**INTERVIEW INSIGHT HUB**

# A PROJECT REPORT

***Submitted by* MRIDULA P**

**SHANJANAA G DHARSAN S**

***In partial fulfillment for the award of the degree Of***

**BACHELOR OF TECHNOLOGY**

IN

INFORMATION TECHNOLOGY



# DEPT OF INFORMATION SCIENCE AND TECHNOLOGY COLLEGE OF ENGINEERING GUINDY

**ANNA UNIVERSITY: CHENNAI 600025**

# TABLE OF CONTENTS

**CHAPTER NO TITLE PAGE NO. ACKNOWLEDGMENT**

# ABSTRACT

# TABULATED REVIEW OF TERMS USED 6

# PROBLEM HISTORY 7

# PROBLEM STATEMENT 7

# OBJECTIVES 7

# WORK TIMELINE 8

# TABULATED REVIEW OF LITERATURE 9

# DATABASE DESIGN 10

# IMPLEMENTATION 14

# LEARNING OUTCOMES 21

# CONCLUSION 26

# REFERENCES 27

**ABSTRACT**

# The project aims to address the challenge of accessing comprehensive insights into past interview experiences faced by students. Current methods are inefficient, relying on time-consuming networking and physical searches. To tackle this, we propose the development of a comprehensive online platform. This platform will enable students to easily share and access university-specific interview experiences, providing dynamic content rendering and user authentication features. By serving as a dynamic repository for detailed interview accounts, the platform facilitates peer-to-peer knowledge sharing, empowering students to prepare better for interviews and make informed career decisions.

# COMPONENTS:

**SOFTWARE USED**:

* 1. Languages used:
     1. NodeJS
     2. MongoDB
     3. HTML
     4. CSS
  2. Software used:
     1. Microsoft visual studio code- used to code in various languages used in project
     2. MongoDB: used to maintain the required database for the project

**HARDWARE USED:** No hardware was required or used for this project.

# TABULATED REVIEW OF TERMS USED

|  |  |
| --- | --- |
| Term | Definition |
| Node.JS | Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JavaScript Engine and executes JavaScript code outside a web browser,  which was designed to build scalable network applications. |
| MongoDB | MongoDB is an open-source NoSQL database management system. |
| HTML | HTML is used to create the basic structure and content of web pages. It utilizes markup tags to define elements such as headings, paragraphs, links, images, and more. |
| CSS | CSS (Cascading Style Sheets) is used to enhance the visual presentation of web content by applying styles such as colors, fonts, layouts, and animations to HTML elements, thereby shaping the overall appearance and design of web pages. |

**PROBLEM HISTORY**

* **Presently, students encounter inefficiencies when seeking past interview experiences, resorting to networking and physical searches.**
* **Existing methods prove insufficient, resulting in prolonged processes.**
* **Additionally, the absence of a streamlined source hampers students' ability to access a comprehensive repository of interview experiences specific to their college or university.**
* **There emerges a demand for an all-encompassing online platform to streamline access to interview insights.**
* **The envisioned platform will boast dynamic content rendering and user authentication features.**
* **Its objective is to act as a centralized repository for elaborate interview accounts, fostering collaborative knowledge sharing among peers.**

# PROBLEM STATEMENT

To create an application that facilitates

* Small law firms often struggle with managing their client information and case details, as well as staying organized and keeping track of their appointments and deadlines. Additionally, they may not have the resources to hire a dedicated IT staff to manage their technological needs. Our goal is to create a user-friendly and affordable website that addresses these challenges and provides small law firms with an easy-to-use platform for managing their clients, cases, and schedules. By providing a cost-effective solution, we hope to help small law firms increase their

efficiency and productivity, ultimately leading to improved client satisfaction and increased revenue.

# OBJECTIVES

There are two primary objectives to this project.

1. To create a website that manages a lawyer’s schedule along with their concerned analytics. This shall be implemented using various web development tools such as NodeJS, MySQL and ReactJS
2. To create a novel text summarizer by evaluating the existing pervasive text summarizers using ROUGE scores, thereby selecting the most effective summarizer for the legal management system

# LITERATURE SURVEY

Literature survey describes the publications that were referred in order to understand and implement the various text summarizers. Different publications and research papers were referred for different modules.

# Transformer Model:

*Vaswani et al.* introduced Transformer as a basis model for most of today's state-of-the-art NLP models. Transformer has a self-attention mechanism to focus on specific parts of the input sequence, as well as an encoder-decoder architecture with six encoder layers and the same number of decoder layers. Refer to the original study for more information.

# BART-L Model

BART is a denoising autoencoder and autoregressive decoder model proposed by *Mike Lewis et al.* It is a transformer-based model which randomly corrupts a part of the sentence by a denoising function, similar to the BERT and a learning model to predict [reconstruct the original text] the corrupted tokens like GPT with modification of ReLU to GeLUs. *Hoang et al.* proposed a model based on GPT and provided two solutions for efficiently adapting pre-trained transformer for text summarization; source embedding: which adds a source embedding to input representation to encode the token type so that the model can identify whether this token belongs to input sequence or output summary, and domain adaptive training: This helps the model to grasp the overall structure and distribution of languages before being fine- tuned on the text summarizing task.

# PEGASUS

Google researchers recently proposed an abstractive summarization approach called PEGASUS which stands for Pre-training with Extracted Gap-sentences Abstractive Summarization Sequence to sequence models. All the other previous models got out-performed by this model with better results. A goal of pre-training was proposed by them for abstractive summarization tasks. This was called Gap Sentence Generation. They used significant sentence selection as their main strategy. HugeNews and C4(Colossal and Cleaned version of Common Crawl) datasets were used. They used Gigaword, CNN/Daily Mail, Wikihow, Reddit TIFU and many more for summarization tasks. Evaluation metric used was ROUGE.

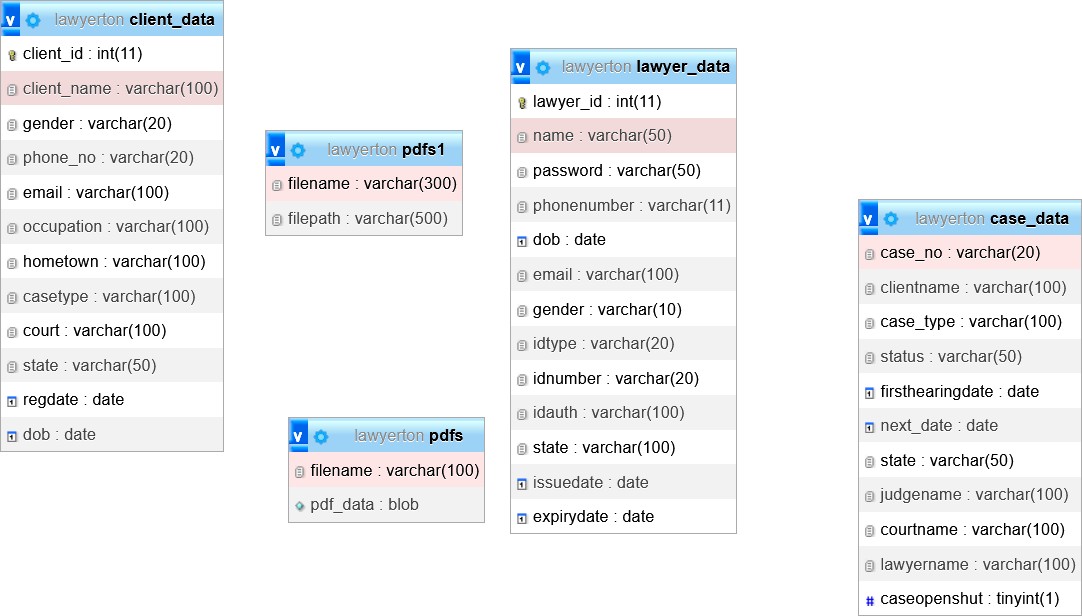
# T5 (Text to Text transfer Transformer)

