CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION ABOUT THE COMPANY PLACED:

I have been placed in **CHOLAMANDALAM INVESTMENT AND FINANCE COMPANY LIMITED(CIFCL)** as a part of my college campus placements and currently proceeding with a 6 months internship with them .Known for its stability in the financial sector, CIFCL has 1029 branches around the country. Remarkably, it is one of the 28 prestigious enterprises that make up the prestigious Murugappa Group. Receiving praise from the financial community, CIFCL secured the ninth spot in The Banking and Finance Post's esteemed list of India's top 50 Non-Banking Financial Companies (NBFCs).Having started out as a company that only provided finance for equipment, CIFCL has transformed into a full-service provider of financial services. Its current offers cover a wide range of financial solutions to meet the complex needs of its affluent clientele. CIFCL covers all the bases, from enabling real estate-secured loans and home loans to providing Small and Medium Businesses (SMEs) with specialised financial help. Its portfolio includes Consumer & Small Enterprise Loans (CSEL), Secured Business Personal Loans (SBPL), and a variety of other financial services and products, all of which are indicative of its dedication to comprehensive financial empowerment.

Fig 1.1 Logo

1.2 INTRODUCTION OF THE PROJECT:

In an era marked by digital transformation and increasing reliance on online services, the need for efficient and secure identity verification processes has become paramount. Aadhaar, India's unique identification system, serves as a cornerstone in this landscape, providing individuals with a universal and verifiable means of establishing their identity. However, the manual handling of Aadhaar card data poses challenges in terms of efficiency, accuracy, and data security. To address these challenges, we present an innovative solution – an Optical Character Recognition (OCR) project tailored specifically for Aadhaar cards. This project leverages cutting-edge technologies such as computer vision, machine learning, and deep learning to automate the extraction of pertinent information from Aadhaar card images captured via webcam. In Addition to it this project used Java Based various OCR technologies like EasyOCR and Tess4J for the backend OCR things to precisely get the text from the image and image preprocessing. OpenCV for Java has been used for the image preprocessing like grayscaling and image rotation. This project can work with the both straight and rotated images. The image can be in 90^0 clockwise rotated and anticlockwise rotated and even can work with 180^0 rotated images. The masking process in the aadhaar images will also be done on any image irrespective of the image orientation. By seamlessly integrating OCR capabilities into the Aadhaar card verification process, our project aims to streamline Know Your Customer (KYC) procedures, enhance data privacy, and reduce reliance on external APIs. Furthermore, the project incorporates advanced features such as the YOLOv8 model for object detection at the frontend, ensuring real-time processing and optimal user experience.

1.3 PROBLEM STATEMENT:

Aadhaar card serves as a crucial identity document in India, facilitating various governmental and private sector transactions. However, the manual extraction of information from Aadhaar cards for verification purposes is prone to errors, time-consuming, and lacks scalability. Moreover, the sensitive nature of Aadhaar numbers necessitates robust data privacy measures to prevent unauthorised access and misuse. To address these challenges, this project aims to develop an efficient and secure Aadhaar card OCR system coupled with an image masking mechanism to safeguard Aadhaar numbers.The primary objective of this project is to automate the extraction of relevant information from Aadhaar cards using Optical Character Recognition (OCR) techniques. By leveraging computer vision and machine learning algorithms, the system will accurately identify and extract key details such as the Aadhaar number, name, date of birth, and address from Aadhaar card images captured via webcam. This automation will significantly reduce manual effort and improve the efficiency of Know Your Customer (KYC) processes across various sectors, including banking, telecommunications, and government services.Additionally, the project addresses the critical issue of safeguarding Aadhaar numbers by implementing an image masking mechanism. Before sending the captured Aadhaar card image to the backend for OCR processing, the system will automatically mask the Aadhaar number region to prevent its exposure. This masking process will involve pixelating or obfuscating the Aadhaar number area, ensuring that sensitive information remains protected throughout the OCR pipeline.Furthermore, the project encompasses the development of a frontend interface capable of detecting Aadhaar cards from webcam input. Utilising state-of-the-art object detection techniques, such as the YOLOv8 model, the frontend will identify and crop Aadhaar card images from live video streams, optimizing the OCR process by focusing solely on relevant regions of interest. Overall, this project addresses the pressing need for automating Aadhaar card verification processes while prioritizing data privacy and security. By developing a comprehensive solution that combines OCR technology with image masking and frontend object detection capabilities, this project aims to revolutionize the way Aadhaar card information is handled, ensuring efficiency, accuracy, and confidentiality in identity verification procedures.

