University Query Handling System

REQUIREMENTS SPECIFICATION

INTRODUCTION

A specific application catering to the need of all individuals in an institution such as universities. It will provide lodging/reporting, answering, forwarding of complaints from a complainant to the respective responder.

BACKGROUND

The most difficult part in an institution is either to get information or report issues with the management. We look forward to automating this system so that the whole process is streamlined. The concerned authorities get the specifics so that they can help students, teaching staff and non-teaching staff.

- **Why did we pick this ?**

A simple to use application that allows every user in an institute to voice his opinion, provide suggestions, report incidents or lodge complaints.

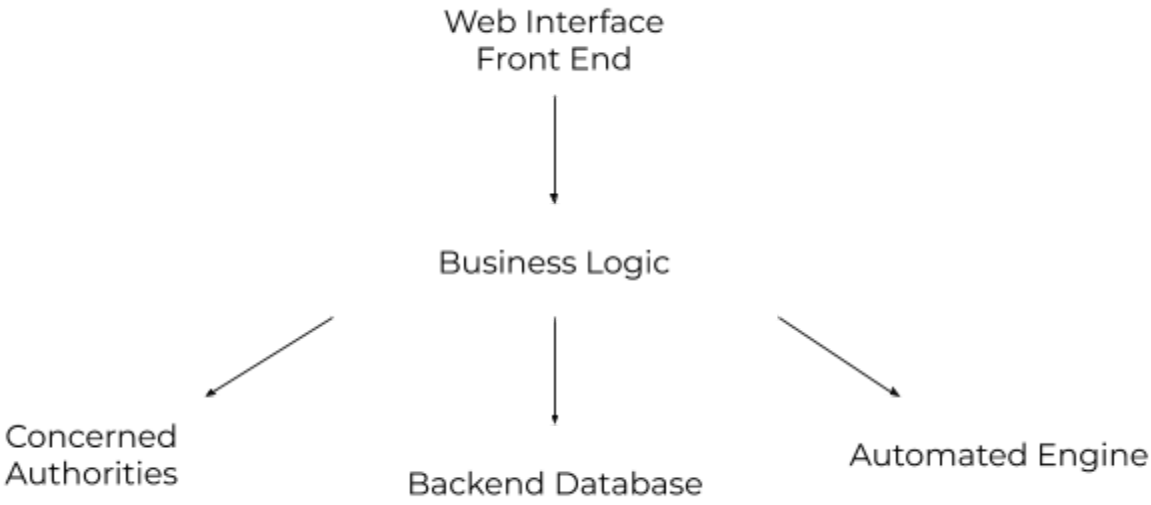
- **What problem is it solving ?**

Complaints in an educational institution often go unaddressed. Queries are a very slow process. Having a process in place will quicken the roundabout time. An interface for the user will make lodging queries much easier.

- **How are we solving this problem ?**

The application will have a few components.

A frontend web application to provide for easy use of lodging queries, business logic that will coordinate the queries along it's required route, and a backend to store, fetch and log data.



FUNCTIONAL REQUIREMENTS

General functionality, end user requirements(what are the end users and what they should be able to do, administrative user requirements, (mention details of what each user should do)

GENERAL

1. Role based login for users(students, teaching staff, etc) , Administrators and system administrator.
2. Queries can be handled in 2 ways: Manually or automatically from a database. We call the former general queries, and the latter specific queries.
3. General queries can be general questions regarding the University guidelines and processes, or complaints and suggestions.
4. All users will be able to make general queries. General queries can be sent anonymously too.

5. General queries will be looked into by Administrators, who will either respond to them, or forward it to the appropriate department/authority.
6. Users would be able to make specific queries too, as described below based on their roles:
    a. *Student specific functionalities:* will be provided with functionality to check details such as fees, courses, grades, etc.
    b. *Teachers specific functionalities:* Upload attendance for a class, upload grades, get timetable for their classes, etc.
    c. *Non- Teaching Staff specific functionalities:* will be able to see work schedule, salary details, student details, etc.
    d. Redressal officer are responsible for answering general queries and redressing complaints.

END USER REQUIREMENTS

1. The end users are basically those who want to send a query.
2. This may comprise as a teacher, student, or general staff.
3. Students should be able to :
    a. Ask for their fees status, receipt and structure.
    b. Make a general query and add appropriate tags (such as “Accounts” for fee related queries).
    c. Track general queries (queries that requires manual intervention).
    d. Lodge a complaint against the services of a particular department.
    e. Report any malicious behaviour that occurs (in the campus/ with a student).
    f. Request for a book in the library, check current book issue status, check for fines.
    g. Check placement status, companies registered, companies yet to come.
    h. Check their assessment scores, current attendance.
    i. Post suggestions.