TS (Transfer Learning with a Unified Text-to- Text Transformer) was also proposed by Google researchers. In NLP they use transfer learning techniques by introducing a framework which converts text-based language problems into a text-to-text format. They used a C4(Colossal Clean Crawled Corpus) as the dataset. They achieved great results on benchmarks like question answering, summarization, text classification and more.

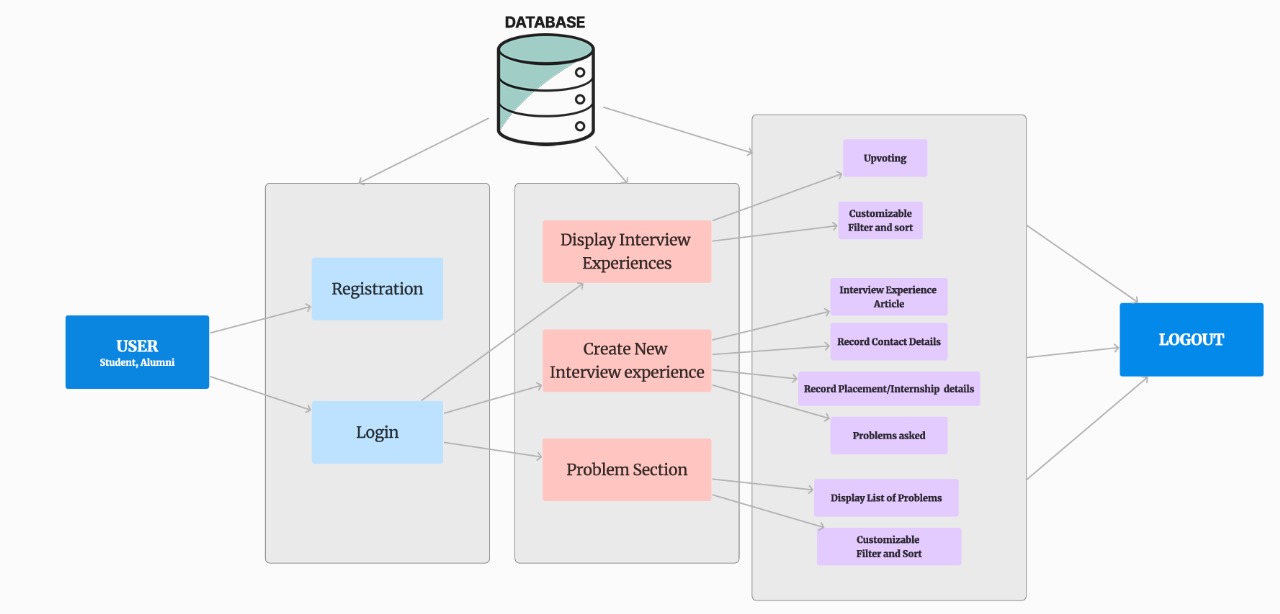
# DATABASE DESIGN

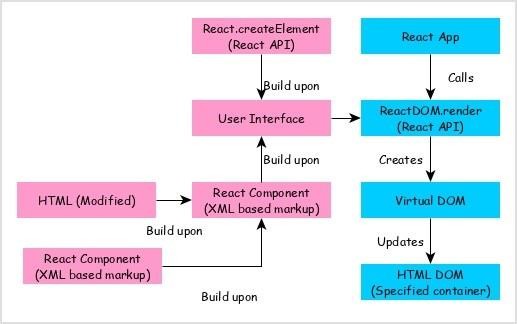
1. client data Table: The client data table stores information about the clients of the organization It includes fields such as:
   * client\_id :(PK) unique id generated for each client to easily search and retrieve information.
   * client\_name: The clients full name is stored here.
   * Gender: stores the gender of the client.
   * phone\_no: stores the phone number of the client
   * Email: stores the email of the client
   * occupation: stores the occupation of the client
   * hometown: stores the hometown of the client
   * case type: stores what type of case the client is fighting for.
   * court: stores which court the case is ongoing
   * state: stores which state the case if ongoing
   * regdate: stores the date client is registered.
   * dob: stores the date of birth of the client
2. lawyer\_data Table: The lawyer\_data table contains information about the products available in your application. It may include fields such as:
   * lawyer\_id: (PK) unique id generated for each client to easily search and retrieve information.
   * name: The lawyer’s full name is stored here.
   * password: the password required for the lawyer to login is stored here.
   * phone number: stores the phone number of the lawyer.
   * dob: stores the date of birth of the lawyer
   * email: stores the email of the lawyer
   * gender: stores the gender of the lawyer
   * idtype: stores the type of id submitted by lawyer for verification.
   * idnumber : stores the id no of the id submitted.
   * idauth: issuing authority of the ID
   * state: state on which the ID was issued
   * issuedate: date the ID was issued on.
   * Expirydate: date the ID expires on.
3. Orders Table: The Orders table stores information about the orders placed by users. It may include fields such as:
   * case\_no: (PK) unique id generated for each case to easily search and retrieve information.
   * clientname: The clients full name involved in the case is stored here.
   * case\_type: the type of case is stored here.
   * status: it stores the status as to how much the case has progressed.
   * firsthearingdate: stores the first hearing date.
   * next\_date: stores the next hearing date.
   * state: stores the state the case is running in
   * judge\_name: stores the name of the judge sitting for the case
   * courtname: stores the name of the court
   * lawyername stores the name of the lawyer in charge for the case
   * caseopenshut: stores whether case is open or shut in Boolean.
4. Pdfs1 Table: It contains the filename and filepath of the various files stored in remote server by lawyers. It may include fields such as:
   * filename: it stores the name of the file stored in server.
   * Filepath:it stores the path it is stored in the server. this is used to access the file.
5. Pdfs2 Table: It contains the filename and filepath of the various library files stored in remote server by lawyers. It may include fields such as:
   * filename: it stores the name of the file stored in server.
   * Filepath:it stores the path it is stored in the server. this is used to access the file.

