Judiciary Information System

119CS0100	POKALA KUSAL	GROUP 4
119CS0101	PRIYANSHU KUMAR	
119CS0102	SUSHREE SATARUPA	
119CS0103	NITIN AGARWAL	

Table of Contents:-

Software Requirements Specification

Judiciary Information System (JIS)

119CS0100	POKALA KUSAL	GROUP 4
119CS0101	PRIYANSHU KUMAR	
119CS0102	SUSHREE SATARUPA	
119CS0103	NITIN AGARWAL	

Table of Contents

Problem Statement

Judiciary Information System (JIS) software:

• The attorney general's office has requested us to develop a Judiciary Information System (JIS),
  - o to help handle court cases
  - o to make the past court cases easily accessible to the lawyers and judges.

• For each court case, the following information is entered by the court registrar.
  - o Name of the defendant
  - o Defendant's Address
  - o The crime type (e.g., theft, arson, etc.)
  - o When committed (date)
  - o Where committed (location)
  - o Name of the arresting officer
  - o The date of the arrest

• Each court case is identified by a unique case identification number (CIN) which is generated by the computer.

• The registrar assigns a date of hearing for each case.
  - o For this the registrar expects the computer to display the vacant slots on any working day during which the case can be scheduled.

• Each time a case is adjourned, the registrar
  - o Enters the reason for the adjournment
  - o Assigns a new hearing date.

• If a hearing takes place on any day for a case, the registrar
  - o Enters the summary of the court proceedings
  - o Assigns a new hearing date.

• Also, on completion of a court case,
  - o the summary of the judgment is recorded
  - o the case is closed
  - o the details of the case is maintained for future reference.

• Other data maintained about a case include
  - o the name of the presiding judge
  - o the public prosecutor
  - o the starting date
  - o the expected completion date of a trial.

• The judges should be able to browse through the old cases for guidance on their judgment.

• The lawyers should also be permitted to browse old cases, but should be charged for each old case they browse.

• Using the JIS software, the Registrar of the court should be able to query the following:

(a) The currently pending court cases.

- In response to this query, the computer should print out the pending cases sorted by CIN.

- For each pending case, the following data should be listed:
  - o the date in which the case started,
  - o the defendant's
    - ▪ name,
    - ▪ address,
    - ▪ crime details,
    - ▪ the lawyer's name,
    - ▪ the public prosecutor's name
    - ▪ the attending judge's name.

(b) The cases that have been resolved over any given period.

- The output, in this case, should chronologically list the
  - o starting date of the case,
  - o the CIN,
  - o the date on which the judgment was delivered,
  - o the name of the attending judge,
  - o the judgment summary.

(c) The cases that are coming up for hearing on a particular date.

(d) The status of any particular case (cases are identified by CIN).

- The lawyers and the judges need to refer to past court cases.
  - o The lawyers need to refer to these to prepare for their line of arguments.
  - o The judges need to refer the past court cases to examine the lines of judgments given previously to similar cases.

- It should be possible to search for the history of past court cases by entering keywords.

- However, the lawyers should be charged for each time they see the details of a court case to recover some of the computerization costs.

- For this purpose, it is necessary to provide separate login accounts to the JIS software and keep track of how many court cases each lawyer views.

- The registrar should be able to create login accounts for the different users (i.e. judges, lawyers, etc) and should be able to delete these accounts.

1. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of the Judiciary Information System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate, and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the office of the Attorney General for its approval.

1.2. Scope of project

The Judiciary Information System will be designed to help handle court cases by maintaining a digital record and to make the complete data of past cases easily available to the lawyers and judges. The Court registrar acts as the system administrator and he has the privileges to enter a new case and maintain its details as it progresses until it is finally closed whereby it is archived and kept in the past record. The lawyers and judges have limited functionality: they can only browse past cases without modifying any details. The JIS will provide a common log-in interface to the registrar and all lawyers and judges from where they can access their respective accounts.

- The judges would be able to browse through the old cases for guidance on their judgment and to examine the lines of the judgment given previously to similar cases.
- It would be possible to search for the history of past court cases by entering keywords.
- The lawyers would be permitted to browse old cases but would be charged for each old case they browse.
- This system will allow the Registrar to see the details of the currently pending cases and the cases which have been resolved or the status of any particular case.

1.3. REFERENCES

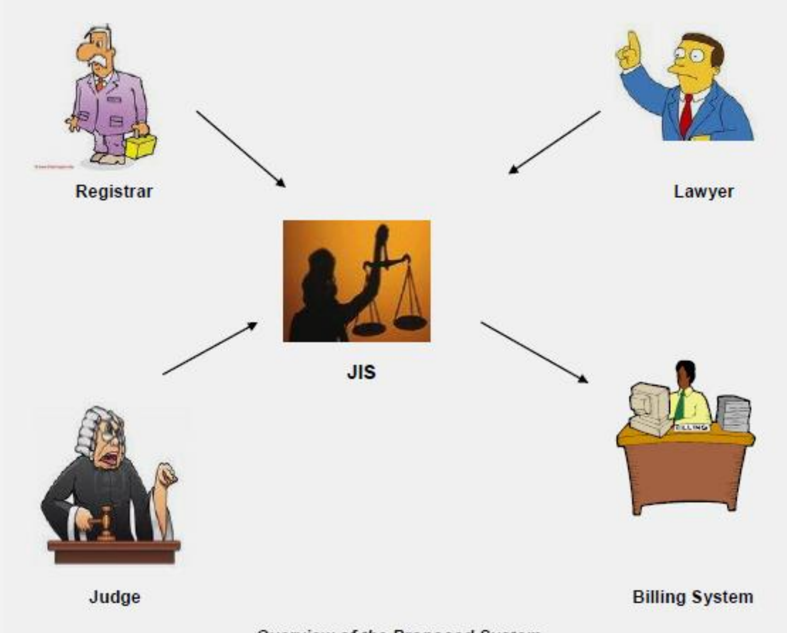
IEEE : IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

2. Overall description

1) PRODUCT PERSPECTIVE

The Judiciary Information System is a package to be used by the judges and lawyers to improve the efficiency in handling court cases. The system provides information related to the cases which have been resolved so that judges can get guidance on their judgment and the lawyers can get guidance on their cases. This system is the first of its kind and replaces the old system of browsing through physical documents and papers thus reducing the maintenance burden.

The complete overview of the system is shown in the overview diagram below:



Overview of the Proposed System

2) USER CHARACTERISTICS

The users of Judiciary Information System are the Registrar, the judges, the lawyers and the administrators who maintain the system.

The users are assumed to have basic knowledge of the computers, internet and the system. The administrators of the system should have more knowledge of the internals of the system and should be able to rectify the small problems that may arise due to disk crashes, power failures and other catastrophes to the system.

3) CONSTRAINTS

a. The information of all the past cases must be stored in a database that is accessible by the Judiciary Information System.
b. The billing system is connected to the Judiciary Information System (JIS) and the database used by the billing system must be compatible with the interfaces of the JIS.
c. The users must have their correct usernames and passwords to enter the JIS.
d. The files in which the information regarding the previous cases are stored should be secured against malicious deformations.

4) ASSUMPTIONS AND DEPENDENCIES

a. Full working of JIS is dependent on the availability of an internet connection.
b. The users have sufficient knowledge of computers and the internet.
c. The users know the English language as the user interface will be provided in English.
d. The system can access the previous cases database.

3. SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACES

3.1.1 User Interfaces

The user interface is basically divided into three main sections: the interface related to the Registrar, the judge and the lawyer and are as follows:

- **Registrar**

(a) **Registrar Log-In:** This button is placed on the home page of the software. The Registrar logs into the system by entering his user name and password. If the Registrar enters the wrong username or password, an error message pops up describing the error.

(b) **Input Case Details:** Once the Registrar logs into the system, he can enter all the case details by selecting this button. The Registrar finishes entering the details by selecting the Done option.

(c) **Display Dates:** After entering the details of the case, the Registrar selects this button to ask the computer to display the vacant slots on any working day during which the case can be scheduled. If no dates are available, a message regarding the same pops up.

(d) **Enter Summary:** After the Registrar logs into the system, he can enter the summary of the case by selecting this button and entering its summary.

(e) **Pending Cases:** This button appears after the Registrar logs into the system. He selects this button to see the details of the pending cases by entering their CIN.

(f) **Resolved Cases:** This button appears after the Registrar logs into the system. He selects this button to see the details of the resolved cases by entering their CIN.

(g) **Due Cases:** This button appears after the Registrar logs into the system. He selects this button to see which cases are scheduled on a particular date by entering the date.

(h) **Case Status:** This button appears after the Registrar logs into the system. He selects this button to see the status of the cases (Pending/Closed/Due) by entering their CIN.

(i) **Create New Account:** After the Registrar logs into the system, he can create a new account for judges or lawyers by selecting this button. On selecting this button, the Registrar has to choose whether he wants to create a new judge account or a new lawyer account.

(j) **Delete Account:** After the Registrar logs into the system, he can delete an existing account of a judge or a lawyer by selecting this button. On selecting this button, the Registrar has to choose whether he wants to delete a judge account or a lawyer account.

(k) **Log-Out:** This button appears once the Registrar logs into the system. On selecting this button, the Registrar logs out of the system and the home page of the software is displayed.

- **Judge**

(a) **Judges Log-In:** This button is placed on the home page of the software. The Judge logs into the system by entering his/her user name and password. If the judge enters the wrong username or password, an error message will pop up describing the error. Also, if his/her account does not exist, then an error message pops up regarding the same.

(b) **Resolved Cases:** This button appears after the judge logs into the system. He/She selects this button to see the details of the resolved cases by entering their CIN.

(c) **Log-Out:** This button appears once the judge logs into the system. On selecting this button, the judge logs out of the system and the home page of the software is displayed.

- **Lawyer**

(a) **Lawyers Log-In:** This button is placed on the home page of the software. The Lawyer logs into the system by entering his/her user name and password. If the lawyer enters the wrong username or password, an error message will pop up describing the error. Also, if his/her account does not exists, then an error message pops up regarding the same.

(b) **Resolved Cases:** This button appears after the judge logs into the system. He/She selects this button to see the details of the resolved cases by entering their CIN.

(c) **Pay Charges:** After the lawyer logs into the system, he/she can pay for the charges by selecting this button. After this, he/she will be redirected to the Billing System where he/she can pay for the dues.

(d) **Log-Out:** This button appears once the lawyer logs into the system. On selecting this button, the lawyer logs out of the system and the home page of the software is displayed.

3.1.2 Software Interfaces

•The Judiciary Information System connects to the database through JDBC. It opens the file which is required to perform a certain function.
•The Judiciary Information System connects directly to the JVM and hence is platform-independent and therefore runs on every operating system.
•A firewall will be used with the server to prevent unauthorized access to the system.

3.2 FUNCTIONS

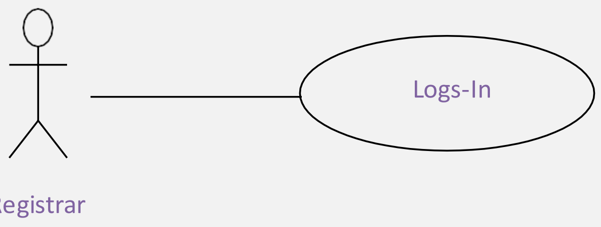
4. APPENDICES

2.2. Functional requirements specification

This section outlines the use cases for each of the active users separately. The lawyer and the judge have only one use case apiece while the registrar is the main actor in this system.

3.2.1. Use Case 1: Registrar Logs-In the Software

Diagram:



Brief Description:

The Registrar logs in the system and inputs the details of the case.

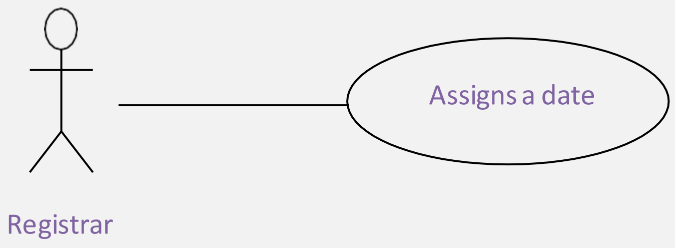
Input: The Registrar logs into the system by selecting the Registrar Log-In option. The defendant's name, defendant's address, crime type, date of crime, place of crime, name of arresting officer and the date of arrest for each case are entered by selecting the Input Case Details option.

Processing: The system opens the file which stores the log-in details of the users and matches it against the input.

Output: The computer automatically generates a unique case identification number (CIN) for each case.