ADMINISTRATIVE REQUIREMENTS

Listed below are the administrators:

1. Query manager
    a. In case a particular query has not been tagged to a particular department or it is a generic query
    b. It is then reviewed by this person and routed as found fit.
2. Suggestion reviewer
    a. Suggestions are reviewed by this person, if found to not be spam,it is then routed over to the respective individual.
3. Special officer
    a. Will look over reports that cite malicious behaviour in the campus.
    b. Will be given privilege over anonymous reports to trace user.
    c. Has functionality to block a person's college accounts and revoke status when needed.

NON – FUNCTIONAL REQUIREMENTS

User interface requirements

- Users should be able to send queries, check responses.
- Minimal use of the interface should lead to the required responses.
- Approximate waiting time must be shown for responses that require manual intervention.

Performance requirements

- Responses not requiring manual intervention must take less than 10s.
- End user must be provided with a seamless interface that eliminates quick interaction.
- Must support upto 1000 users at a time.

Security

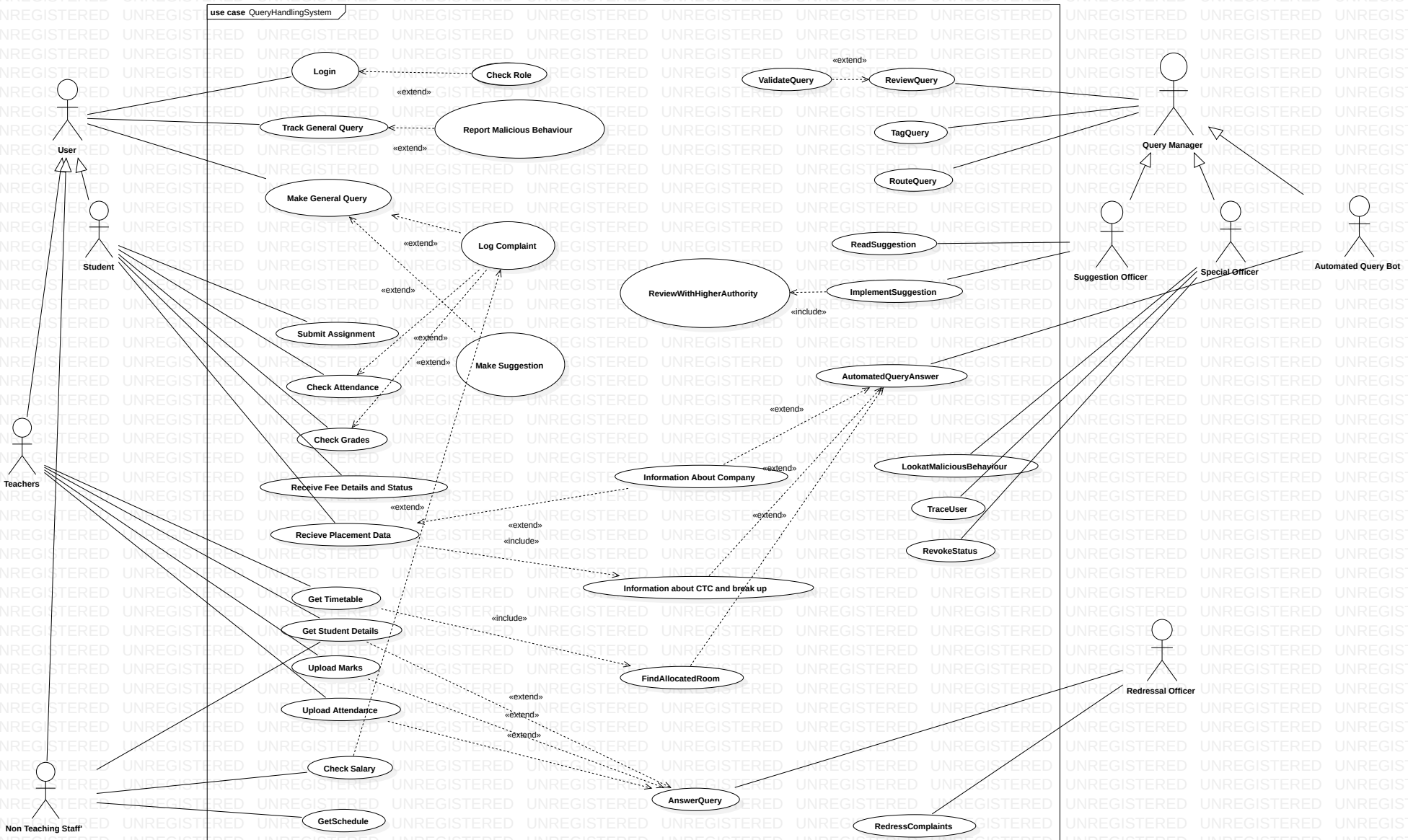
- Each kind of user will have specific access control.