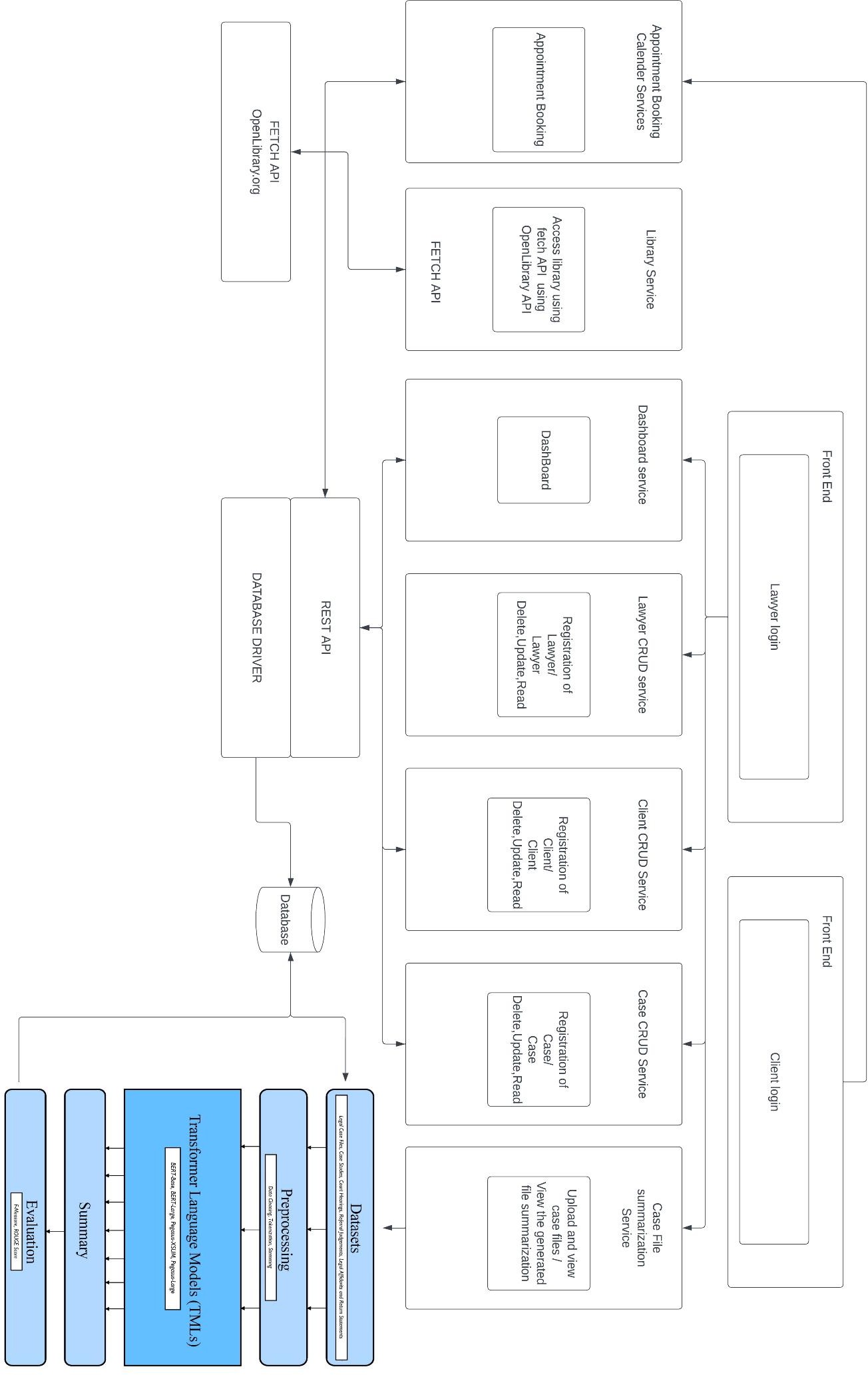
DATABASE DESIGN DIAGRAM

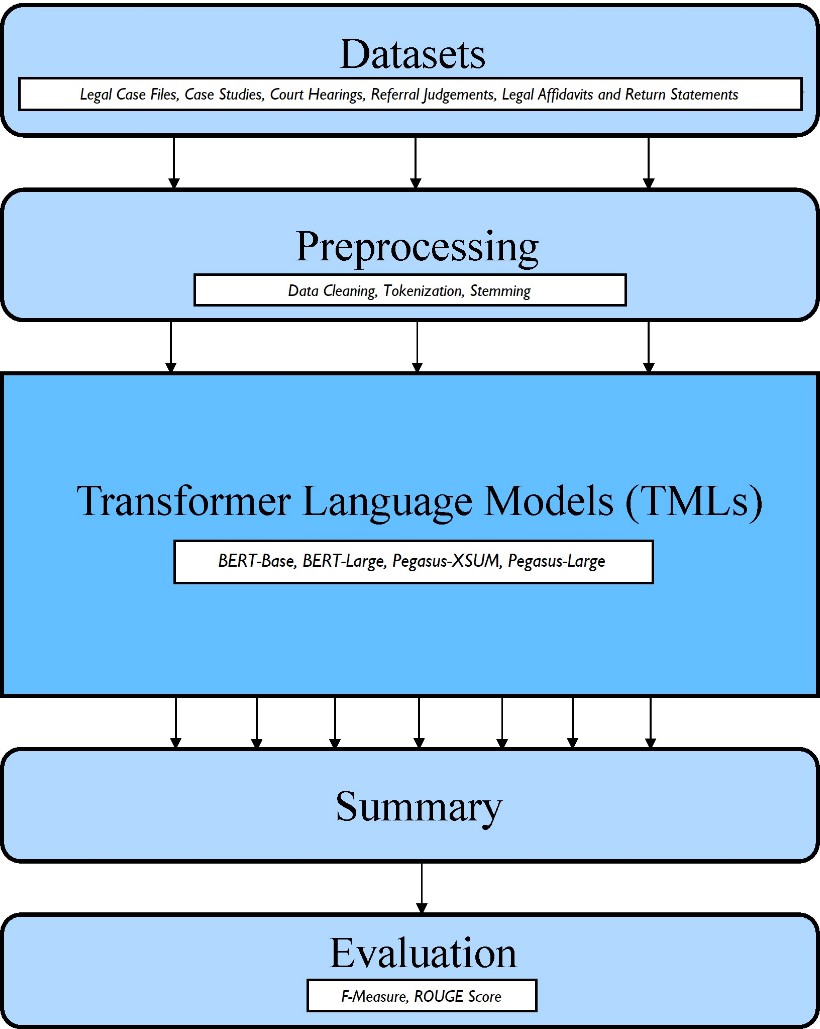


# Architecture Diagrams

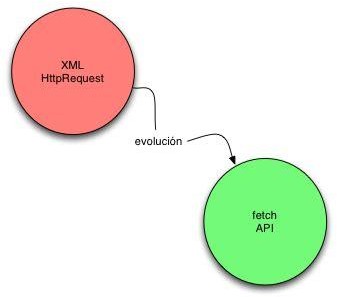
****

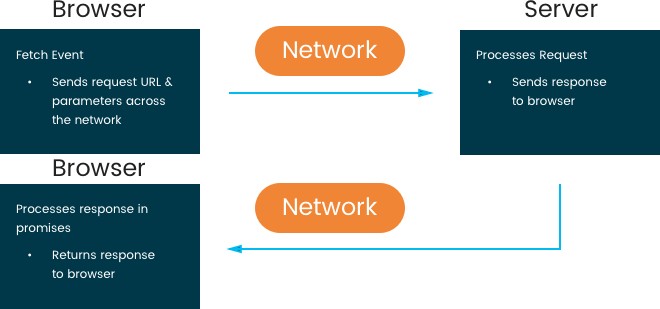
Working of react component





Evolution of fetch API





Fetch API

# Performance Evaluation



**Function List**

1. Dashboard
2. Client CRUD
3. Lawyer CRUD
4. Case CRUD
5. Library facility
6. Case summarization module
7. Login (Lawyer)