3.2.2. Use Case 2: Date of Hearing

Diagram:



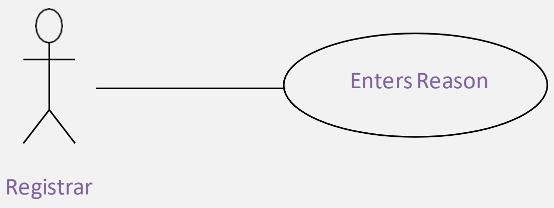
Brief Description:
After the unique CIN is generated, the Registrar assigns a date of hearing for the case.
Input: The Registrar selects the Display Dates option.

Processing: The system opens the file which stores the dates and checks if they are occupied or not and prints the non-occupied dates.

Output: The computer displays the vacant slots on any working day during which the case can be scheduled.

3.2.3. Use Case 3: Reason for Adjournment

Diagram:



Brief Description:
Reason of adjournment is entered if any case is adjourned.

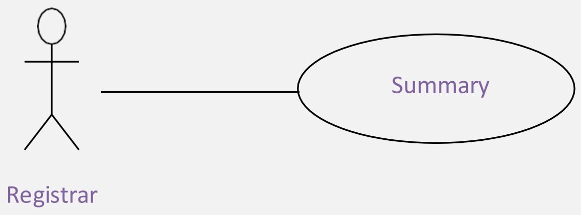
Input: The Registrar enters the reason due to which the case was adjourned by selecting Enter Summary option and selects the Display Dates option.

Processing: The system opens the file which stores the case details and the Registrar writes the reason into that file and closes it.

Output: A new hearing date is assigned for that case.

3.2.4. Use Case 4: Summary of Court Proceedings

Diagram:



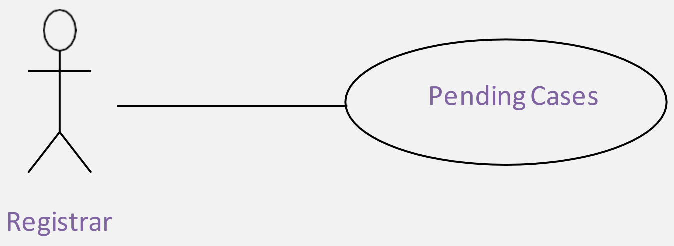
Brief Description:
If hearing of a case takes place, the summary of the court proceedings are entered, the judgement is recorded and the case is closed but the details of the case are maintained for future reference.
Input: The Registrar enters the summary of the case by selecting Enter Summary option and selects the Display Dates option for new hearing date.

Processing: The system opens the file which stores the case details and the Registrar writes the summary into that file and closes it.

Output: A new hearing date is assigned for the case.

3.2.5. <u>Use Case 5:</u> Currently Pending Court Cases

Diagram:



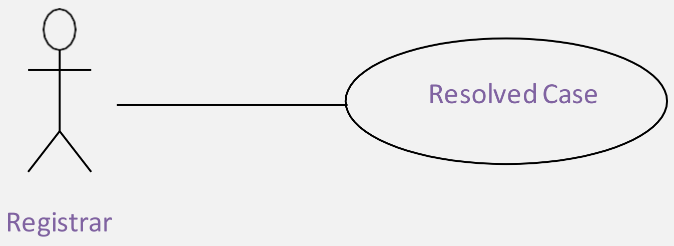
Brief Description:
This function gives the details of the currently pending cases when queried by the Registrar.
<u>Input:</u> The Registrar queries about the pending cases by selecting the Pending Cases option.

<u>Processing:</u> The system opens the file which stores the pending cases details and the Registrar reads from that file and closes it.

<u>Output:</u> The computer prints out the pending cases sorted by their CIN. For each pending case, the following data are listed: the date in which the case started, the defendant's name, address, crime details, the lawyer's name, the public prosecutor's name and the attending judge's name.

3.2.6. <u>Use Case 6:</u> Resolved Cases

Diagram:



Brief Description:

This function displays the details of the resolved cases over any given period.

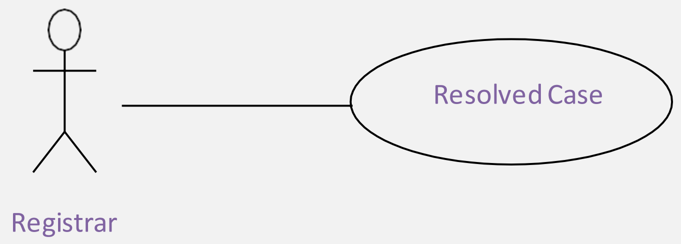
<u>Input:</u> The Registrar queries about the resolved cases by selecting the Resolved Cases option.

<u>Processing:</u> The system opens the file which stores the resolved cases details and the Registrar reads from that file and closes it.

<u>Output:</u> The computer chronologically lists the starting date of the case, the CIN, the date on which the judgement was delivered, the name of the attending judge and the judgement summary.

3.2.7. <u>Use Case 7:</u> Cases on a particular date

Diagram:



Brief Description:

This function lists the cases due on a particular date.

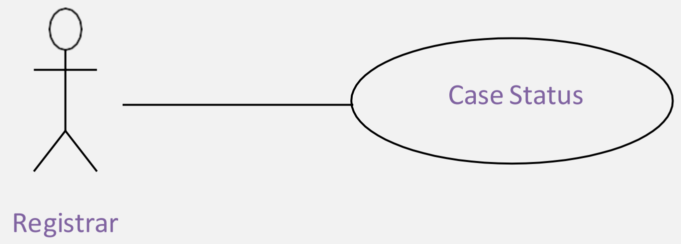
<u>Input:</u> The Registrar selects the Due Cases option and enters the date of hearing.

<u>Processing:</u> The system opens the file which stores the due cases details and the Registrar reads from that file and closes it.

<u>Output:</u> All the cases that are scheduled on that day are listed in the form of their CIN.

3.2.8. <u>Use Case 8:</u> Case Status

Diagram:



Brief Description:

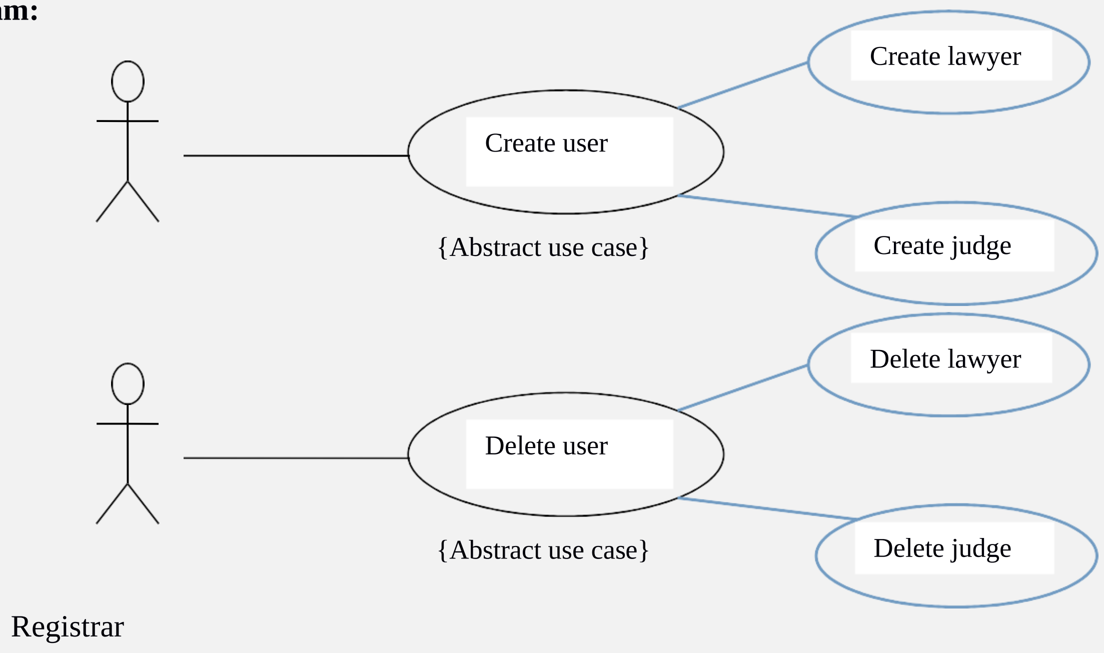
This function displays the status (Pending/Closed/Due) of any particular case queried by the Registrar.
<u>Input:</u> The Registrar selects the Case Status option and enters the CIN of the case he is interested in.

<u>Processing:</u> The system opens the file which stores the cases details and the Registrar reads the summary from that file and closes it.

<u>Output:</u> The computer displays the status of the particular case.

3.2.9. Use Case 9: Create/Delete Accounts

Diagram:



Brief Description:

This function allows the Registrar to create or delete judges/lawyers' accounts.

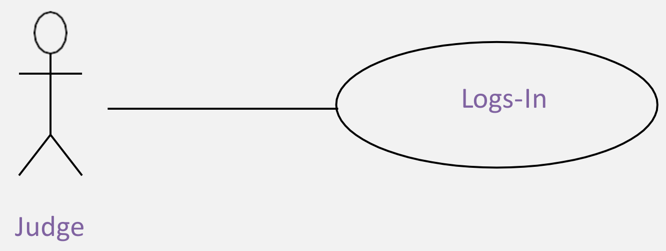
Input: The Registrar creates accounts by selecting the Create New Account option and entering the name of the judge/lawyer. He deletes an account by selecting the Delete Account option and entering the name of the judge/lawyer.

Processing: The system opens the file which stores the log-in details of the users and creates/deletes the corresponding user's details.

Output: A username and password are created for every account created and deleted for every account deleted.

3.2.10. Use Case 10: Judges Log-In

Diagram:



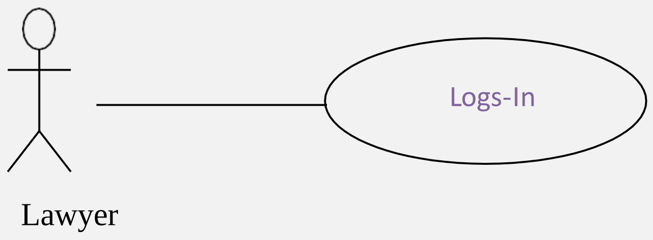
Brief Description:

This function allows the judges to log into the JIS and browse through the previous case history to get guidance on their decisions.
Input: The judges log into the system by selection the Judges Log-In option and can

select the previous cases by selecting the Resolved Cases option and entering key words like their CIN.
Processing: The system opens the file which stores the log-in details of the users and matches it against the input.
Output: The case details of the particular case are displayed.

3.2.11. Use Case 11: Lawyers Log-In

Diagram:



Brief Description:

This function allows the lawyers to log into the JIS and browse through the previous case history to get guidance on similar cases.

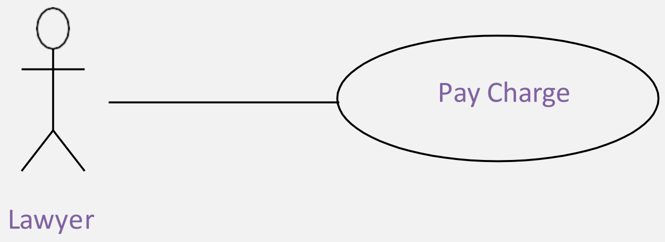
Input: The lawyers log into the system by selection the Lawyers Log-In option and can select the previous cases by selecting Resolved Cases option and entering key words like their CIN.

Processing: The system opens the file which stores the log-in details of the users and matches it against the input.

Output: The case details of the particular case are displayed. Also, the number of previous cases views for each lawyer is displayed.

3.2.12. Use Case 12: Pay Charge

Diagram:



Brief Description:

This function allows the lawyers to clear their dues for viewing previous court cases.

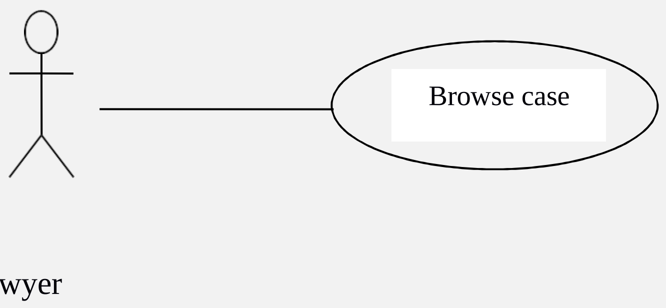
Input: The lawyers can pay for their charges by logging into JIS and selecting Pay Charges.

Processing: The system opens the file which stores the amount details of the lawyers and resets the amount to the NIL of the corresponding lawyer.