- Special Officer can change the access control of any user during emergency circumstances.
- Unsuccessful login attempts must be logged and an alert must be generated.

OTHER REQUIREMENTS

- Application is expected to run 24x7 365 with an availability of 99%.
- Minimize the effort the end user has to put to query.

Model::QueryHandlingSystem



USE CASE DESCRIPTION

1.

Use Case : Login
Summary : General login for an end user.
Actors : All end users such as students, teachers and non teaching staff
Preconditions : Login credentials must be entered and login button must be pressed.
Description : This is a general login for all end users.
All end users have the same portal to login from.
Credentials are checked and then routed to their particular web page accordingly.
Exceptions : Incorrect credentials/ not existing.
Postconditions : User is routed to his/her respective site.

2.

Use Case : Check role
Summary : Role checking for user.
Actors : All end users such as students, teachers and non teaching staff.
Preconditions : Credentials must be received.
Description : Each user tries to login, then credentials are checked and routed to their respective site.
This functionality is to route each user properly.
Exceptions : Wrong credentials.
Postconditions : Result is sent back to login and successfully routed.

3.

Use Case : Track general query
Summary : To track the result of query.
Actors : All end users such as students, teachers and non teaching staff.
Preconditions : Query must be selected.
Description : When a user makes a query he will want to know the response.
In case he doesn't get the response, he can check the status to see where the query is and approximately how long it takes
Exceptions : No query has been made.
Postconditions : Query status is shown.

4.

Use Case : Make general query
Summary : Combination of different functionalities to make a generic query.
Actors : All end users such as students, teachers and non teaching staff. Extended functionality.

Preconditions : Entering of query.
Description : Making of a general query such as a complaint/ suggestion etc that has manual intervention.
Exceptions : User has been banned.
Postconditions : Query has been made and sent to the respective department.

5.
Use Case : Report malicious behaviour
Summary : Makes a malicious behaviour query.
Actors : All end users such as students, teachers and non teaching staff.
Preconditions : Entering of details.
Description : Malicious behaviour is that which is against the campus rules.
This kind of query is routed to the special officer.
Action can be taken to block the account.
Exceptions : User has been banned.
Postconditions : Query has been made and sent to the special officer.

6.
Use Case : Log Complaint
Summary : Logs a complaint.
Actors : All end users such as students, teachers and non teaching staff.
Preconditions : Entering of complaint.
Description : In case of any inconvenience, users can log complaints, they will be handled by the query handler admin who transfers it accordingly.
Action will be taken by the respective department.
Exceptions : User has been banned, too many complaints in a short time.
Postconditions : Complaint has been logged and sent to the query handler.

7.
Use Case : Make suggestion.
Summary : Functionality to make suggestions.
Actors : All end users such as students, teachers and non teaching staff.
Preconditions : Entering of suggestion.
Description : In case of any user has suggestions to make, he can do it with this.
The suggestion is sent to the suggestion reviewer.
Suggestion reviewer routes it to the respective department if seemed fit.
Exceptions : User has been banned, too many suggestions in a short time.
Postconditions : Suggestion has been logged and sent to the suggestion reviewer.

8.
Use Case : Submit assignment.
Summary : Functionality to make submit an assignment.
Actors : Student End User.
Preconditions : Assignment solution must be selected.
Description : Every user has assignments given by the teacher.
A user can submit his assignment which is then sent to the teacher.
Exceptions : User has been banned, can't submit after deadline.
Postconditions : Assignment has been submitted and can be graded by the teacher.

9.
Use Case : Check attendance.
Summary : Functionality to make check attendance for how many classes attended.
Actors : Student End User.
Preconditions : Clicking of option.
Description : Attendance is logged in for each and every student for every class.
This can be checked by the student at any time.
Exceptions : User has been banned.
Postconditions : Attendance for all courses is returned and shown.

10.
Use Case : Check grades.
Summary : Functionality to check grades.
Actors : Student End User.
Preconditions : Clicking of option.
Description : Every user has grades given by the teacher.
This functionality lets users see their grades
Exceptions : User has been banned, grades not issued.
Postconditions : List of grades are sent and viewed.