# Function description

1. **Dashboard**

The Dashboard module in a lawyer management website is responsible for providing an overview of the system's key performance indicators and data insights. The module is an essential component of any lawyer management website, as it provides a way for administrators to quickly and easily understand the current status of the system and its performance.

Dashboard Widgets: The dashboard is typically composed of a series of widgets that display data related to key performance indicators. These widgets may include graphs, charts, tables, or other visualizations that provide insights into the system's performance. Common widgets may include information about the number of lawyers registered in the system, the number of cases currently being handled, the average case resolution time, or other relevant metrics.

Dashboard Case tracker: This dashboard feature shows the status of the various active cases registered in the system along with its integral information. All this information is displayed in the form of tables and includes fields like case id, lawyer name, status of the case and customer name.

# Client crud

The Client Information CRUD module in a lawyer management website is responsible for managing information related to clients who have engaged the services of lawyers registered in the system. The module typically includes functionality to create, read, and delete (CRUD) data related to client information.

The module includes the following functionality:

1. Create Client Information: This function allows administrators to add new clients to the system by entering their details, such as name, contact information, case information, and other relevant information.
2. Read Client Information: This function allows administrators to view the details of all clients registered in the system. The module may also provide the ability to search for clients based on various criteria, such as their name or case information.
3. Delete Client Information: This function allows administrators to remove clients from the system if they are no longer active or if they have requested to be removed.

Overall, the Client Information CRUD module is critical to the smooth operation of a lawyer management website, as it allows administrators to keep track of all clients who have engaged the services of lawyers registered in the system and manage their information effectively. The module helps ensure that the lawyers have the necessary information to provide effective legal representation and helps administrators maintain accurate records of their organization's clients.

# 3. Lawyer crud

The Lawyer Information CRUD module in a lawyer management website is responsible for creating, reading and deleting (CRUD) information about lawyers registered in the system. The module is an essential component of any lawyer management website, as it provides a way for administrators to manage the data related to lawyers.

The module typically includes the following functionality:

1. Create Lawyer Information: This function allows administrators to add new lawyers to the system by entering their details, such as name, contact information, areas of practice, and other relevant information.
2. Read Lawyer Information: This function allows administrators to view the details of all lawyers registered in the system. The module may also provide the ability to search for lawyers based on various criteria, such as their name or areas of practice.
3. Delete Lawyer Information: This function allows administrators to remove lawyers from the system if they are no longer practicing or if they have requested to be removed.

Overall, the Lawyer Information CRUD module is critical to the smooth operation of a lawyer management website, as it allows administrators to keep

track of all the lawyers registered in the system and manage their information effectively.

# Case crud

The Case Information CRUD module in a lawyer management website is responsible for managing information related to cases that lawyers are handling on behalf of their clients. The module typically includes functionality to create, read, and delete (CRUD) data related to case information.

The module includes the following functionality:

1. Create Case Information: This function allows lawyers or administrators to add new cases to the system by entering details such as case name, case number, client information, opposing counsel information, and other relevant information.
2. Read Case Information: This function allows lawyers or administrators to view the details of all cases registered in the system. The module may also provide the ability to search for cases based on various criteria, such as case name, case number, or client information.
3. Delete Case Information: This function allows lawyers or administrators to remove cases from the system if they are no longer active or if they have been resolved.

Overall, the Case Information CRUD module is critical to the smooth operation of a lawyer management website, as it allows lawyers and administrators to keep track of all cases that are being handled by the system and manage case information effectively. The module helps ensure that lawyers have access to the necessary information to provide effective legal representation and helps administrators maintain accurate records of their organization's cases.

# 5. Library

The library module using OpenLibrary API in a lawyer management website is responsible for allowing access to books that can be useful for lawyers. The module typically includes functionality to search, browse, and download books related to various legal topics.

The module includes the following functionality:

1. Book Search: This function allows users to search for books related to legal topics using the OpenLibrary API. Users can enter keywords related to the topic they are interested in, and the module will display a list of relevant books.
2. Book Browsing: This function allows users to browse the collection of legal books available through the OpenLibrary API. The module may include functionality to organize books by topic, author, or other criteria.
3. Book Download: This function allows users to download legal books that are available through the OpenLibrary API. Users can choose to download books in various formats, such as PDF or ePub, and may be required to log in or create an account to access certain books.

Overall, the Library module using OpenLibrary API is critical to the smooth operation of a lawyer management website, as it allows lawyers and clients to access a wide range of legal resources and books easily. The module helps ensure that users have access to the information they need to provide effective legal representation and make informed decisions.

# 6. Login page

The Login Page module in a lawyer management website is responsible for managing the authentication process for lawyers and clients who want to access the system. The module typically includes functionality to verify user credentials, grant access to authorized users, and prevent unauthorized access to the system.

The module includes the following functionality:

1. User Authentication: This function allows users to log in to the system by entering their login credentials, such as username and password. The module will verify the user's credentials and grant access to authorized users.
2. User Authorization: This function ensures that only authorized users can access the system. The module may include functionality to manage user roles and permissions, so that lawyers can only access information related to their clients and clients can only access their own case information.

# Implementation

NLP MODULE

a.The three main stages of conducting NLP processing

1. Collect the Legal Documents Data Collection:
   * The first step in the NLP (Natural Language Processing) process is to collect the legal documents data. This involves gathering a comprehensive collection of legal documents such as judgments, case laws, legal opinions, and any other relevant legal texts. The data collection phase is crucial as it forms the foundation for training and fine-tuning NLP models to perform specific tasks related to legal text analysis.
   * The legal documents required for this project that is the case judgements and the lawyer generated summaries were acquired form local law offices in confidential manner without any specifics.
2. Finetuning Text Summarizer - BART Abstractive Text Summarizer Using Datasets of Judgment and Summaries:
   * In this step, the focus is on training and fine-tuning a text summarizer specifically tailored for legal documents. BART (Bidirectional and Autoregressive Transformer) is an abstractive text summarization model that can generate concise summaries by understanding the context and meaning of the input text. The training process involves utilizing datasets of legal judgments and their corresponding summaries to train the BART model. Through this process, the model learns to generate accurate and relevant summaries of legal texts.
   * A pre-trained abstractive text summarizer model, such as BART, and fine-tune it using the prepared dataset. Fine-tuning involves training the model on our specific task (in this case, generating summaries) using our dataset. Popular deep learning library PyTorch has been used for this purpose. Fine-tuning a model requires defining the architecture, configuring hyperparameters, and feeding the pre-processed data into the model for training. The process typically involves iterations of training epochs until the model achieves satisfactory performance.
   * After fine-tuning, performance of the summarizer model was evaluated. Evaluation metric ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is used to measure the quality of the generated summaries compared to the reference summaries. After

analysing the results, the process was iterated for fine-tuning process. Hyperparameters were adjusted to experiment with different architectures, or explore techniques like transfer learning to improve the summarization quality.