Output: This connects the JIS to the Billing System which generates the printed bill and resets the charges to NIL for the lawyer.

3.2.13. Use Case 13: Browse the case

Diagram:

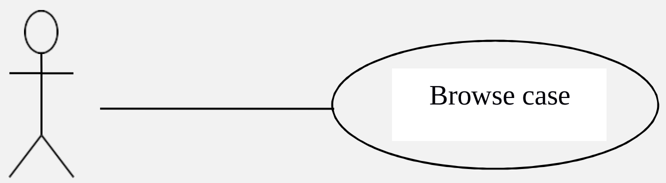


Brief description:
The lawyer searches cases by keyword and then selects the case to be viewed.
Initial step-by-step description:
Before this use case can be initiated, the lawyer has to be logged in to the system.

1. Lawyer selects ‘Browse case’ option.
2. Lawyer enters keywords to search by.
3. The system generates a list of cases that match the keywords.
4. Lawyer selects a case to be viewed.
5. System displays details of the selected case.
6. Lawyer returns to list of cases.
7. Lawyer selects another case or returns to the home screen.
8. When the lawyer finally returns to the home screen, the system updates the record of number of cases viewed by a lawyer.

3.2.14. Use Case 14: Browse the case

Diagram:

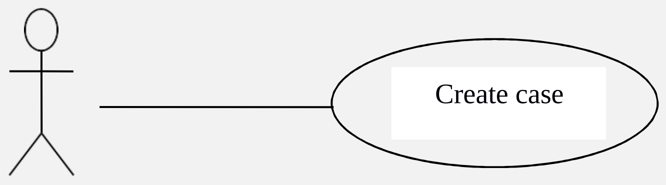


Judge

Brief description:
The judge searches cases by keyword and then selects the case to be viewed.
Initial step-by-step description:
Before this use case can be initiated, the judge has to be logged in to the system.
1. Judge selects the ‘Browse case’ option.
2. Judge enters keywords to search by.
3. The system generates a list of cases that match the keywords.
4. Judge selects a case to be viewed.
5. System displays details of the selected case.
6. Judge returns to list of cases.
7. Lawyer selects another case or returns to the home screen.

3.2.15. <u>Use Case 15:</u> Create a case

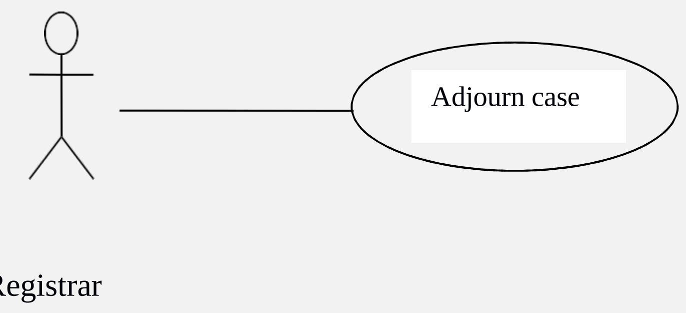
Diagram:



Registrar

Brief description:
The registrar enters the details of a new case into the system.
Initial step-by-step description:
Before this use case can be initiated, the registrar has to be logged in to the system.
1. Registrar selects the ‘Create case’ option.
2. Registrar enters details of the new case and assigns a date of hearing.
3. The system generates a list of empty slots on the assigned date if any.
4. Registrar selects a slot or assigns a new date of the hearing if no slot is available until a slot is finally selected.
5. System generates a Case Identification Number
6. Registrar returns to home screen.

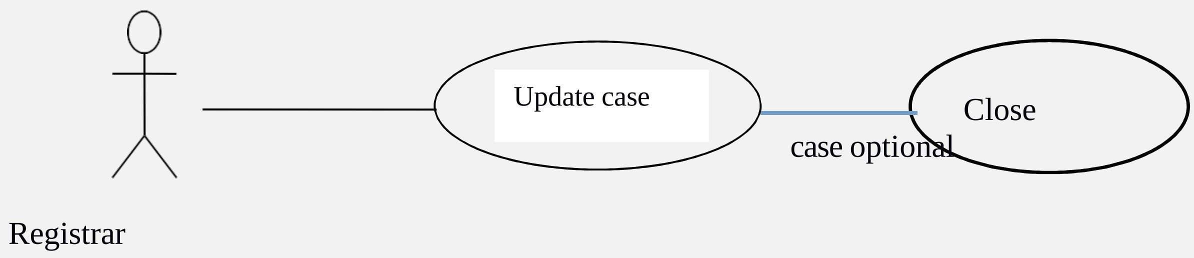
3.2.16. **Use Case 16:** Adjourn case
Diagram:



Brief description:
The registrar adjourns a case to a later date.
Initial step-by-step description:
Before this use case can be initiated, the registrar has to be logged in to the system.

1. Registrar selects a case from the list of all cases (or a particular query result) displayed on his home screen.
2. System displays details of the case along with various options.
3. Registrar selects the ‘Adjourn’ option.
4. Registrar enters reason of adjournment and assigns a new date of hearing.
5. The system generates a list of empty slots on the assigned date if any.
6. Registrar selects a slot or assigns a new date of the hearing if no slot is available until a slot is finally selected.
7. System updates case details on confirmation.
8. Registrar returns to the home screen.

3.2.17. **Use Case 17: Update the case**
Diagram:



Brief description:
The registrar updates case after a hearing happens.
Initial step-by-step description:
Before this use case can be initiated, the registrar has to be logged in to the system.

1. Registrar selects a case from the list of all cases (or a particular query result) displayed on his home screen.

2. System displays details of the case along with various options.
3. Registrar selects the ‘Update’ option.
4. Registrar enters summary of proceedings and assigns new date of hearing.
5. The system generates a list of empty slots on the assigned date if any.
6. Registrar selects a slot or assigns a new date of the hearing if no slot is available until a slot is finally selected.
7. System updates case details on confirmation.
8. Registrar returns to the home screen.

3.2.18. <u>Use Case 18:</u> Close the case

Diagram:

<<included in update case>>

Brief description:

The registrar closes a current case.

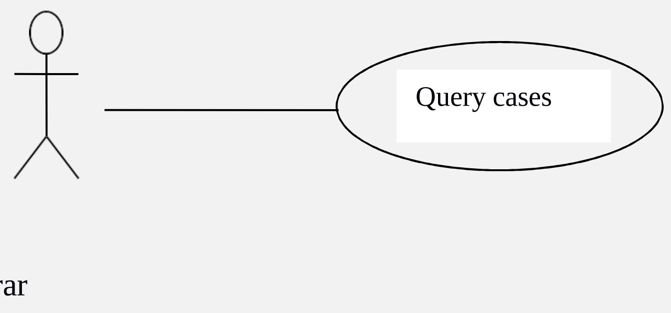
Initial step-by-step description:

Before this use-use case be initiated, the registrar has to be logged in to the system. Further, he must have selected the ‘Update’ option for a case and entered the summary of proceedings.

1. Registrar selects the ‘Close’ option.
2. Registrar enters judgment summary of the case to be closed.
3. System updates case details on confirmation.
4. Registrar returns to the home screen.

3.2.19. <u>Use Case 19:</u> Query cases

Diagram:



Brief Description:

The registrar queries the list of stored cases by various parameters.

Initial step-by-step description:

Before this use case can be initiated, the registrar has to be logged in to the system.

1. Registrar selects query parameter from a list: pending, resolved, by date of next hearing or by CIN.
2. System prompts for additional parameters depending on the query parameter.
3. System generates a list of the matching cases.
4. Registrar may choose to view one or more cases from the list or change/reset the query parameter.

Functional Requirements

3.2.1 Browse case

Use Case Name:	Browse case
Priority	Essential
Trigger	Button selection
Precondition	User logged in as judge/lawyer, currently on the home screen
Basic Path	1. User selects the ‘Browse case’ option. 2. User enters keywords to search by. 3. The system generates a list of cases that match the keywords. 4. User selects a case to be viewed. 5. System displays details of the selected case. 6. User returns to list of cases. 7. User selects another case or returns to the home screen. 8. When the user finally returns to the home screen, if the user type is Lawyer, the system updates record of number of cases viewed by lawyer.
Alternate Path	N/A

Postcondition	User is on home screen
Exception Path	If there is a connection failure the user returns to home screen

3.2.2 Create case

Use Case Name:	Create case
Priority	Essential
Trigger	Button selection
Precondition	Registrar logged in, currently on home screen
Basic Path	1. Registrar selects ‘Create case’ option. 2. Registrar enters details of the new case and assigns a date of hearing. 3. The system generates a list of empty slots on the assigned date, if any. 4. Registrar selects a slot or assigns a new date of hearing if no slot is available, until a slot is finally selected. 5. System generates a Case Identification Number. 6. Registrar returns to home screen.
Alternate Path	N/A
Postcondition	Registrar is on home screen; new case has been created in the record
Exception Path	If there is a connection failure the user returns to home screen

3.2.3 Adjourn case

Use Case Name:	Adjourn case
Priority	Essential
Trigger	Button selection
Precondition	Registrar logged in, currently on home screen
Basic Path	1. Registrar selects a case from the list of all cases (or a particular query result) displayed on his home screen. 2. System displays details of the case along with various options. 3. Registrar selects ‘Adjourn’ option. 4. Registrar enters reason of adjournment and assign new date of hearing. 5. The system generates a list of empty

	slots on the assigned date, if any. 6. Registrar selects a slot or assigns a new date of hearing if no slot is available, until a slot is finally selected. 7. System updates case details on confirmation. 8. Registrar returns to home screen.
Alternate Path	N/A
Postcondition	Registrar is on home screen; the desired case has been adjourned
Exception Path	If there is a connection failure the user returns to home screen

3.2.4 Update case

Use Case Name:	Update case
Priority	Essential
Trigger	Button selection
Precondition	Registrar logged in, currently on home screen
Basic Path	1. Registrar selects a case from the list of all cases (or a particular query result) displayed on his home screen. 2. System displays details of the case along with various options. 3. Registrar selects 'Update' option. 4. Registrar enters summary of proceedings and assign new date of hearing. 5. The system generates a list of empty slots on the assigned date, if any. 6. Registrar selects a slot or assigns a new date of hearing if no slot is available, until a slot is finally selected. 7. System updates case details on confirmation. 8. Registrar returns to home screen.
Alternate Path	N/A
Postcondition	Registrar is on home screen; the desired case has been updated
Exception Path	If there is a connection failure the user returns to home screen

3.2.5 Close case

Use Case Name:	Close case
Priority	Essential
Trigger	Button selection
Precondition	Registrar logged in, currently in ‘update case’, has entered proceedings summary
Basic Path	1. Registrar selects ‘Close’ option. 2. Registrar enters judgment summary of the case to be closed. 3. System updates case details on confirmation. 4. Registrar returns to home screen.
Alternate Path	N/A
Postcondition	Registrar is on home screen; the desired case has been updated and closed
Exception Path	If there is a connection failure the user returns to home screen

3.2.6 Query cases

Use Case Name:	Query cases
Priority	Essential
Trigger	Button selection
Precondition	Registrar logged in, currently on home screen
Basic Path	1. Registrar selects query parameter from a list: pending, resolved, by date of next hearing or by CIN. 2. System prompts for additional parameters depending on the query parameter. 3. System generates a list of the matching cases. 4. Registrar may choose to view one or more cases from the list or change/reset the query parameter.
Alternate Path	N/A
Postcondition	Registrar is on home screen
Exception Path	If there is a connection failure the user returns to home screen