11.
Use Case : Fee details and status.
Summary : Functionality to view fee details and status.
Actors : Student End User.
Preconditions : Clicking of option.
Description : Each student will have fee to be paid.
They may have different fees due to various factors.
This functionality lets users see their fee details.
Fee status is whether paid or not and due date.

Exceptions : None
Postconditions : Fee details and status are sent.

12.
Use Case : Receive placement data.
Summary : Functionality to view placement information.
Actors : Student End User.
Preconditions : Student must be undergoing placements.
Description : Applicable to only those students undergoing placements.
Has two functionalities in itself - information out the company and CTC breakup.
Placement status is also shown - placed in how many companies, which tier etc.
Exceptions : Student not undergoing placements will not have this functionality.
Postconditions : Placement information is sent and viewable.

13.
Use Case : Information about company.
Summary : Returns company information.
Actors : Student End User.
Preconditions : Student must be undergoing placements.
Description : Companies often have details which they want to showcase.
They include and are not limited to sector, tier, website.
Such details are useful when applying.
Exceptions : Student not undergoing placements will not have this functionality.
Postconditions : Company details are sent - sector, tier etc.

14.
Use Case : Information about CTC and break-up.
Summary : Returns CTC and break-up.
Actors : Student End User.
Preconditions : Student must be undergoing placements.
Description : Students often apply to companies after looking at the CTC.
Sometimes the CTC can be misleading.
A break-up will give further information - base salary, stock-options etc.
Exceptions : Student not undergoing placements will not have this functionality.
Postconditions : CTC and break-up data is sent.

15.
Use Case : Get timetable.
Summary : Returns timetable.

Actors : Teacher End User.
Preconditions : None.
Description : Teachers often require to refer to a timetable due to busy schedule. This provides a teacher with all their assigned classes according to the subject and time.
Exceptions : Only for teachers.
Postconditions : Timetable is sent to be shown.

16.
Use Case : Get student details.
Summary : Returns details of all students under a course of a teacher.
Actors : Teacher End User.
Preconditions : Must have typed the course name/ student name.
Description : If a teacher requires to search for a student, then this functionality may be used.
Can use course name or student name to search.
Exceptions : Only for teachers.
Postconditions : List of possible students are sent and shown.

17.
Use Case : Upload marks.
Summary : Set marks/ grades for students.
Actors : Teacher End User.
Preconditions : Course should be selected.
Description : Easily assign grades to tests for each student of a course taught by that teacher.
Exceptions : Only for teachers, teacher can only assign grades for a course she's teaching.
Postconditions : Grades are set.

18.
Use Case : Upload attendance.
Summary : Upload attendance for a class.
Actors : Teacher End User.
Preconditions : A set of students must have been selected.
Description : Teachers can select attendance for those students who are posted. This is updated for each and every individual student.
Exceptions : Only for teachers, can update for only a class that has been taught.
Postconditions : Attendance is updated.

19.
Use Case : Find Allocated Room.
Summary : Functionality to allocate room for course.
Actors : Teacher End User.
Preconditions : Used for time-table generation.
Description : Algorithms have to be used to allocate rooms for courses and classes This functionality does so and is an extension of the time-table functionality.
Exceptions : Only for teachers. Extended functionality.
Postconditions : Room, time and course for it are sent back.

20.
Use Case : Check salary.
Summary : Functionality to check for salary.
Actors : Non -teaching staff.
Preconditions : Select the functionality.
Description : Staff will want to see their salary.
Salary, break-up and other benefits details are returned.
Exceptions : Only for non-teaching staff.
Postconditions : Salary is sent back to be displayed.

21.
Use Case : Check schedule.
Summary : Functionality to check for schedule.
Actors : Non-teaching staff End User.
Preconditions : Select the functionality.
Description : Schedule is returned for those who have it.
Exceptions : Only for those whose schedule is pre-set.
Postconditions : Schedule is returned and shown.

22.
Use Case : Check student details.
Summary : Functionality to check for student details.
Actors : Non-teaching staff End User.
Preconditions : Select the functionality and enter the USN.
Description : In case an admin needs to see a student's details, after entering USN, they can be returned and displayed.
Exceptions : Wrong USN.
Postconditions : Particular details are returned.