* + In this project, after the training loop, we save the fine-tuned model and the tokenizer to the specified output\_dir. 'path\_to\_output\_folder' was updated with the desired path where the fine-tuned model had to be saved.
  + After executing this , the fine-tuned model will be saved, which can be used for generating summaries or further fine-tuning if required.
  + It was ensured that the articles and summaries have the same names in their respective folders, as the code assumes that they correspond to each other based on the file names.

1. Build Website and Deploy Using Flask:
   * Once the text summarizer model is trained and fine-tuned, the next step is to build a website or web application that incorporates this NLP functionality. Flask, a popular Python web framework, can be utilized to develop the website. Flask provides a simple and flexible way to create web applications with Python. The trained text summarizer model can be integrated into the Flask application, allowing users to input legal texts and receive concise summaries generated by the NLP model. The website can be deployed using Flask's built-in server or deployed to a production server for public access.
   * Flask the web framework was used to create a website for the text summarizer.
   * How to run the flask application:
     + Web browser is opened and “http://localhost:5000” is opened to access the text summarizer website. Text is entered in the text area and the button "Summarize” is clicked to generate a summary.
     + The Flask application loads the fine-tuned model and tokenizer, creates routes for the homepage and the summarization functionality, and uses the generate\_summary() function to generate summaries based on user input. The results are then rendered in the index.html template.

b. Summary of steps for NLP module

Import necessary libraries:

* Define the SummarizationDataset class:

This class is a custom dataset class that loads and preprocesses the data.

It takes the paths to the folders containing articles and summaries, a tokenizer, and the maximum length of the input sequence as input.

The \_getitem\_ method loads an article and a summary from the corresponding files and encodes them using the tokenizer.

The encoded input, attention mask, and summary are returned as a dictionary.

Load the pre-trained summarization model:

* Specify the model name, in this case, 'facebook/bart-base'.
* Load the tokenizer and the pre-trained model using the specified model name. Move the model to the GPU if available.
* Set hyperparameters:

Set the maximum length of the input sequence, batch size, number of epochs, and learning rate.

* Load and preprocess the dataset:

Specify the paths to the folders containing articles and summaries. Create an instance of the SummarizationDataset class.

Create a dataloader to load the dataset in batches.

* Set optimizer and loss function:

Set the AdamW optimizer with the model parameters and learning rate.

* Use the cross-entropy loss function, ignoring the padding token. Training loop:

Set the model in training mode using model.train(). Iterate over each batch in the data loader.

Move the input tensors to the GPU if available. Zero the gradients of the optimizer.

Pass the inputs to the model and compute the loss and logits. Backpropagate the loss and update the model parameters.

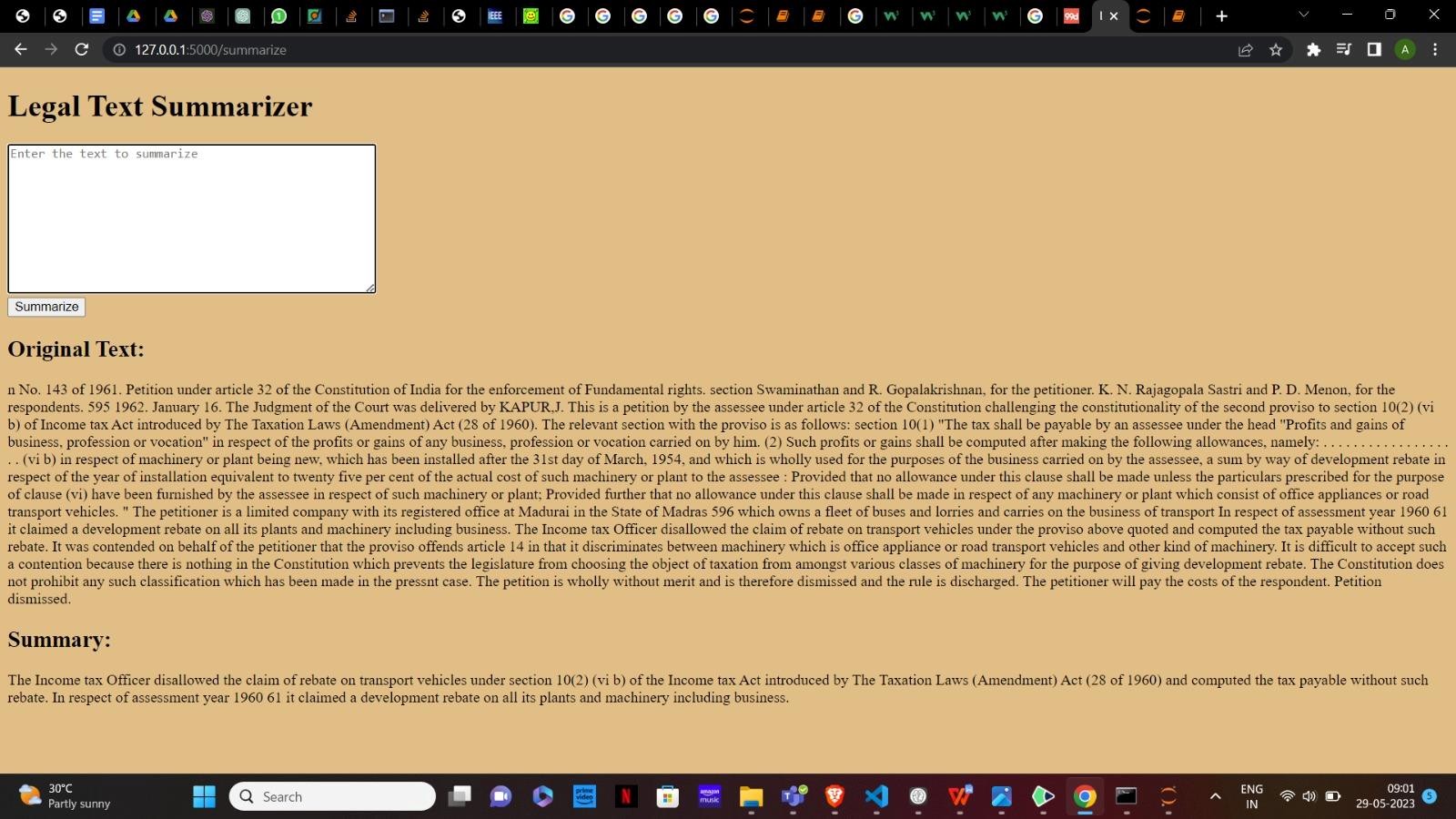
Print the loss value for monitoring.