3.2.7 Create user

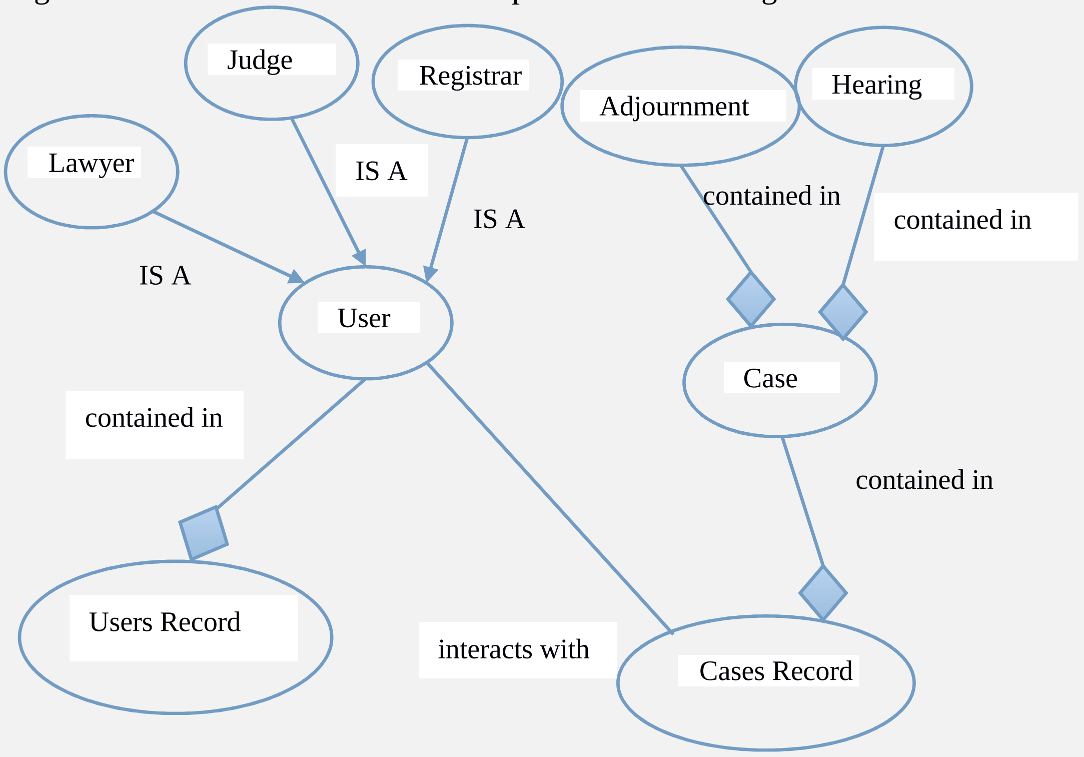
Use Case Name:	Create user
Priority	Essential
Trigger	Button selection
Precondition	Registrar logged in, currently on home screen
Basic Path	1. Registrar selects ‘Create user’ option. 2. Registrar enters user type and details. 3. System updates users record on confirmation.
Alternate Path	N/A
Postcondition	Registrar is on home screen, new user has been added
Exception Path	If there is a connection failure the user returns to home screen

3.2.8 Delete user

Use Case Name:	Delete user
Priority	Essential
Trigger	Button selection
Precondition	Registrar logged in, currently on home screen
Basic Path	1. Registrar selects ‘Delete user’ option. 2. Registrar selects user type and the user to be deleted from list of users. 3. System updates users record on confirmation.
Alternate Path	N/A
Postcondition	Registrar is on home screen, specified user has been deleted
Exception Path	If there is a connection failure the user returns to home screen

3.3 LOGICAL DATABASE REQUIREMENTS

The logical structure of the data has been represented in the diagram below.



The descriptions of the various data entities are as follows:

User Data Entity

Data Item	**Type**	**Description**	**Comment**
Username	Text	Username of the user	Assigned by Registrar
Password	Text	Password of the user	Assigned by Registrar initially
Type	Text	Type of the user	One of ‘Lawyer’, ‘Judge’, ‘Registrar’

Lawyer Data Entity

Data Item	**Type**	**Description**	**Comment**
ID	Integer	Lawyer ID	
Number of views	Integer	Number of cases viewed by the lawyer	

Judge Data Entity

Data Item	**Type**	**Description**	**Comment**
ID	Integer	Judge ID	

Registrar Data Entity

Data Item	**Type**	**Description**	**Comment**
(none)			

Adjournment Data Entity

Data Item	**Type**	**Description**	**Comment**

Original date of hearing	Date	The date on which hearing was scheduled	
Reason	Text	Reason of adjournment	

Hearing Data Entity

Data Item	**Type**	**Description**	**Comment**
Date of hearing	Date	The date of the hearing	
Summary	Text	Summary of proceedings	

Case Data Entity

Data Item	**Type**	**Description**	**Comment**
Defendant name	Text	Self explanatory	
Defendant address	Text	Self explanatory	
Type of crime	Text	Self explanatory	Murder, assault, etc
Date of crime	Date	Self explanatory	
Location of crime	Text	Self explanatory	
Arresting officer	Text	Self explanatory	
Date of arrest	Date	Self explanatory	
CIN	Integer	Self explanatory	Generated by system
Date of hearing	Date	Self explanatory	
Slot of hearing	Char	Self explanatory	A, B, C
Presiding judge	Text	Self explanatory	
Public prosecutor	Text	Self explanatory	
Starting date	Date	Self explanatory	
Expected completion date	Date	Self explanatory	May be updated
Status	Text	Self explanatory	Scheduled (Active), Closed
Judgment summary	Text	Self explanatory	Valid for closed cases
Adjournments	Adjournment[]	Record of all the times the cases is adjourned	
Hearings	Hearing[]	Record of all hearings	
Closing date	Date	Self explanatory	Valid for closed cases

3.4 SOFTWARE SYSTEM QUALITY ATTRIBUTES

3.4.1 **Portability**: Universally available operating systems such as Windows, Linux, etc should be used to make this software portable. This software is capable to adapting to different specified environments.

3.4.2 **Maintainability**: The tutorials and user's manuals provided should be thoroughly read to efficiently maintain the software. This software is capable of modifying for purpose of making corrections, improvements and adaptation.

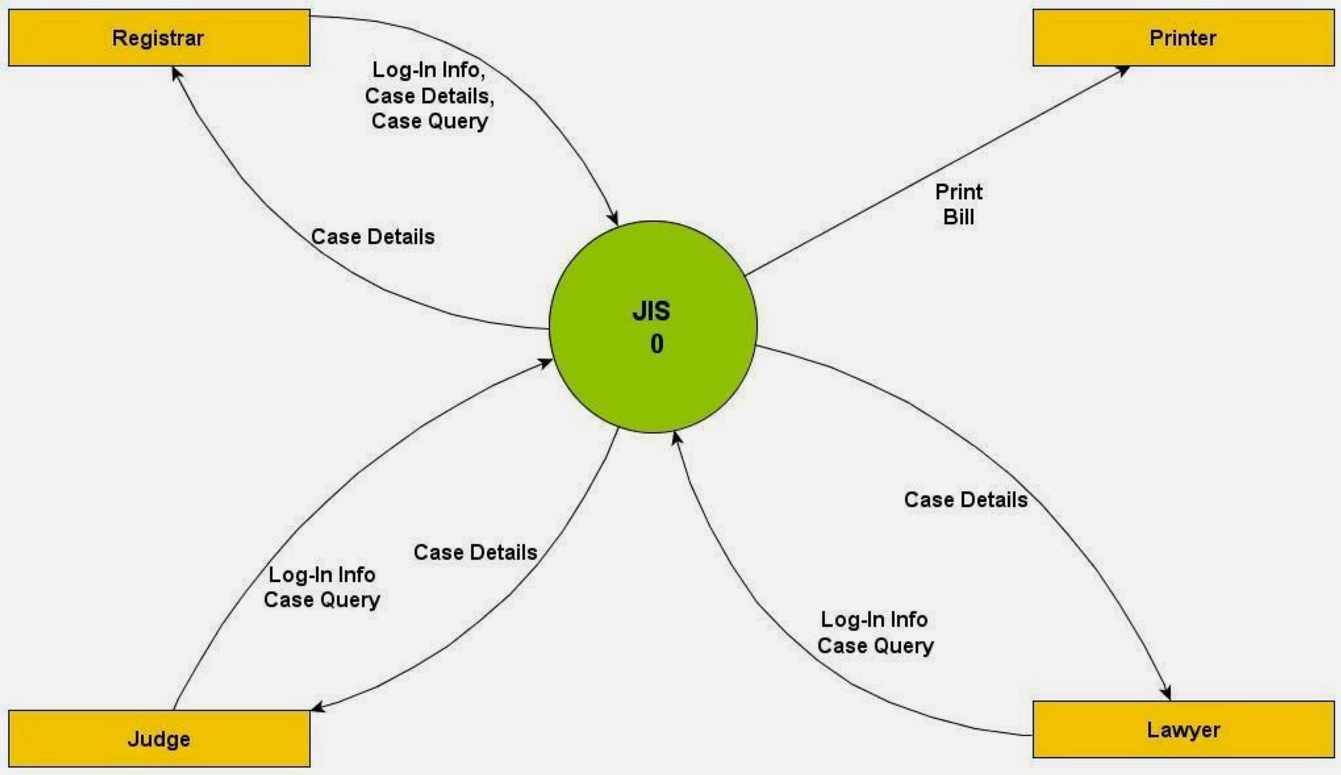
3.4.3 **Performance**: Internet connection should be available 24 hours a day for excellent performance. Performance is optimum as requirements for the given software is minimum.

Judiciary Information System (JIS)

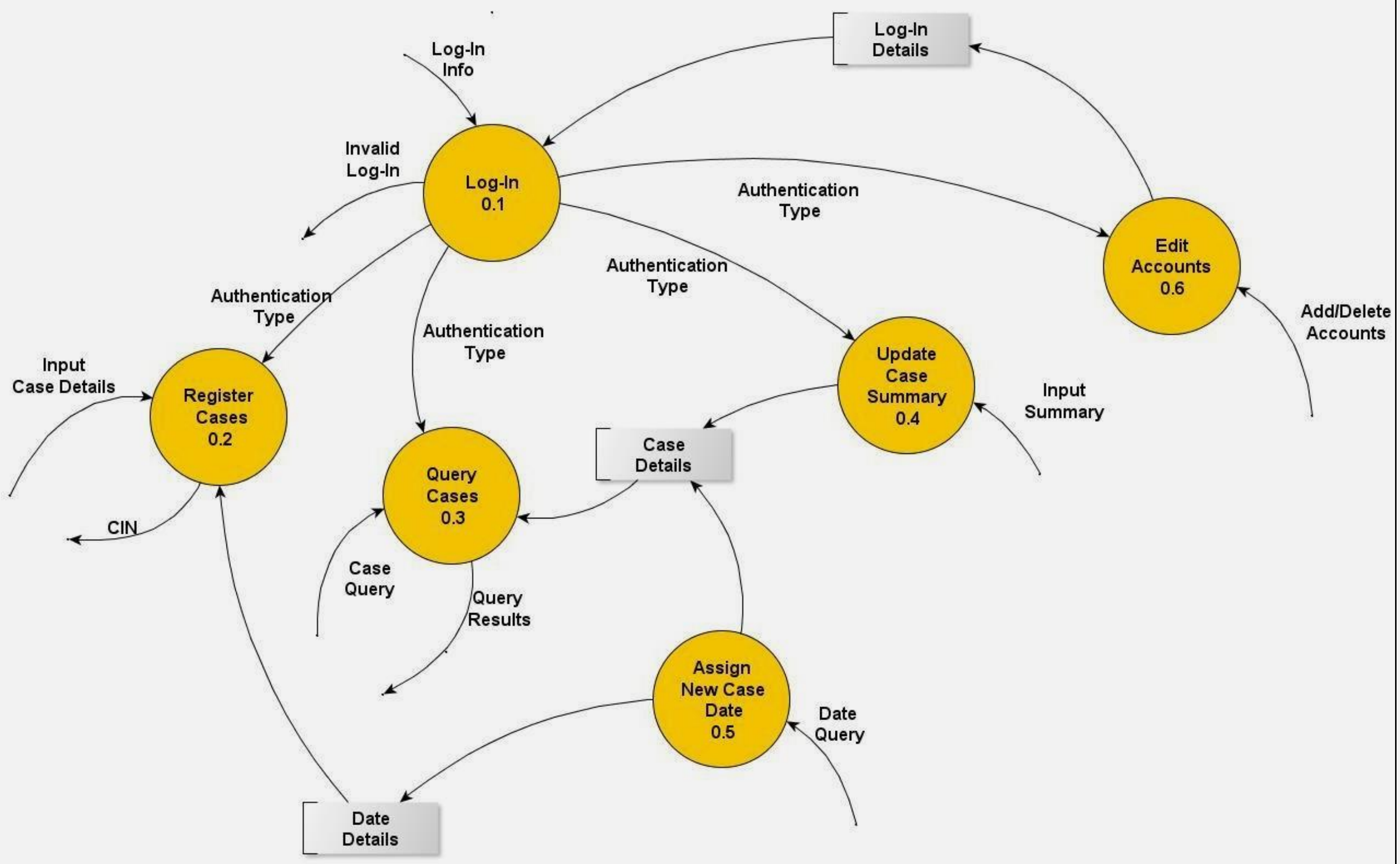
STRUCTURED ANALYSIS
AND
STRUCTURED DESIGN

