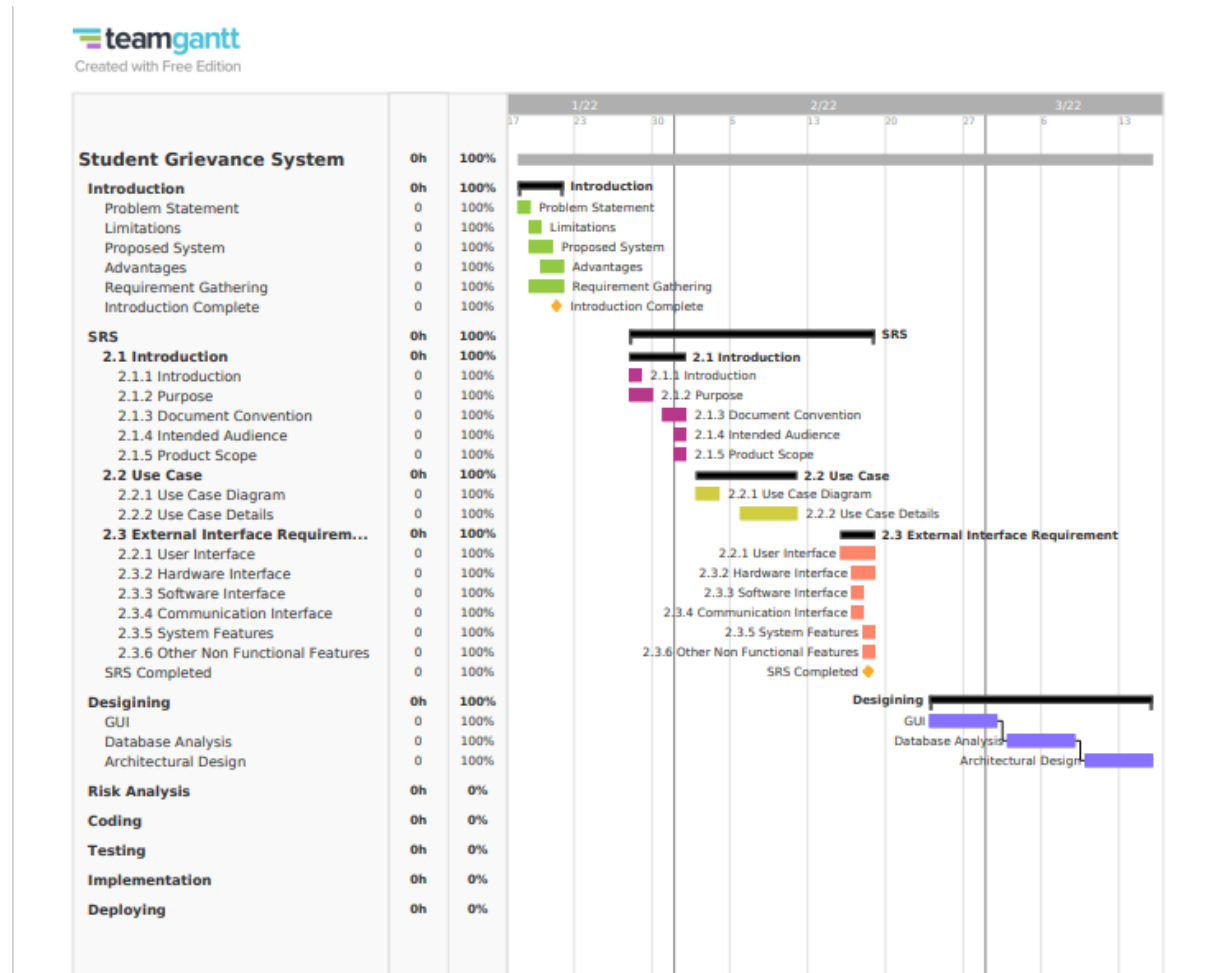
2.4.3 View query status/solution

2.4.3.1 Functional requirements

- Internet connectivity is a must.