* Save the fine-tuned model:

Specify the output directory where the fine-tuned model and tokenizer will be saved.

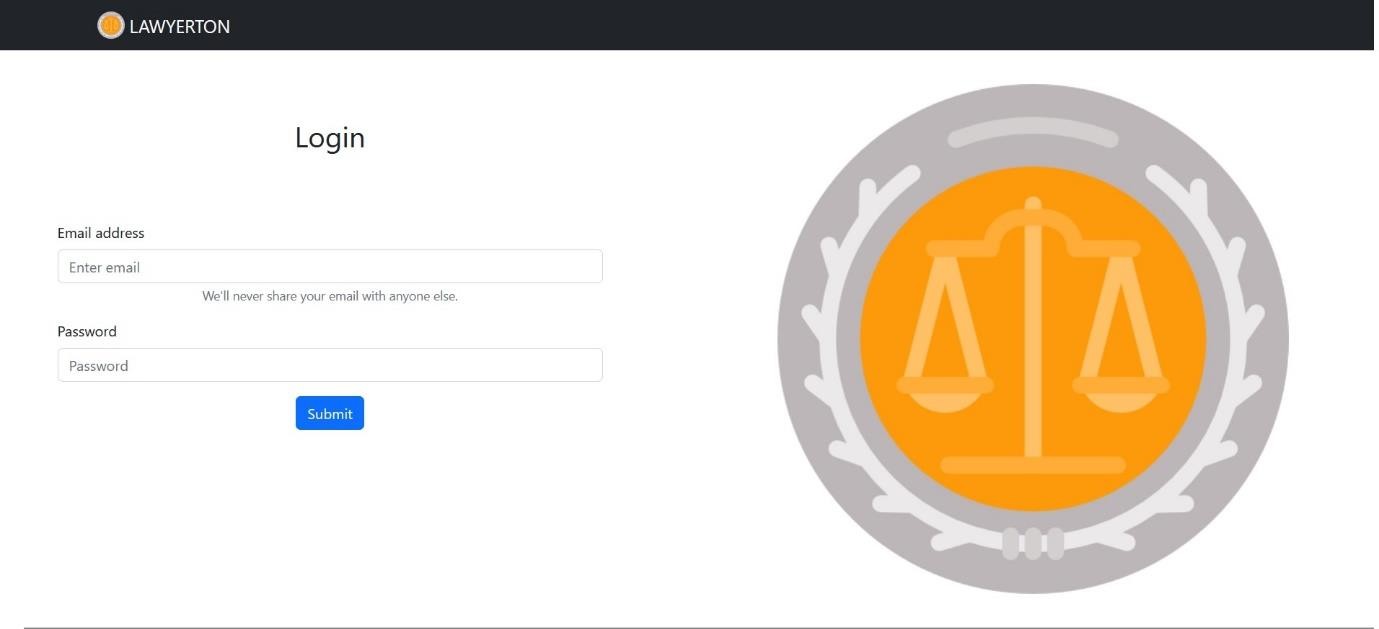
Save the model and tokenizer using the save\_pretrained method.

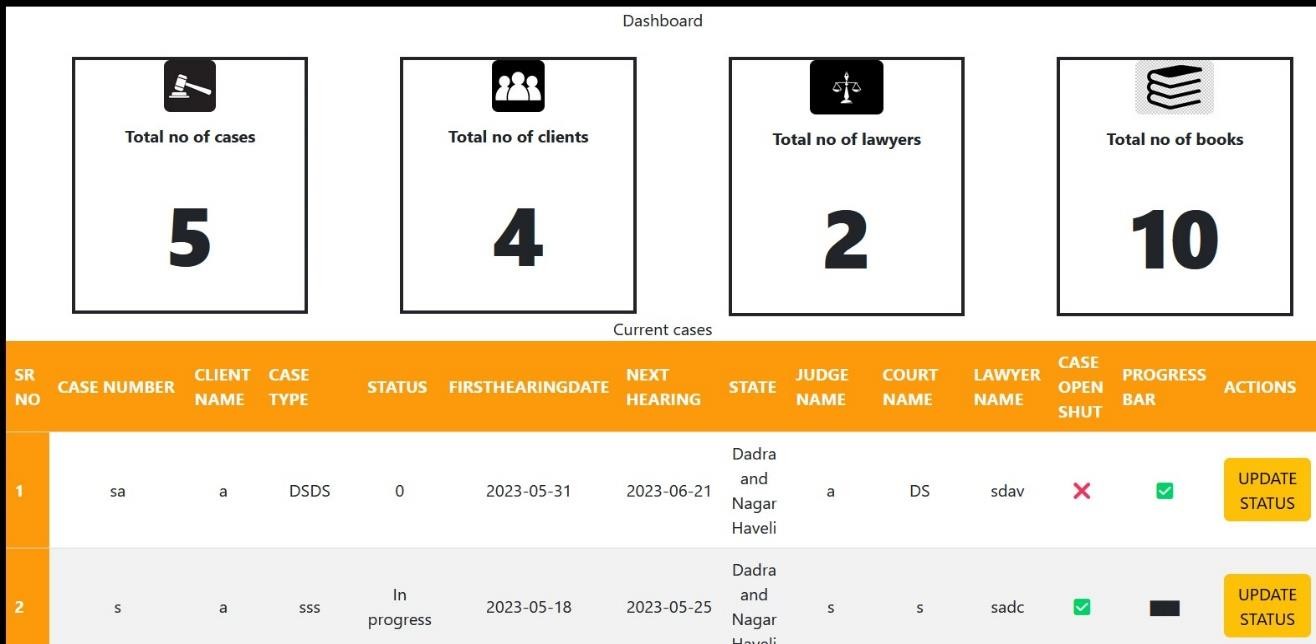
Overall, the code loads and pre-processes a dataset, fine-tunes the BART model on the dataset, and saves the trained model for future use.