119CS0100	POKALA KUSAL	GROUP 4
119CS0101	PRIYANSHU KUMAR	
119CS0102	SUSHREE SATARUPA	
119CS0103	NITIN AGARWAL	

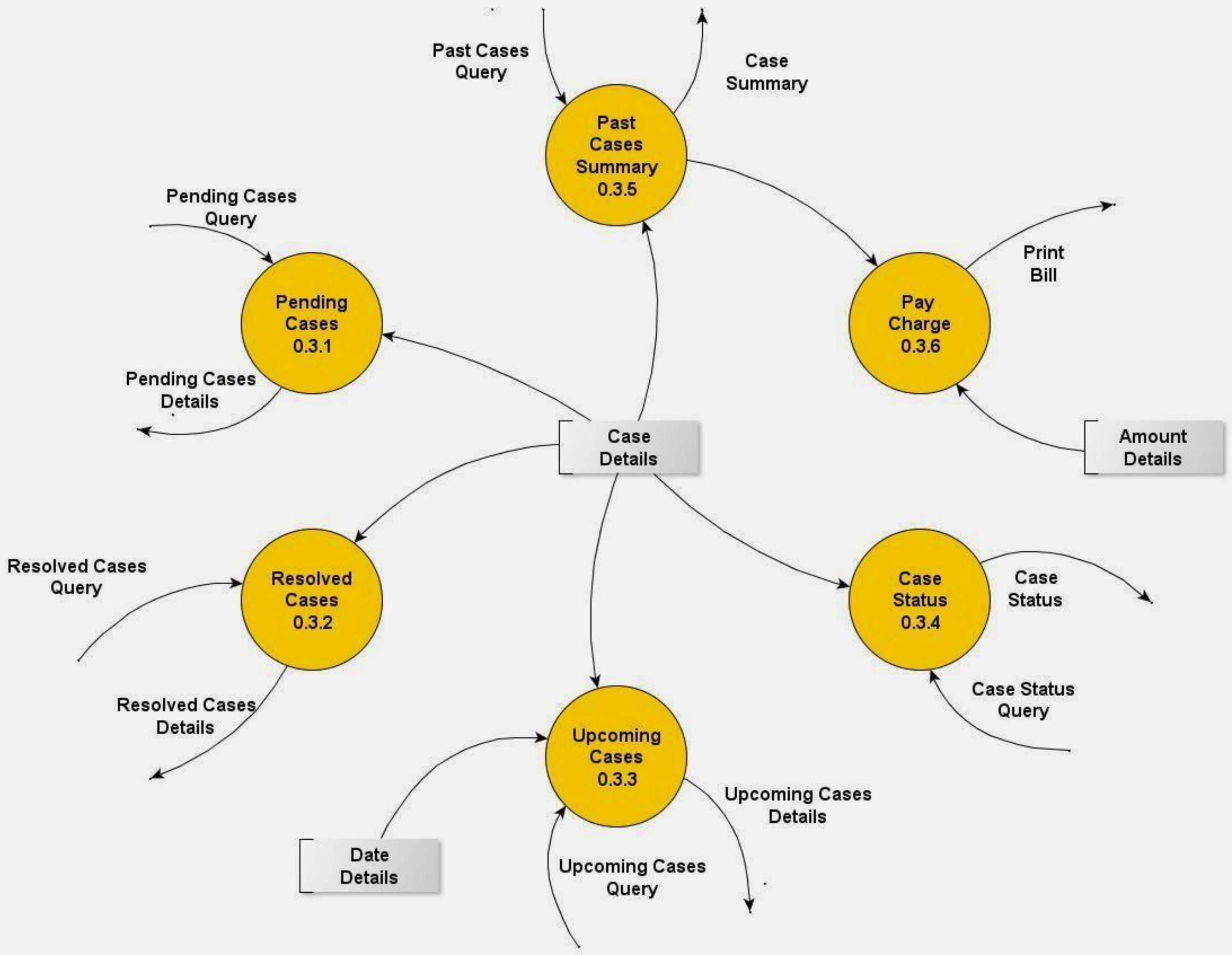
1.0. Context Diagram



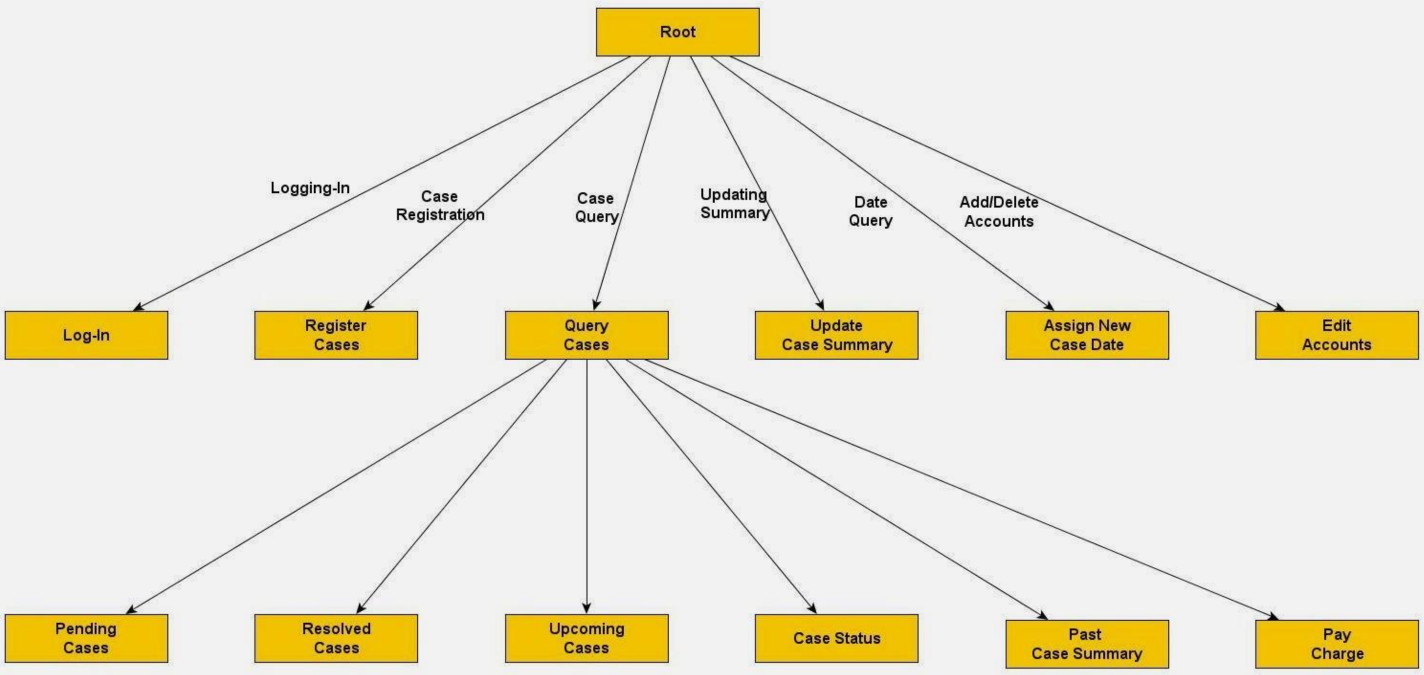
2.0. Level 1 Diagram



3.0. Level 2 Diagram



4.0. Structure Chart



5.0. Data Dictionary

- **Log-In Info:** Username + Password
- **Invalid Log-In:** Message
- **Authentication Type:** [Registrar, Judge, Lawyer]
- **Input Case Details:** Defendant’s Name + Defendant’s Address + Crime Type + Crime Date + Crime Location + Arresting Officer’s Name + Date of Arrest
- **Case Query:** [Pending Cases Query, Resolved Cases Query, Upcoming Cases Query, Case Status Query, Past Cases Query]
- **Query Results:** [Pending Cases Details, Resolved Cases Details, Upcoming Cases Details, Case Status, Case Summary]
- **Date Query:** Integer
- **Input Summary:** String
- **Pending Cases Query:** Boolean
- **Resolved Cases Query:** Boolean
- **Upcoming Cases Query:** Date **Case**
- **Status Query:** CIN
- **Past Cases Query:** CIN
- **Pending Cases Details:** Case Starting Date + Defendant’s Name + Address + Crime Details + Lawyer’s Name + Public Prosecutor’s Name + Attending Judge’s Name
- **Resolved Cases Details:** Case Starting Date + CIN + Judgement Delivering Date + Attending Judge’s Name + Judgement Summary
- **Upcoming Cases Details:** CIN
- **Case Status:** [Pending, Closed, Due] \
- **Case Summary:** CIN + String
- **Print Bill:** Lawyer’s Name + Amount Charged + Amount Left
- **CIN:** Integer
- **Lawyer’s Name:** First Name + (Middle Name) + Last Name
- **Attending Judge’s Name:** First Name + (Middle Name) + Last Name
- **Amount Charged:** Integer
- **Amount Left:** Integer
- **Date:** Day + Month + Year
- **Day:** Integer
- **Month:** Integer
- **Year:** Integer
- **Message:** String

Unified ModelingLanguage (UML)Based Design

Judiciary Information System (JIS)

119CS0100	POKALA KUSAL	GROUP 4
119CS0101	PRIYANSHU KUMAR	
119CS0102	SUSHREE SATARUPA	
119CS0103	NITIN AGARWAL	

Table of contents

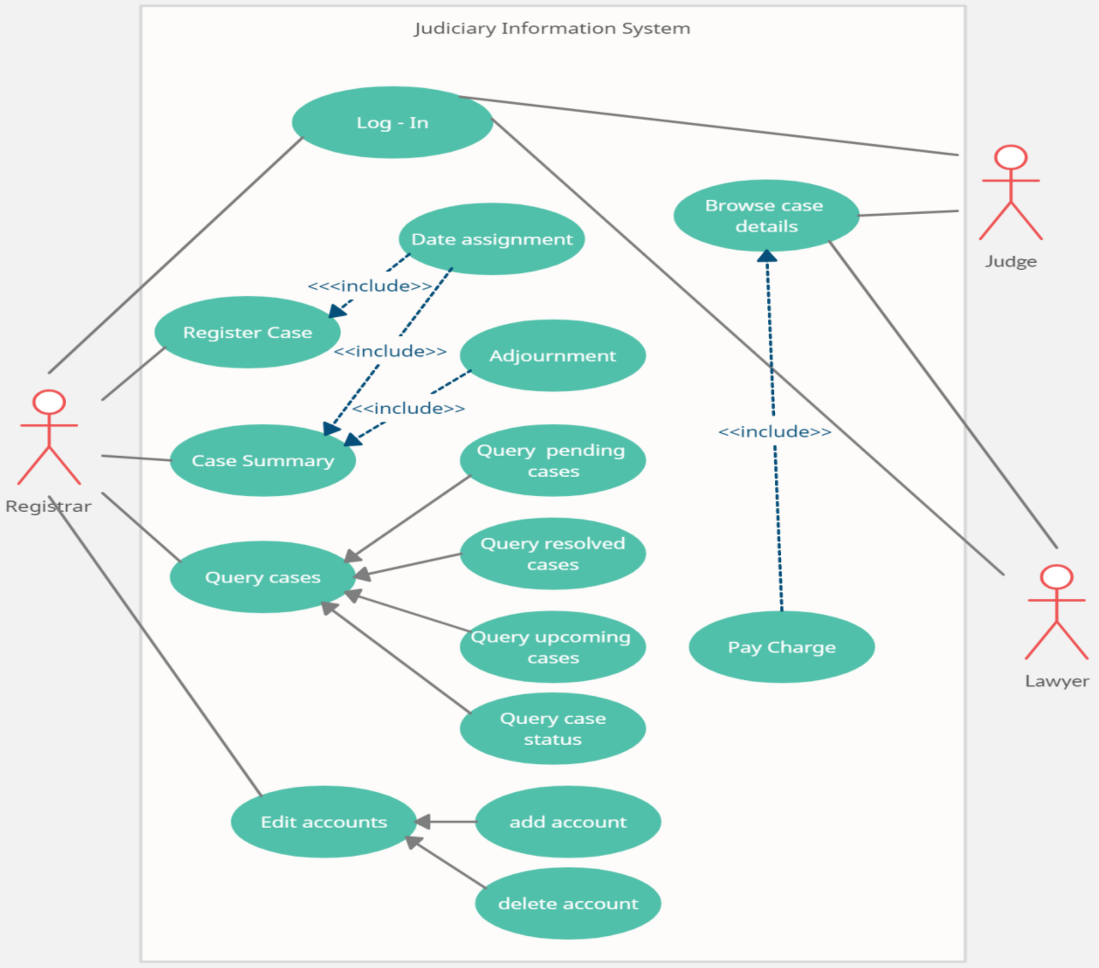
1. Introduction

This document builds upon the Software Requirements Specification and the SA/SD documents to detail a much more concrete view of the JIS to be implemented. The UML diagrams presented below have been drawn using **Umbrello**, an open-source tool available for this purpose.