CHAPTER 3 : PROJECT MANAGEMENT

3.1 Gantt Chart :



3.2 Risk Management and analysis :

RISK	CATEGORY	PROBABILTY	IMPACT	RMMM
OTP not sent	TR	5%	3	Resend <u>otp</u> after 30 sec.
Too many users	TR	5%	2	Activate backup users
Data corruption	TR	5%	1	Backup database is loaded
Registration failed	PR	2%	2	Re-register
Security break	TR	4%	3	Proper step authentication takes place
Wrong password by bot	PR	10%	2	Bot constantly tested and updated
Unable to raise query	PR	2%	1	Try login again

3.3 Functional point analysis :

EIF : 1. Cell member refer (Professors / Student from college database)

EF : 1. Student log in

2. Student registration
3. Cell member login
4. Cell member registration
5. Admin manages users
6. Admin manages bot
7. Resolve query

EQ : 1. Status / History

2. Raise Query – Get Solution
3. Bot

EO : 1. No of pending queries (total) (processing)
2. How many queries resolved

ILF : 1. Databases

Ranking :

EI ->FTR =2 DET=6-5 AVG

EO->LOW

EQ->LOW

FTR->{Files uploaded / references}

DET->{User recognisable fields (attributes)}

RET->{Tables in files}

ILF = avg

ELF = avg (DET = 6-15 , RFT=2-5)

	LOW	AVG	HIGH
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

EI EO EQ ILF ELF

UFP = $7*4 + 2*4 + 3*3 + 20 + 7 = 72$

CAF= $(0.65 + 0.01*14*3)$

$$= 0.65 + 0.42$$

$$= 1.07$$

$$\text{FUNCTION POINT} = \text{URP} * \text{CAF}$$

$$= 72 * 1.07$$

$$= 77.04$$

CHAPTER 4 : DESIGN IMPLEMENTATION

4.1 Interface Design (GUI) :

STUDENT GRIEVANCE CELL

LOGIN

REGISTER

STUDENT GRIEVANCE CELL

REGISTER

enter email

enter phone number

enter name

enter password

confirm password

☐ register as student ☐ register as member


STUDENT GRIEVANCE CELL

LOGIN


email/mobile number

password

REGISTER

 [HOME](#) [HISTORY](#) [PROFILE](#) [STATUS](#) [RAISE QUERY](#) [LOGOUT](#)

QUERY ID	DATE	TYPE	TITLE	STATUS
10234	22-03-21	DEPT	FEES	PENDING
2345	21-01-19	UNI	ID CARD	RESOLVED
12432	19-05-18	DEPT	LIBRARY	RESOLVED

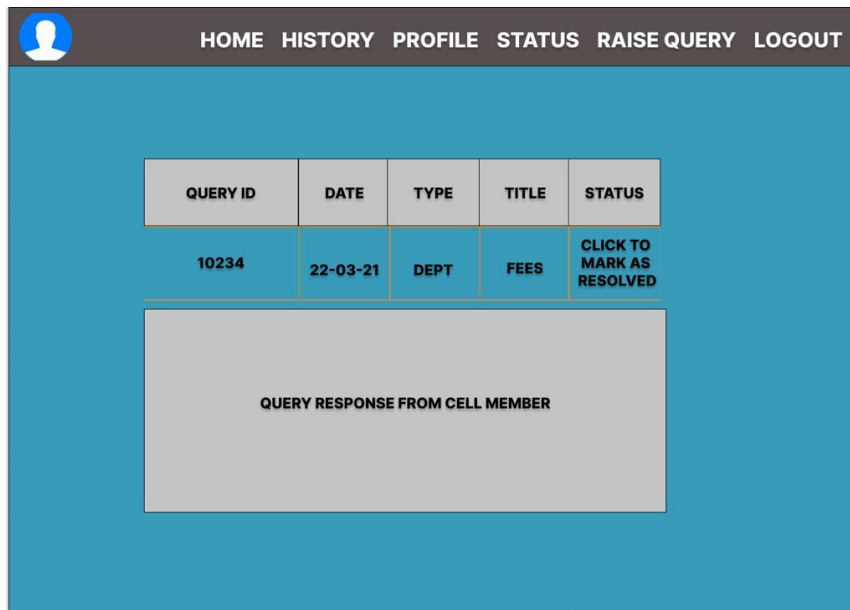
 [HOME](#) [HISTORY](#) [PROFILE](#) [STATUS](#) [RAISE QUERY](#) [LOGOUT](#)