23.
Use Case : ReviewQuery
Summary : When a query is posted at the portal, the Actor Reviews the query in order to either tag it or forward to the concerned Department.
Actors : Query Manager
Preconditions : A query is to be present for it to be reviewed.
Description : Suppose one of the user adds a query/complaint, it has to be checked for grammatical errors, that means a thorough review is done and is made ready to be tagged or routed.
Exceptions : Query is corrupted or is not present.
Postconditions : Reviewed Query to be Tagged is made available.

24.
Use Case : TagQuery
Summary : When a query is posted at the portal, the Actor tags the query to a department.
Actors : Query Manager
Preconditions : Reviewed Query.
Description : Suppose one of the user adds a query/complaint, When the query is reviewed, it has to be tagged to a department in order to route the query.
Exceptions : Query is not reviewed.
Postconditions : Tagged Query to be Routed is made available.

25.
Use Case : RouteQuery
Summary : Route the query to the concerned Department.
Actors : Query Manager
Preconditions : A query is to be tagged to a department.
Description : Suppose one of the user adds a query/complaint, it has to be tagged to a department, then the query is forwarded to the concerned department.
Exceptions : Query is not tagged
Postconditions : Query reaches concerned department

26.
Use Case : ValidateQuery
Summary : Check if query is standard and is free of errors/corrupted.
Actors : Query Manager

Preconditions : A query is to be present for it to be reviewed.
Description : The query is checked for any corruption or errors, if it contains the query is discarded.
Exceptions :
Postconditions : Query validated.

27.
Use Case : ReadSuggestion
Summary : Suggestion provided is to be read and understood
Actors : Suggestion Officer
Preconditions : Suggestion is given and not a query
Description : In case a user suggests a suggestion to the already management, it can be incorporated by the Suggestion Officer.
Exceptions : Suggestion is not understandable, suggestion is not compliant with system.
Postconditions : Suggestion is read.

28.
Use Case : ImplementSuggestion
Summary : Suggestion provided is to be implemented.
Actors : Suggestion Officer
Preconditions : Suggestion is given and not a query
Description : In case a user suggests a suggestion to the already management, it can be incorporated by the Suggestion Officer.
Exceptions : Higher Authority doesn't agree
Postconditions : Suggestion is implemented.

29.
Use Case : ReviewWithHigherAuthority
Summary : a higher authority in the management reviews and checks if it's feasible.
Actors : Suggestion Officer
Preconditions : suggestion is sent from the Suggestion Officer
Description : In case a user suggests a suggestion to the already management, it can be incorporated by the Suggestion Officer.
Exceptions :
Postconditions : Suggestion is implementable.

30.
Use Case : AutomatedQueryAnswer

Summary : Easily accessible information can be directly retrieved by the Automated Query Bot and presented to the users.
Actors : Automated Query Bot
Preconditions : simple queries
Description : A user inputs general information query which can be retrieved from the database, the bot can do it easily and send it back. This reduces the overhead of having a dedicated query manager for simple queries.
Exceptions : Query is complex and data cannot be retrieved easily
Postconditions : Query is successfully processed

31.
Use Case : AnswerQuery
Summary : General Query is answered by the redressal officer
Actors : Redressal Officer
Preconditions : complex tagged queries
Description : A user inputs general information query which can be retrieved from the database, the Redressal Officers takes into account various parameters and then answers the query. The Query is properly tagged and it belongs to a department.
Exceptions : query is tagged wrong
Postconditions : Query is successfully processed

32.
Use Case : LookAtMaliciousBehaviour
Summary : When a general query has malicious behaviour, the special officer reviews it.
Actors : Special Officer
Preconditions : Malicious Event of Query
Description : A query is checked and found malicious by the software, the special officer takes a look at it and reviews it.
Exceptions :
Postconditions : Query is successfully reviewed for malicious behaviour.

33.
Use Case : TraceUser
Summary : When a general query has malicious behaviour, the special officer reviews it, since the user is anonymous, it has to be traced and therefore checked for its origin
Actors : Special Officer
Preconditions : Malicious Query

Description : A query is checked and found malicious by the software, the special officer takes a look at it and reviews it.Once that is done, the user is tracked and traced back to find the origin in order to punish the offender.
Exceptions : User cannot be traced
Postconditions : Query is successfully reviewed for malicious behaviour.

34.
Use Case : Revoke User
Summary : Remove access to the particular user.
Actors : Special Officer
Preconditions : Traced User with violations
Description : A query is checked and found malicious by the software, the special officer takes a look at it and reviews it.Once that is done, the user is tracked and traced back to find the origin in order to punish the offender. The offender(user) is revoked of its access to the portal.
Exceptions : Revoke unsuccessful due to internal error.
Postconditions : User access is revoked