NOTE: All the notations are part of the standard UML, thus they have not been documented separately.

2. Use case diagram

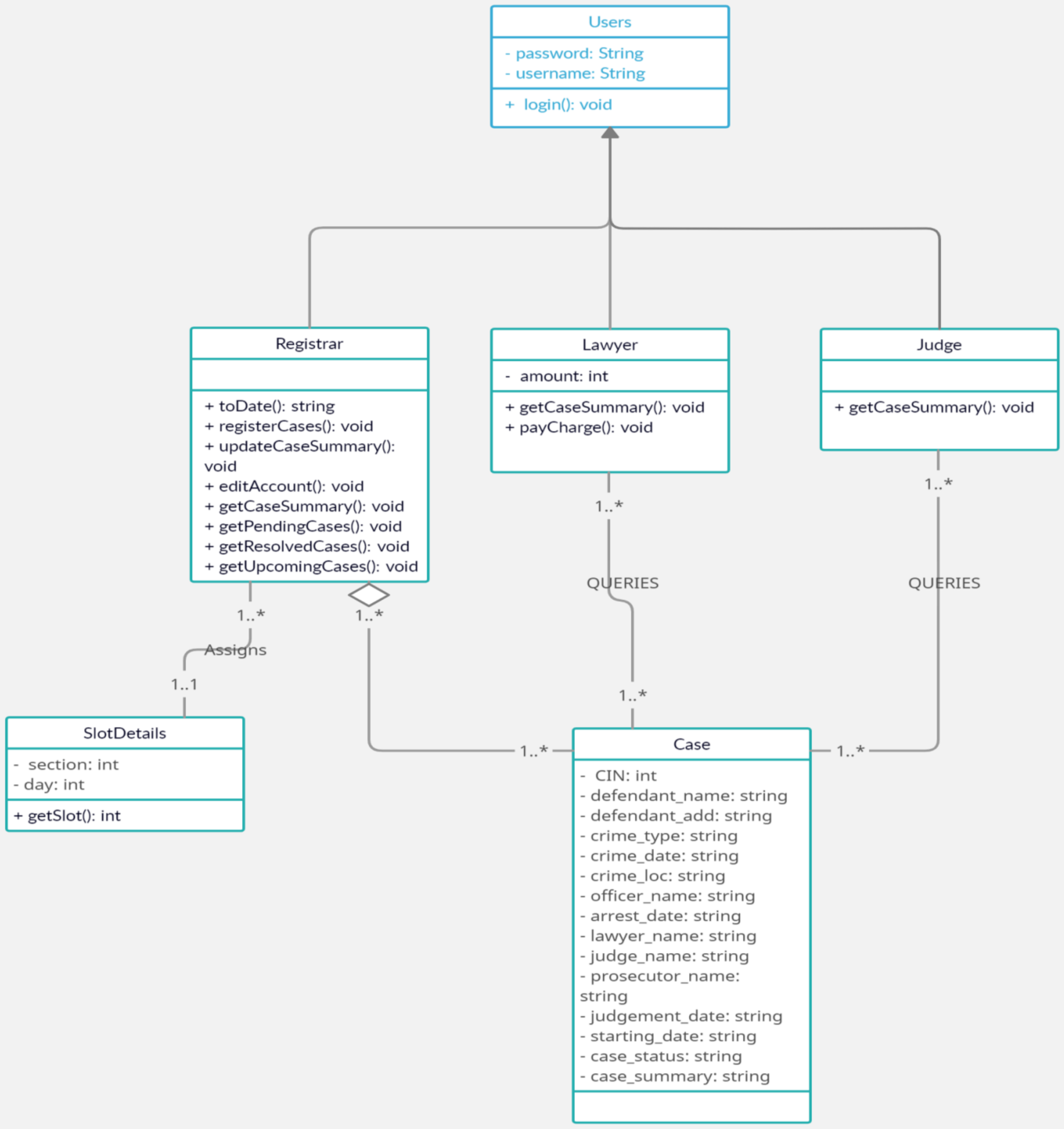
Brief description: All the three users of the system have a common use case for browsing existing cases. Apart from this, the registrar has many special use cases as shown below. Particularly, the 'close case' use case is an extension of the 'update case' use case because the registrar must have first initiated the latter use case to invoke the former. Further, the use cases for creation and deletion of user accounts are generalized versions of creation and deletion of specialized users, i.e. lawyers and judges. This relationship has been modeled appropriately.



3. Class diagrams

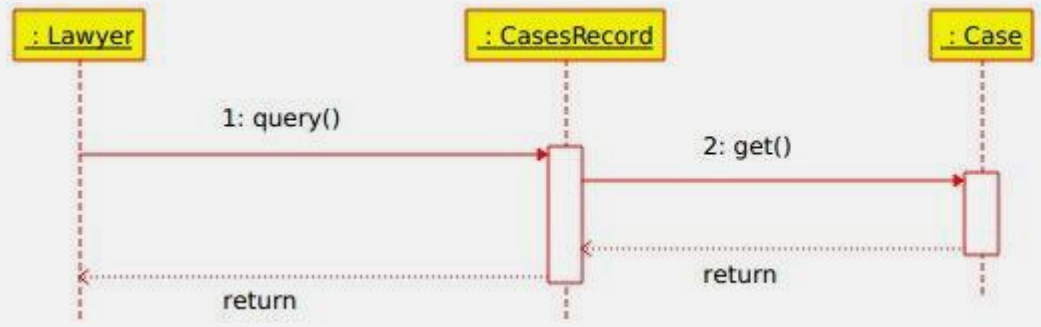
Brief description: User is a base class for all user types. Three classes are derived from the User class for the three stakeholders of the JIS. All the User objects are wrapped in an invariant container object of UsersRecord class, of which exactly instance is present in the system at all times. It contains functions for creating and deleting users. As marked in the diagram, there may be only one object of Registrar class.

All the details to be maintained for a court case are modeled as data members in the Case class. Further, objects of Adjournment and Hearing classes are used to model every update of the case. Analogous to the UsersRecord class, the CasesRecord class provides an invariant container object to wrap all the Case objects maintained in the system.



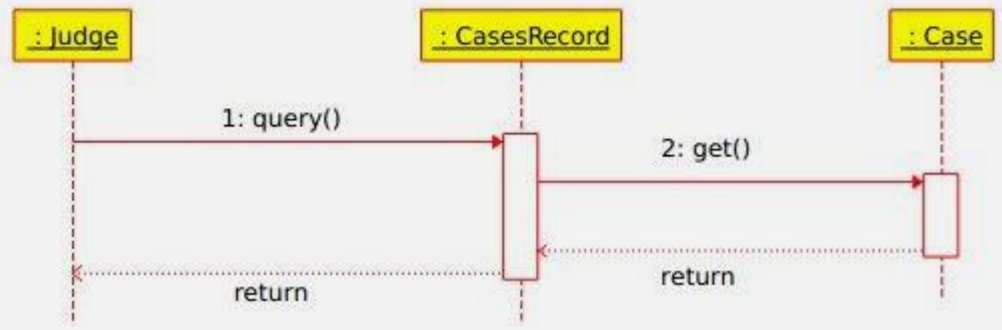
4. Sequence diagrams

4.1. Lawyer



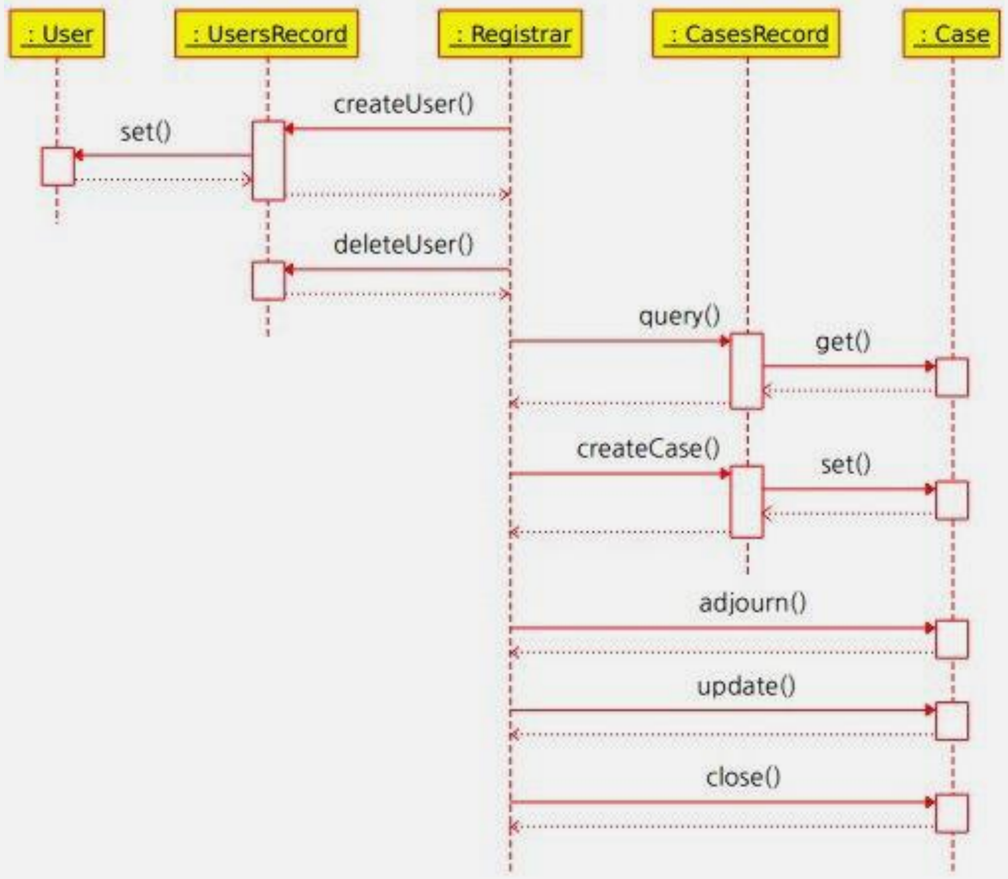
Brief description: Lawyer has only one function, i.e. browsing cases. For this purpose he sends a query containing keywords to the CasesRecord object in the system. This object then searches the data fields of all the cases to compile a list of matching cases. Here get() denotes getter functions for the various case data fields.

4.2. Judge



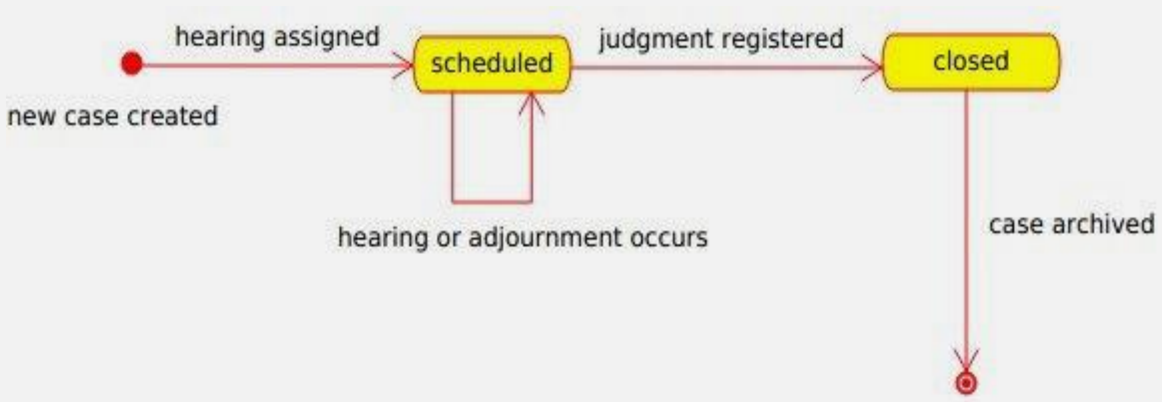
Brief description: As in the case of Lawyer, Judge also has only one function, i.e. browsing cases. For this purpose he sends a query containing keywords to the CasesRecord object in the system. This object then searches the data fields of all the cases to compile a list of matching cases. Here get() denotes getter functions for the various case data fields.