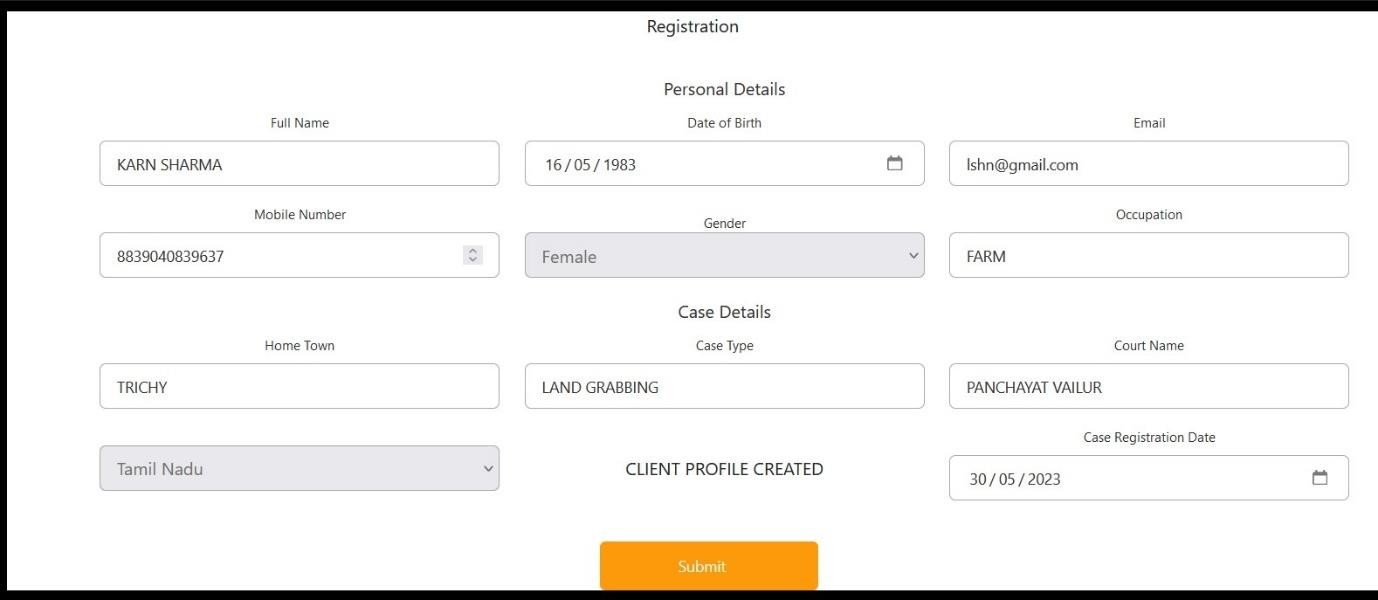


WEBSITE

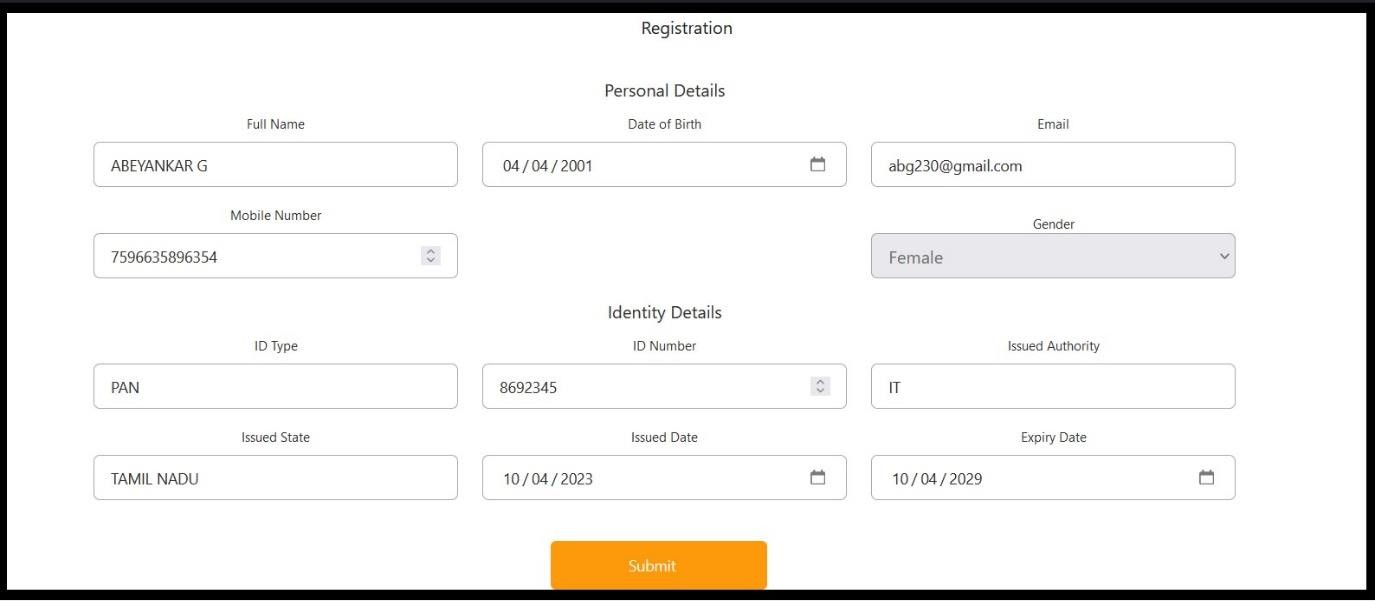
1. LOGINPAGE.JSX

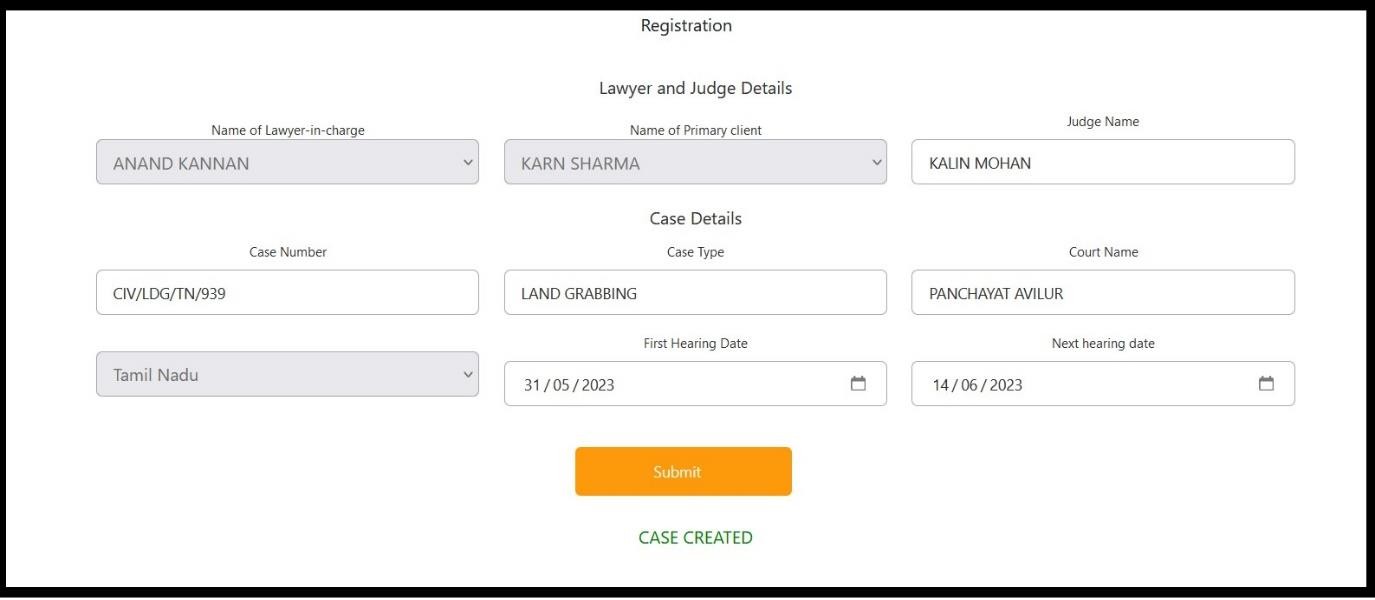
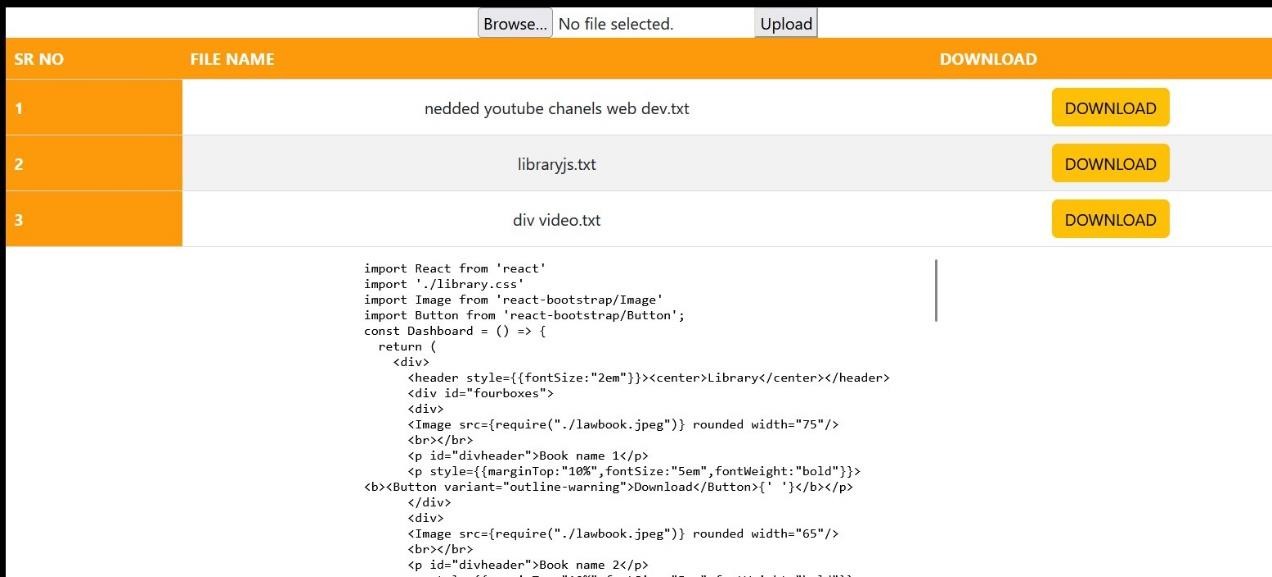


1. SIDEBAR COMPONENT
2. DASHBOARD.JSX
3. CLIENT REGISTRATION.JSX



1. LAWYER REGISTRATION.JS



1. CASE REGISTRATION
2. UPLOAD.JSX
3. CASE DETAILS AND PROGRESS.JSX



# LEARNING OUTCOMES

* + Proficiency in Full-Stack Web Development:

Developing a lawyer management project with React.js and Node.js involves working with both the front-end and back-end technologies. This project provides an opportunity to enhance your skills in full-stack web development, including designing responsive user interfaces, implementing server-side logic, and integrating different components of a web application.

* + Understanding React.js and Component-based UI Development:

By working with React.js, you will gain a deeper understanding of component-based UI development. You will learn how to create reusable and modular UI components, manage state and props, handle user input, and efficiently update the UI using Reactjs virtual DOM. This knowledge will be valuable for building interactive and scalable web applications.

* + Backend Development with Node.js:

Using Node.js as the backend technology allows you to learn about server- side JavaScript development. You will gain experience in building RESTful APIs, handling HTTP requests, implementing middleware, working with databases, and managing server-side resources. This knowledge will enable you to develop robust and efficient server-side solutions.

* + Database Management with MySQL:

Integrating MySQL as the database for the lawyer management project will give you hands-on experience in database management. You will learn how to design and implement database schemas, define tables, create relationships between entities, and perform CRUD (Create, Read, Update, Delete) operations. Understanding database management is crucial for building data- driven applications.

* + NLP Component Integration:

Incorporating a component of NLP into the lawyer management project introduces you to the field of natural language processing and its application in real-world scenarios. You will learn how to leverage NLP libraries or APIs to perform tasks like text summarization, sentiment analysis, or entity extraction. This experience will enhance your understanding of NLP concepts and how they can be used to automate or augment legal document analysis.

* + Project Management and Collaboration:

Developing a lawyer management project with multiple technologies requires effective project management and collaboration skills. You will learn how to plan and organize tasks, manage project timelines, and coordinate efforts among team members. This experience will prepare you for working in a collaborative software development environment.