ENTER QUERY TYPE <select from choice>

ENTER QUERY SUBJECT <select from choice>

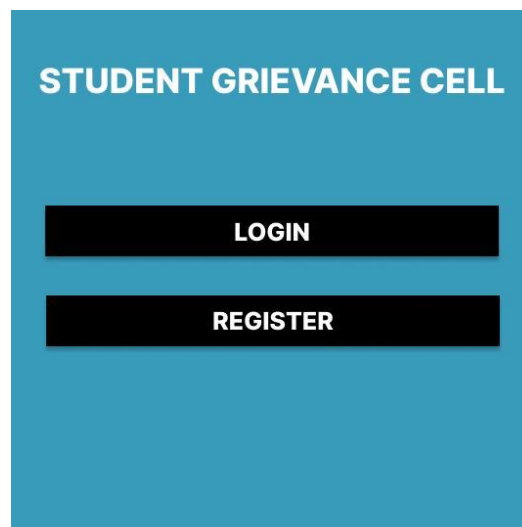
ENTER ADDITIONAL QUERY INFORMATION

RAISE QUERY



4.2 Architectural Design :

1st page:



```
function index(){
    if(Button=="login"){
        go to login page
    }
    if(Button=="register"){
        go to register page
    }
}
```

}

2nd page:

STUDENT GRIEVANCE CELL

REGISTER

enter email

enter phone number

enter name

enter password

confirm password

☐ register as student ☐ register as member

```
Function register(USER,NAME,PASSWORD,REGISTER_AS){
```

```
    if(Email/ph_no is valid){
```

```
        USER=Entered name
```

```
    }
```

```
    If(NAME){
```

```
        NAME=Entered name
```

```
    }
```

```
    If(PASSWORD){
```

```
        PASSWORD=Entered password
```

```
    }
```

```
    If(Confirm Password==PASSWORD)
```

```
    {
```

```
        Return true
```

```
}  
If(Registered as){  
    If(Registered as=="STUDENT"){  
        REGISTER_AS="STUDENT"  
    }  
    If(Registered as=="CELL MEMBER"){  
        REGISTER_AS="CELL MEMBER"  
    }  
}  
  
If(BUTTON=="REGISTER"){  
    If(USER&&NAME&&PASSWORD&&REGISTER_AS{  
        Store data into database(USER TABLE)  
    }  
}  
  
If(BUTTON=="LOG IN"){  
    Go to log in page  
}
```

3rd page:

STUDENT GRIEVANCE CELL

LOGIN

email/mobile number

password

REGISTER

```
Function login(USER,PASSWORD){  
    If(Entered Email/phone no){  
        USER=Entered Email\phone no  
    }  
    If(Entered password){  
        PASSWORD=Entered password  
    }  
    If(BUTTON=="log in "){  
        If(USER==Database user && PASSWORD==Databse password){  
            If(Database User type=="Student"){  
                Login as student  
                Go to homepage  
            }  
            If(Database User type=="Cell member"){  
                Login as cell member
```

```
        Go to homepage
    }
}
Else{
    Return false
}
}
If(BUTTON=="REGISTER"){
    Go to register page
}
```

4th page:

```
Function Homepage(){
    If(BUTTON=="HOME"){
        Go to Homepage
    }
    If(BUTTON=="HISTORY"){
        Go to History page
    }
    If(BUTTON=="PROFILE"){
        Go to Users Profile page
    }
    If(BUTTON=="STATUS"){
        Go to STATUS of current query page
    }
}
```

```
If(BUTTON=="RAISE QUERY"){
```

```
    Go to Raise query page
```

```
}
```

```
If(BUTTON=="LOGOUT"){
```

```
    Go to login/register page
```

```
}
```

5th page:



The screenshot shows a web application interface. At the top, there is a dark blue navigation bar with a user profile icon on the left and links for HOME, HISTORY, PROFILE, STATUS, RAISE QUERY, and LOGOUT. Below the navigation bar, the main content area has a light blue background. In the center, there is a table with the following data:

QUERY ID	DATE	TYPE	TITLE	STATUS
10234	22-03-21	DEPT	FEES	PENDING
2345	21-01-19	UNI	ID CARD	RESOLVED
12432	19-05-18	DEPT	LIBRARY	RESOLVED

Function History{

```
    Display(Raised Queries list from Query table)
```

```
    If(Query.clicked())
```

```
    {
```

```
        Display Solution from query table
```

```
    }
```

```
}
```

6th page:

Function Profile(){

image= image from student table

Roll no =roll no from student table

College = College name from student table

Student Name=Name from student table

Course=Course from student table

Email =Email form student table

Phone=Phone from student table

Display(image, Roll no, College, Student Name, Course, Email, Phone)
}

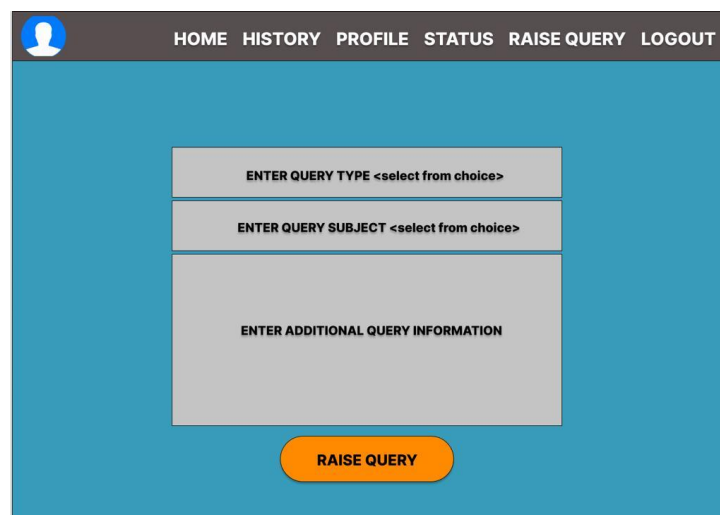
7th page:

Function Status(){

Display the status of current query

}

8th page:

A screenshot of a web application interface for raising a query. The interface has a dark blue header bar with a user profile icon on the left and navigation links: HOME, HISTORY, PROFILE, STATUS, RAISE QUERY, and LOGOUT. The main content area has a light blue background. In the center, there is a white form with three input fields: 'ENTER QUERY TYPE <select from choice>', 'ENTER QUERY SUBJECT <select from choice>', and 'ENTER ADDITIONAL QUERY INFORMATION'. Below these fields is an orange button labeled 'RAISE QUERY'.

Function Raise Query(){

Select Query type

Type query description which follows minimum criteria

Click send query button

Display query sent notification

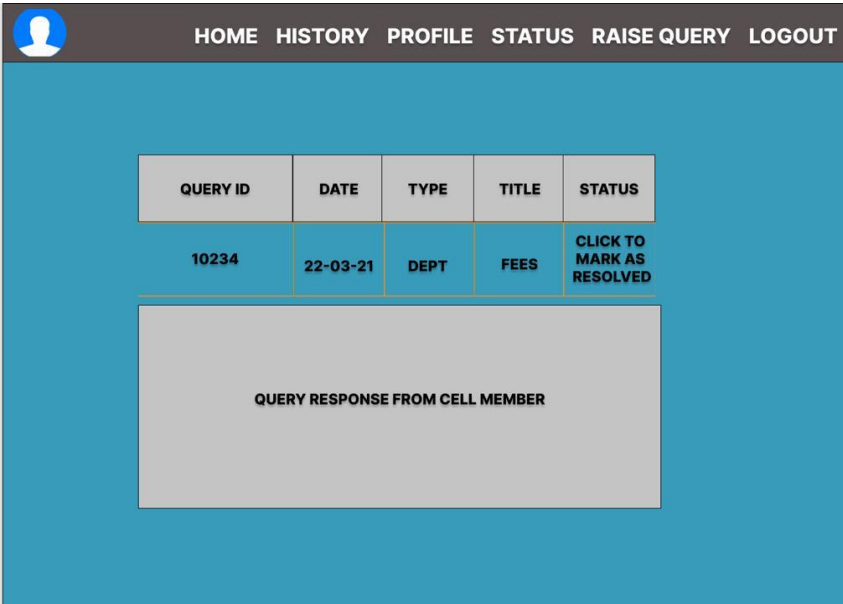
}

9th page:

Function Log out(){

Go to login/register page

}



The screenshot shows a web application interface with a dark blue header bar. On the left of the header is a user profile icon. To the right of the icon are navigation links: HOME, HISTORY, PROFILE, STATUS, RAISE QUERY, and LOGOUT. The main content area has a light blue background. It features a table with the following structure:

QUERY ID	DATE	TYPE	TITLE	STATUS
10234	22-03-21	DEPT	FEES	CLICK TO MARK AS RESOLVED

Below the table is a large grey rectangular box containing the text: QUERY RESPONSE FROM CELL MEMBER

4.3 ER Diagram and database Schema :

CHAPTER 5 : TESTING

5.1 Test cases :

- **Function index()**

Sub case 1: Input: Button=login

Output: direct to login page.

Sub case 2: Input: Button=register

Output: direct to register page.

- **Function register(user_name,password,register_as)**

Sub case 1: pass=confirmpass

Inputs: Paras (username)

paras.ch123 (pass)

student (register_as)

Output: registered as student and data is stored in the database(user table).

Sub case 2: pass!=confirmpass

Inputs: Paras (username)

paras.ch123 (pass)

student (register_as)

Output: will not be registered .

Sub case 3: pass=confirmpass

Inputs: Sagrika (username)

mh015 (pass)

cell member (register_as)

Output: registered as cell member and data is stored in the database(user table).

Sub case 4: pass!=confirmpass

Inputs: Sagrika (username)

mh015 (pass)

cell member (register_as)

output: will not be registered .

- **Function login(User,pass)**

If button= register à directed to register page

If button= login à the cases are as follows:-

Sub case 1: if user = database user & pass = database pass

Inputs: paraskhampa@gmail.com (user)

paras.its123 (pass)

student (user type)

Output: logged in as student and directed to home page.

Sub case 2: if user = database user & pass != database pass

Inputs: paraskhampa@gmail.com (user)

paras.its123 (pass)

student (user type)

Output: shows “incorrect password”.

Sub case 3: if user = database user & pass = database pass

Inputs: sagrika.malhotra@gmail.com (user)

itsmalhotra123 (pass)

cell member (user type)

Output: logged in as cell member and directed to home page.

Sub case 2: if user = database user & pass != database pass

Inputs: sagrika.malhotra@gmail.com (user)

itsmalhotra123 (pass)

cell member (user type)

Output: shows “incorrect password”.

- **Function homepage()**

Sub case 1: Input: Button=home

Output: direct to homepage.

Sub case 2: Input: Button=history

Output: direct to history page.

Sub case 3: Input: Button=profile

Output: direct to user’s profile.

Sub case 4: Input: Button=status

Output: shows status of current query page.

Sub case 5: Input: Button=raise query

Output: direct to raise query page.

Sub case 6: Input: Button=logout

Output: direct to login/register page.

- **Function history()**

Sub case 1: Input: Button=history

Output: list of raised queries is displayed.

Sub case 2: Input: a query is being clicked

Output: solution from the query table is displayed.

- **Function Profile()**

Inputs: button = profile

Output : image , roll no. clg , student name, course, email,
phone no

from student table(database) is displayed.

- **Function status()**

Sub case 1: Input: Button=status

Output: status of current query is displayed.