4.3. Registrar

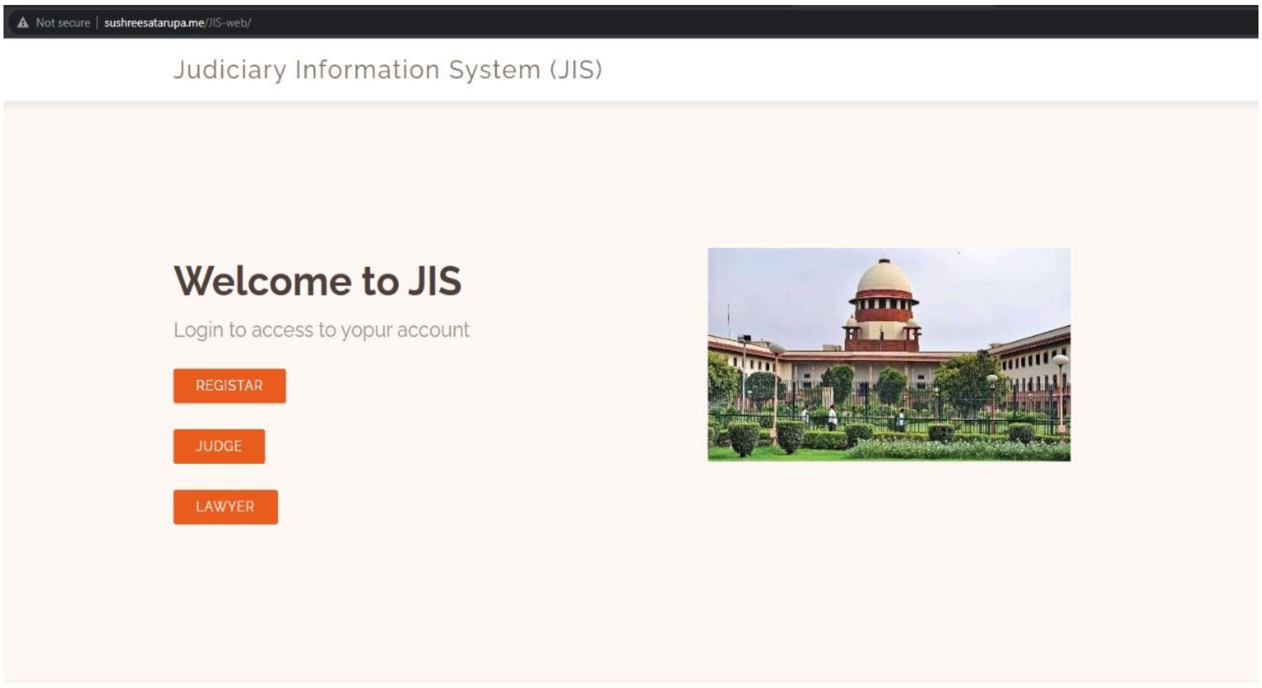


Brief description: The user management and case management capabilities of the registrar are shown on two different sides of the diagram. While querying and creating cases the registrar needs to interact with the CasesRecord object whereas for other functions like adjourn, update and close, he can directly call methods of the Case objects. The get() and set() methods represent various getter and setter functions.

5. State chart diagram



Brief description: The above is the state chart diagram for a *case* object. When a new case is registered, its details are entered by the Registrar and a date of hearing is assigned, at which stage the case attains the ‘scheduled’ state. From here it continues to be in this state through all further hearings and adjournments. When a judgment is finally registered, it moves to the ‘closed’ state permanently. Thereafter it is archived in the system and ceases to feature in the currently scheduled (active) cases.


Not secure | sushreesatarupa.me/JIS-web/

Judiciary Information System (JIS)

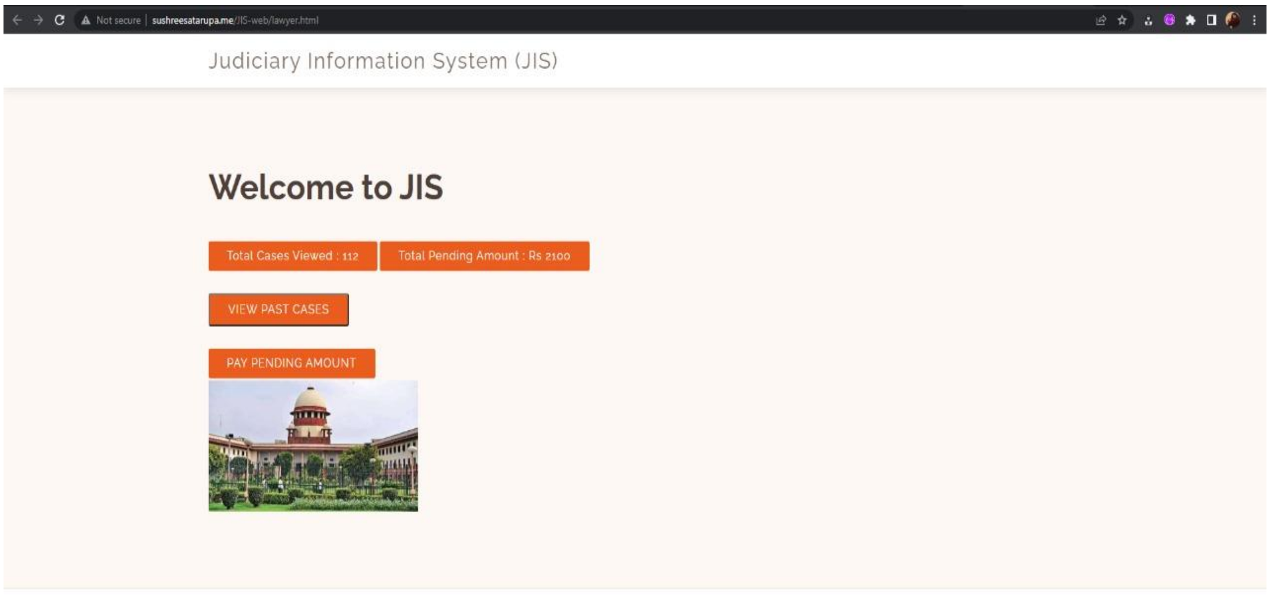
Welcome to JIS

Login to access to yopur account

REGISTAR

JUDGE

LAWYER


Not secure | sushreesatarupa.me/JIS-web/lawyer.html

Judiciary Information System (JIS)

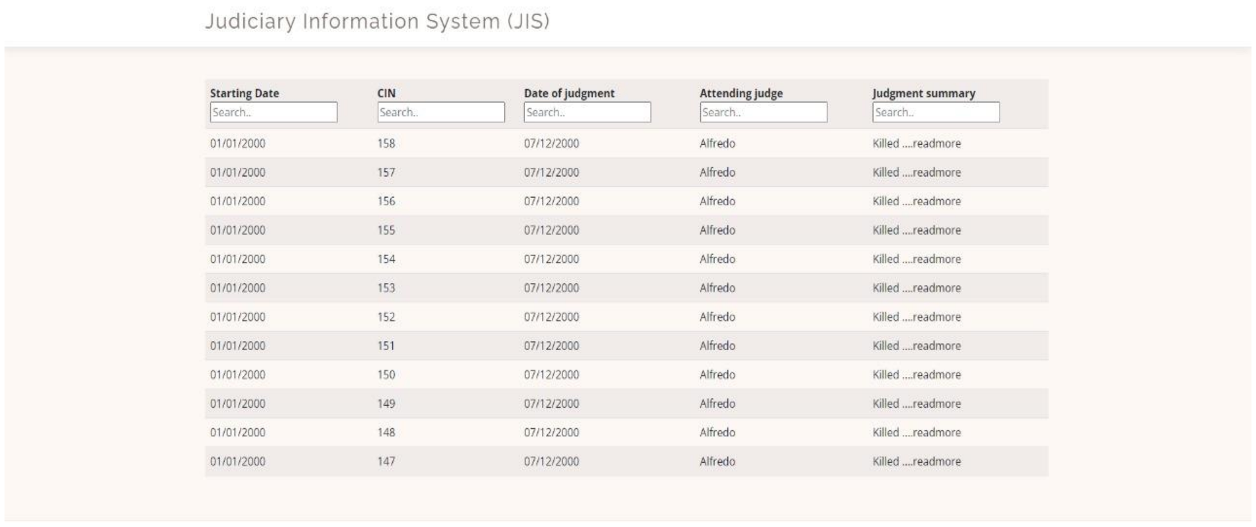
Welcome to JIS

Total Cases Viewed : 112

Total Pending Amount : Rs 2100

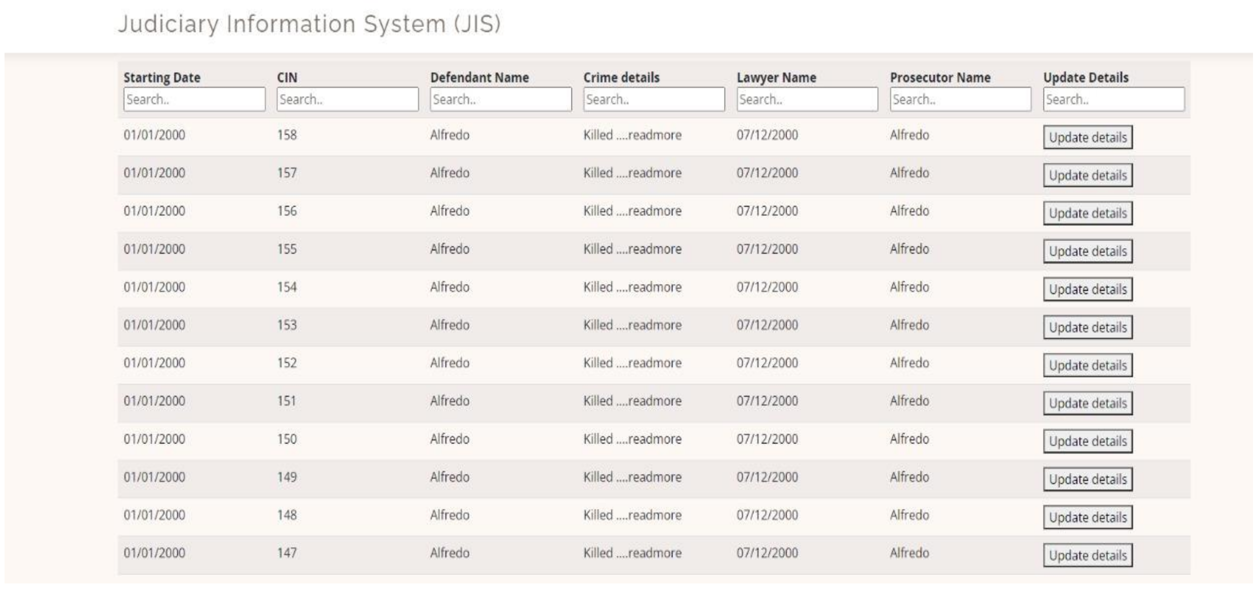
VIEW PAST CASES

PAY PENDING AMOUNT



Judiciary Information System (JIS)

Starting Date	CIN	Date of judgment	Attending judge	Judgment summary
Search..	Search..	Search..	Search..	Search..
01/01/2000	158	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	157	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	156	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	155	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	154	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	153	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	152	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	151	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	150	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	149	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	148	07/12/2000	Alfredo	Killed ....readmore
01/01/2000	147	07/12/2000	Alfredo	Killed ....readmore



Judiciary Information System (JIS)

Starting Date	CIN	Defendant Name	Crime details	Lawyer Name	Prosecutor Name	Update Details
Search..	Search..	Search..	Search..	Search..	Search..	Search..
01/01/2000	158	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	157	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	156	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	155	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	154	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	153	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	152	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	151	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	150	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	149	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	148	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details
01/01/2000	147	Alfredo	Killed ....readmore	07/12/2000	Alfredo	Update details