1.4 OBJECTIVE:

The objective of the OCR Project is

- To automate the extraction of relevant information from Aadhaar cards using Optical Character Recognition (OCR) techniques, thereby reducing manual effort and improving the efficiency of Know Your Customer (KYC) processes across various sectors.

- To implement an image masking mechanism to safeguard Aadhaar numbers by automatically obscuring the Aadhaar number region in captured Aadhaar card images before sending them to the backend for OCR processing.

- To develop a frontend interface capable of detecting Aadhaar cards from webcam input, utilizing state-of-the-art object detection techniques such as the YOLOv8 model.

- To enhance data privacy and security by ensuring that sensitive Aadhaar information remains protected throughout the OCR pipeline, from image capture to text extraction.

- To optimize the OCR process by focusing solely on relevant regions of interest within the Aadhaar card images, thus improving the accuracy and speed of information extraction.

- To facilitate seamless integration with existing systems and workflows by adopting a modular architecture, allowing for easy customization and scalability.

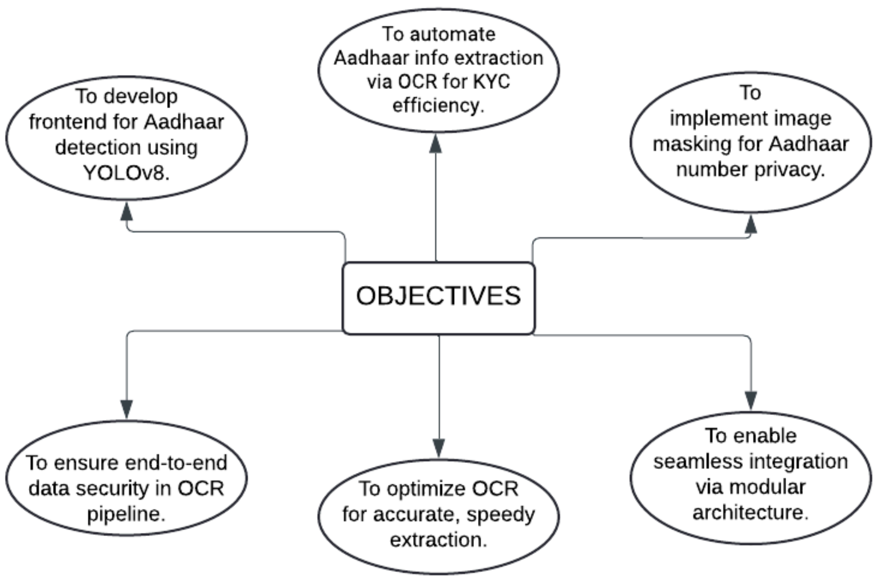

To automate Aadhaar info extraction via OCR for KYC efficiency.
To develop frontend for Aadhaar detection using YOLOv8.
To implement image masking for Aadhaar number privacy.
OBJECTIVES
To ensure end-to-end data security in OCR pipeline.
To optimize OCR for accurate, speedy extraction.
To enable seamless integration via modular architecture.

Fig 1.2 Objectives

CHAPTER 2
LITERATURE REVIEW

[1]The research paper describes the development of a Java-based application, supported by the Spring framework and libraries such as Tess4J and PDFDocument, aimed at facilitating text extraction from PDF documents through Optical Character Recognition (OCR). The application functions as a web service, receiving PDF files and returning their textual content after OCR processing. Tess4J serves as the interface to the Tesseract OCR engine, which is renowned for its versatility in handling various image formats and PDF documents. Tesseract employs a multi-stage process including image preprocessing, character segmentation, optical recognition using language-specific databases, and post-processing to refine results. Notably, Tesseract utilizes Long Short-Term Memory (LSTM) neural networks for character recognition, enabling sequential data processing and memory retention for improved accuracy. Furthermore, the application utilizes PDFDocument for PDF manipulation and metadata extraction, all within the bounds of the Apache 2.0 license. Overall, this research presents a comprehensive solution for efficient text extraction from PDF documents, contributing to advancements in document processing and accessibility.

[2]The paper focuses on addressing the challenges of designing and implementing Smart Parking Systems (SPSs) using computer vision techniques tailored for Saudi license plates. It introduces a novel multistage learning-based pipeline utilizing existing surveillance cameras within car parks and a self-collected dataset of Saudi license plates. The approach involves three stages: license plate detection, character detection, and character recognition. The authors train and fine-tune state-of-the-art YOLO series models for robust license plate and character detection, achieving high precision rates. Additionally, a new convolutional neural network architecture is proposed for

improved license plate character recognition. Experimental results demonstrate the superiority of the multistage approach over the single-stage approach, with a web-based dashboard developed for real-time monitoring and statistical analysis of car park occupancy. The paper concludes with a review of existing SPSs, detailed explanation of the proposed approach, experimental results, and future research directions.