- **Function Raisequery()**

Input: admission (query type)

Regarding verification of documents (description)

Send query (button clicked)

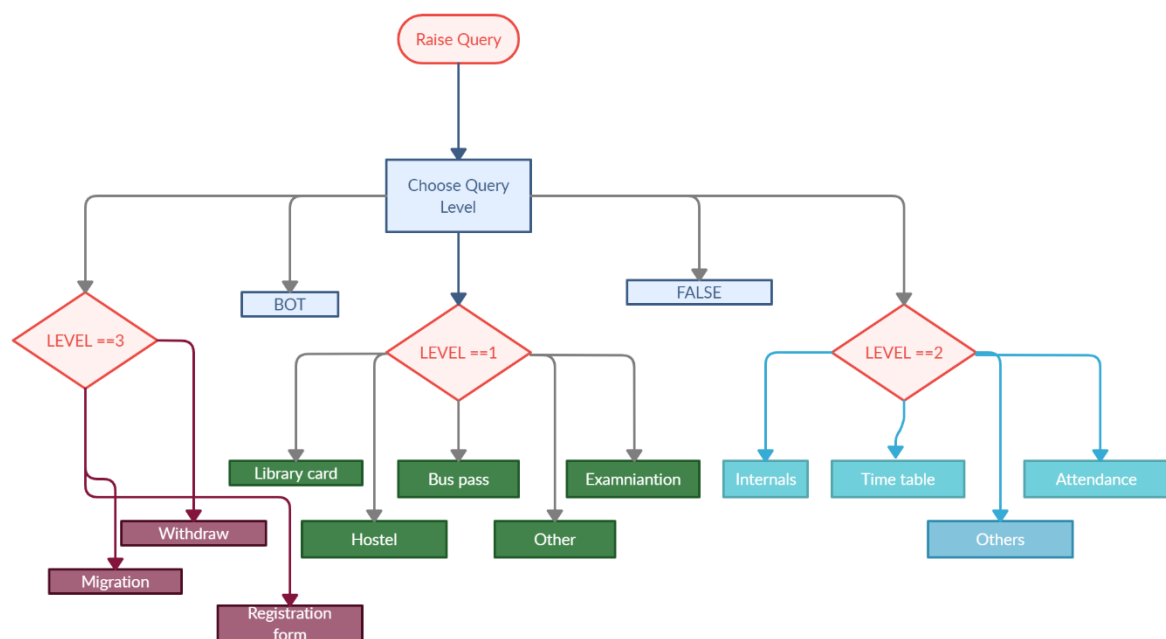
Output: sent query notification is displayed.

- **Function log_out()**

Sub case 1: Input: Button=logout

Output: direct to login/register page.

5.2 Flow Graph :



Module Coding :

```

[Untitled1]
1  #include<iostream>
2  #include<string>
3  #include<bits/stdc++.h>
4
5  using namespace std;
6
7  void post_query(int level,int type,string description) {
8  }
9
10 int bot(int type,int level)
11 {
12     return 0;
13 }
14
15 void raise_query() {
16     cout<<endl<<"CHOOSE YOUR QUERY LEVEL: ";
17     int level,type;
18     string description;
19     cout<<"\n1.COLLEGE\n2.DEPARTMENT\n3.UNIVERSITY\n";
20     cin>>level;
21     if(level==1) {
22         cout<<endl<<"CHOOSE YOUR QUERY TYPE: ";
23         cout<<"\n1.library card\n2.bus-pass\n3.examinations\n4.hostel\n5.other\n";
24         cin>>type;
25     }
26     else if(level==2) {
27         cout<<endl<<"CHOOSE YOUR QUERY TYPE: ";
28         cout<<"\n1.time-table\n2.attendance\n3.internal assessment\n4.other\n";
29         cin>>type;
30     }
31     else if(level==3) {
32         cout<<endl<<"CHOOSE YOUR QUERY TYPE: ";
33         cout<<"\n1.registration form\n2.migration\n3.withdrawal\n4.exam/college fees\n5.other\n";
34         cin>>type;
35     }
36     if(!bot(level,type)){
37         cout<<"\ndescribe your query in 50 words:\n";
38         fflush(stdin);
39         getline(cin,description);
40
41         post_query(level,type,description);
42         cout<<endl<<"Keep checking your status tab for updates from grievance cell.";
43     }
44     else cout<<endl<<"Wrong choice. Try again.";
45 }
46
47
48
49
50
51 }
52
53 int main() {
54     raise_query();
55
56     return 0;
57 }
58

```

5.3 Cyclometric Complexity :