Code

```
#include <iostream>

#include <string>

#include <map>

#include <sstream>

using namespace std;


class Users{

public:

   string username;

   string password;


   int login(){

     cout<<"Enter the username and password:- ";

     string _username, _password;

     cin>>_username >>_password;

     cout<<endl;


     if(username==_username and password==_password){

        cout<<"Logged In successfully!!\n\n";

        return 1;

     }

     else

        cout<<"Wrong details!!\n\n";

     return 0;

    }


 };


class Case;

map<int, Case> database;
```

```
class Case{

public:

    int CIN;

    string defendant_name;

    string defendant_add;

    string crime_type;

    string crime_date;

    string crime_loc;

    string officer_name;

    string arrest_date;

    string lawyer_name;

    string judge_name;

    string prosecutor_name;

    string judgement_date;

    string starting_date;

    string case_status;

    string case_summary;


    Case(int _CIN=0, string _defendant_name="", string _defendant_add="", string _crime_type="", string
_crime_date="", string _crime_loc="", string _officer_name="",

   string _arrest_date="", string _lawyer_name="", string _judge_name="", string _prosecutor_name="", string
_starting_date="", string _case_status="", string _case_summary=""){

        CIN=_CIN;

        defendant_name=_defendant_name;

        defendant_add=_defendant_add;

        crime_type=_crime_type;

        crime_date=_crime_date;

        crime_loc=_crime_loc;

        officer_name=_officer_name;

        arrest_date=_arrest_date;

        lawyer_name=_lawyer_name;

        judge_name=_judge_name;

        prosecutor_name=_prosecutor_name;
        starting_date=_starting_date;
        case_status=_case_status;
        case_summary=_case_summary;
    }
};

class Lawyer: public Users{
public:
    int amount;
    Lawyer(){
        username = "Shubham";
        password = "0000@";
    }
    void getCaseSummary(int CIN){
        string summary = database[CIN].case_summary;
        cout<<"The case summary is:- "<<summary<<endl<<endl;
    }
    void payCharge(int t){
        amount=t*10;
        cout<<"Total Amount to be paid:- Rs. "<<endl<<endl;
    }
};

class SlotDetails{
public:
    static int section;
    static int day;
    int getSlot(){
        if(section%4==0){
            day++;
            section++;
        }
        if(section%4==1){
            cout<<"Slot is "<<day<<" days later in the morning slot"<<endl;
        }

        else if(section%4==2){

            cout<<"Slot is "<<day<<" days later in the afternoon slot"<<endl;
        }

        else if(section%4==3){
            cout<<"Slot is "<<day<<" days later in the evening slot"<<endl;
        }

        section++;
        return day;
     }
 };
int SlotDetails::section=1;
int SlotDetails::day=0;

class Registrar: public Users{
 public:
    Registrar(){
        username = "Shantanu";
        password = "1234";
    }
    string toDate(int d,string starting_date){
        int day=-1,month=-1,year=-1;
        int c=0;
        for(int i=0;i<starting_date.length();i++){
            int r=0,w=0;

            while(starting_date[i]!='/'&&i<starting_date.length()){
```

```
        r=starting_date[i];
        r-=48;
        w=w*10+r;
        i++;
    }
    if(c==0)
        day=w;
    else if(c==1)
        month=w;
    else
        year=w;
    c++;
}
if((day+d)/30==0)
    day+=d;
else if((day+d)/30<12){
    month+=(day+d)/30;
    day=(day+d)%30+1;
}
else{
    year+=(day+d)/360;
    month+=((day+d)%360)/30;
    day=((day+d)%360)%30+1;
}
string judgement_date;
ostringstream str1;
str1<<day;
judgement_date.append(str1.str());
judgement_date.append("/");
ostringstream str2;
str2<<month;
judgement_date.append(str2.str());
judgement_date.append("/");
```

```
        ostringstream str3;

        str3<<year;

        judgement_date.append(str3.str());

        return judgement_date;

    }


    void registerCases(int CIN, string defendant_name, string defendant_add, string crime_type, string
crime_date,

    string crime_loc, string officer_name, string arrest_date, string lawyer_name, string judge_name, string
prosecutor_name, string starting_date, string case_status, string case_summary){


        Case new_case(CIN, defendant_name, defendant_add, crime_type, crime_date,

    crime_loc, officer_name, arrest_date, lawyer_name, judge_name, prosecutor_name, starting_date,
case_status, case_summary);

        SlotDetails st;

        int d=st.getSlot();

        new_case.judgement_date=toDate(d,starting_date);

        database.insert({CIN, new_case});


        cout<<"Successfully Added!!\n\n";

    }


    void editAccounts(){

        cout<<"Enter 1 to add an account and 2 to delete an account:-\n";

        int op;

        cin>>op;

        cout<<endl;

        if(op==1){

                                cout<<"Enter 1 to add a judge account and 2 to add a lawyer account:-\n";

                                int ch;

                                cin>>ch;

          cout<<endl;

                                if(ch==1){

                                        cout<<"Enter the new judge's username:-\n";
                                        string new_username;

                                        cin>>new_username;

          cout<<endl;

                                        cout<<"Enter the new judge's password:-\n";

                                        string new_password;

                                        cin>>new_password;

          cout<<endl;

                                        cout<<"Successfully added!!\n\n";

                                 }

        else{

          cout<<"Enter the new lawyer's username:-\n";

          string new_username;

          cin>>new_username;

          cout<<endl;

          cout<<"Enter the new lawyer's password:-\n";

          string new_password;

          cin>>new_password;

          cout<<endl;

          cout<<"Successfully added!!\n\n";

        }

      }

        else{

          cout<<"Enter the username to be deleted:-\n";

          string new_username;

          cin>>new_username;

          cout<<endl;

          cout<<"Successfully deleted!!\n\n";

        }

  }


 void updateCaseSummary(int CIN){

    string new_summary;

    cout<<"Enter new summary:- ";
```

```
        getline(cin>>ws,new_summary);

        cout<<endl;

        database[CIN].case_summary=new_summary;

        cout<<"Enter new case status:- ";

        string new_case_status;

        cin>>new_case_status;

        cout<<endl;

        database[CIN].case_status=new_case_status;

        if(new_case_status=="Pending"){

          cout<<"Case still pending\nRescheduling\n";

          SlotDetails st;

          int d=st.getSlot();

          database[CIN].judgement_date=toDate(d,database[CIN].judgement_date);

        }

    }


    void getCaseSummary(int CIN){

        string summary = database[CIN].case_summary;

        cout<<"The case summary is:- "<<summary<<endl<<endl;

    }

    void getPendingCases(){

      for(auto it: database){

          if(it.second.case_status=="Pending")

              cout<<"Starting Date:- "<<it.second.starting_date<<" ,Judge Name:- "<<it.second.judge_name<<"
,Defendant Name:- "<<it.second.defendant_name<<" ,Defendant Address:- "<<

              it.second.defendant_add<<" ,Crime Type:- "<<it.second.crime_type<<" ,Crime Location:-
"<<it.second.crime_loc<<" ,Prosecutor Name:- "<<it.second.prosecutor_name<<" ,Lawyer Name:- "<<

              it.second.lawyer_name<<"\n\n";

      }

    }

    void getResolvedCases(){

        for(auto it: database){

          if(it.second.case_status=="Resolved")
```

```
              cout<<"Starting Date:- "<<it.second.starting_date<<" ,CIN:- "<<it.first<<" ,Judge Name:- 
"<<it.second.judge_name<<" ,Judgement Date:- "<<it.second.judgement_date<<" ,Case Summary:- "<<

              it.second.case_summary<<"\n\n";

        }

    }

    void getUpcomingCase(string input_date){

        for(auto it: database){

          if(it.second.starting_date==input_date)

              cout<<"Starting Date:- "<<it.second.starting_date<<" ,Judge Name:- "<<it.second.judge_name<<"
,Defendant Name:- "<<it.second.defendant_name<<" ,Defendant Address:- "<<

              it.second.defendant_add<<" ,Crime Type:- "<<it.second.crime_type<<" ,Crime Location:- 
"<<it.second.crime_loc<<" ,Prosecutor Name:- "<<it.second.prosecutor_name<<" ,Lawyer Name:- "<<

              it.second.lawyer_name<<"\n\n";

        }

    }

};


class Judge: public Users{

public:

    Judge(){

        username = "Himanshu";

        password = "qwerty";

    }

    void getCaseSummary(int CIN){

        string summary = database[CIN].case_summary;

        cout<<"The case summary is:- "<<summary<<endl<<endl;

    }

};


int main(){

    int start=100;

    while(1){

        cout<<"Enter 1 for Registrar, 2 for Judge, 3 for Lawyer and 4 to exit:- ";

        int choice;
    cin>>choice;

    cout<<endl;

    if(choice==1){

      Registrar registrar;


         if(registrar.login()){

           while(1){

             cout<<"Enter 1 for register new case, 2 for update case summary, 3 for query case, 4 for edit
account and 5 for exit:-\n";

             int option;

             cin>>option;

             cout<<endl;

             if(option==1){

               int CIN;

               string defendant_name;

               string defendant_add;

               string crime_type;

               string crime_date;

               string crime_loc;

               string officer_name;

               string arrest_date;

               string lawyer_name;

               string judge_name;

               string prosecutor_name;

               string starting_date;

               string case_status;

               string case_summary;


               cout<<"Enter the defendant name:- ";

               getline(cin>>ws, defendant_name);

               cout<<endl;


               cout<<"Enter the defendant address:- ";
```

```
getline(cin>>ws, defendant_add);

cout<<endl;


cout<<"Enter the type of crime:- ";

cin>>crime_type;

cout<<endl;


cout<<"Enter the date of crime(dd/mm/yyyy):- ";

cin>>crime_date;

cout<<endl;


cout<<"Enter the location of crime:- ";

getline(cin>>ws, crime_loc);

cout<<endl;


cout<<"Enter the name of officer:- ";

getline(cin>>ws, officer_name);

cout<<endl;


cout<<"Enter the date of arrest(dd/mm/yyyy):- ";

cin>>arrest_date;

cout<<endl;


cout<<"Enter the name of lawyer:- ";

getline(cin>>ws, lawyer_name);

cout<<endl;


cout<<"Enter the name of judge:- ";

getline(cin>>ws, judge_name);

cout<<endl;


cout<<"Enter the name of officer:- ";

getline(cin>>ws, officer_name);
```

```
        cout<<endl;


        cout<<"Enter the name of prosecutor:- ";
        getline(cin>>ws, prosecutor_name);
        cout<<endl;


        cout<<"Enter the starting date of case(dd/mm/yyyy):- ";
        cin>>starting_date;
        cout<<endl;


        cout<<"Enter the status of case:- ";
        cin>>case_status;
        cout<<endl;


        cout<<"Enter the summary of case:- ";
        getline(cin>>ws, case_summary);
        cout<<endl;


        CIN=start;
        start++;
        cout<<"The CIN of this case is:- "<<CIN<<endl;
        registrar.registerCases(CIN, defendant_name, defendant_add, crime_type, crime_date,
crime_loc, officer_name, arrest_date, lawyer_name, judge_name, prosecutor_name, starting_date,
case_status, case_summary);
      }


      else if(option==2){
        int CIN;
        cout<<"Enter the CIN of case:- ";
        cin>>CIN;
        cout<<endl;
        registrar.updateCaseSummary(CIN);
      }
```

```
            else if(option==3){
                cout<<"Enter 1 to view pending cases, 2 to view resolved cases, 3 to view upcoming case and 4
to get summary of a case:-\n";
                int ch;
                cin>>ch;
                cout<<endl;
                if(ch==1){
                    registrar.getPendingCases();
                }

                else if(ch==2){
                    registrar.getResolvedCases();
                }

                else if(ch==3){
                    string input_date;
                    cout<<"Enter the date:- ";
                    cin>>input_date;
                    cout<<endl;
                    registrar.getUpcomingCase(input_date);
                }

                else if(ch==4){
                    int CIN;
                    cout<<"Enter the CIN of case:- ";
                    cin>>CIN;
                    cout<<endl;
                    registrar.getCaseSummary(CIN);
                }
                else
                    cout<<"Invalid Selection\n";
            }
            else if(option==4){
                registrar.editAccounts();
            }
            else if(option==5)
                break;
            else
                cout<<"Invalid selection\n";
        }
    }
}

else if(choice==2){
    Judge judge;
    if(judge.login()){
        int CIN;
        cout<<"Enter the CIN of case:- ";
        cin>>CIN;
        cout<<endl;
        judge.getCaseSummary(CIN);
    }
 }

else if(choice==3){
    Lawyer lawyer;
    if(lawyer.login()){
        cout<<"Enter the number of case files to be viewed:- ";
        int n;
        cin>>n;
        cout<<endl;
        for(int i=1;i<=n;i++){
        int CIN;
        cout<<"Enter the CIN of case:- ";
        cin>>CIN;
        cout<<endl;
```

```
            lawyer.getCaseSummary(CIN);
            }
            lawyer.payCharge(n);
          }
        }

        else if(choice==4)
          break;

       else
          cout<<"Invalid Selection\n";
   }
   return 0;
}
```

Contributions:-

119CS0100	POKALA KUSAL	DFD, Code, Ppt
119CS0101	PRIYANSHU KUMAR	Code, UML, Report
119CS0102	SUSHREE SATARUPA	SRS, SA/SD, UI
119CS0103	NITIN AGARWAL	UML, Code, SRS