* + Problem Solving and Debugging:

Throughout the project, you will encounter various challenges, including debugging issues, handling errors, and optimizing performance. By troubleshooting and finding solutions to these challenges, you will enhance your problem-solving and debugging skills, which are critical for any software development project.

# CONCLUSION

The Lawyer Management Project is a comprehensive solution designed for small law firms and individuals, incorporating technologies like React.js, Node.js, MySQL, and a component of Natural Language Processing (NLP). This project aims to streamline legal operations by providing a user-friendly web- based platform. React.js enables the development of a dynamic and responsive interface, while Node.js facilitates efficient communication between the client and server. MySQL ensures secure and efficient data storage. The integration of NLP enhances the system's capabilities, assisting with tasks like text summarization and legal document analysis. Overall, the Lawyer Management Project empowers users to efficiently manage their legal matters, enhance productivity, and improve client service.

# REFERENCES

1. React.js Documentation: The official documentation for React.js provides a comprehensive guide on building user interfaces using React.js. It covers topics such as component-based architecture, state management, and handling user interactions.

<https://reactjs.org/>

1. Node.js Documentation: The Node.js documentation is a valuable resource for understanding how to build server-side applications using Node.js. It covers topics such as setting up a server, handling requests, and working with databases.

<https://nodejs.org/en/docs/>

1. MySQL Documentation: The MySQL documentation offers extensive information on working with the MySQL database. It covers topics such as database design, querying data, and managing database operations. <https://dev.mysql.com/doc/>
2. Natural Language Processing with Python: The book "Natural Language Processing with Python" by Steven Bird, Ewan Klein, and Edward Loper is a widely recommended resource for learning the basics of NLP using the Python programming language.

<https://www.nltk.org/book/>

1. NLTK (Natural Language Toolkit): NLTK is a popular Python library for NLP tasks. It provides various functionalities for tasks such as tokenization, stemming, part-of-speech tagging, and more. The official NLTK documentation provides a comprehensive guide on using the library:

<https://www.nltk.org/documentation.html>