[3] This research paper proposes a conceptual framework aimed at enhancing the security of sensitive data, particularly Aadhaar information, in the digital realm. The framework integrates the Diffie-Hellman key exchange protocol and Hash-based Message Authentication Code (HMAC) to fortify security measures. Beginning with a preprocessing phase to ensure data integrity, including noise removal and format standardization, the system segments data for efficient cloud storage. Each segment undergoes further processing to extract meaningful features while preserving relevant information and reducing unauthorized access risks. To safeguard stored Aadhaar data, the framework employs the Diffie-Hellman key exchange protocol, facilitating the establishment of a shared secret key between data owners and cloud service providers. Additionally, HMAC authentication verifies user identity during login, enhancing security through cryptographic hash functions and shared secret keys. This dual-layered approach ensures data confidentiality and integrity, bolstering the security of Aadhaar information. Empirical results underscore the efficiency of the proposed scheme, with encryption and decryption times of 0.19 s and 0.05 s, respectively.[4]This research paper addresses the automatic detection of image orientation, a crucial aspect of computer vision research with applications in various intelligent devices and software. Existing orientation detection methods often fall short in accurately expressing high-level semantics of images, leading to limited generalization ability in classification models. To overcome these limitations, the paper proposes a novel method based on the fusion of attention features (AF) and

rotation features (RF). Firstly, AF is obtained by fusing attention mechanism features extracted from different scales of ResNet50, enabling efficient screening of high-value information. Secondly, RF, which reflects directional characteristics more accurately, is obtained by residual dilated convolution combined with ResNet50. Finally, AF and RF are fused to detect four orientations of the image comprehensively. The proposed method is evaluated on five diverse datasets, demonstrating enhanced expression of directional semantics and improved classification accuracy, thereby extending the model's applicability.[5]This paper delves into the pivotal role of machine learning (ML) within software systems and highlights the existing gap in ML libraries tailored specifically for JavaScript (JS) developers. While ML libraries are predominantly geared towards Python and C++, the burgeoning JS community in both frontend and backend development underscores the pressing need for ML capabilities within the JS environment. In response to this demand, the paper introduces TensorFlow.js, a cutting-edge ML library meticulously crafted to empower JS developers and unlock novel classes of web-based applications. TensorFlow.js stands out for its seamless integration with TensorFlow, offering unparalleled high-performance ML and numeric computation capabilities within the browser and Node.js environments. The paper emphasizes TensorFlow.js's unwavering commitment to productionization, boasting high-level libraries, rigorous testing, and robust extensibility, all of which have contributed to its widespread adoption across the JS community. Furthermore, the paper delves into the intricacies of the JS environment, elucidates the design intricacies of the library and its APIs, and outlines the innovative use cases facilitated by TensorFlow.js, showcasing its transformative potential in spearheading innovation within web-based ML applications.

CHAPTER 3
SOFTWARE REQUIREMENTS

3.1.SOFTWARE SPECIFICATIONS

To meet the project's software requirements, a comprehensive tech stack has been assembled, tailored to ensure efficiency and functionality across both backend and frontend development. Java serves as the backbone for backend operations, supported by the Spring Boot framework, renowned for its ease of use and rapid development capabilities. On the frontend, HTML, CSS, and JavaScript form the foundation, enabling the creation of dynamic and interactive user interfaces. The integration of YOLOv8 for model creation underscores the project's commitment to cutting-edge machine learning techniques, while TensorFlow.js facilitates seamless deployment of these models directly within the frontend environment. Additionally, for Optical Character Recognition (OCR) tasks, both EasyOCR and Tess4jOCR are utilized, leveraging their respective strengths for accurate text extraction. Furthermore, OpenCV is employed for backend image preprocessing, ensuring that images are appropriately prepped for subsequent analysis and processing. This comprehensive software ecosystem, blending backend stability with frontend versatility and incorporating state-of-the-art machine learning and image processing tools, positions the project to deliver robust and efficient solutions for its intended purposes.This comprehensive software ecosystem, comprising a diverse range of backend and frontend technologies, machine learning frameworks, and image processing libraries, underscores the project's commitment to delivering robust, efficient, and feature-rich solutions. By harnessing the collective power of these tools, the project is poised to address complex challenges and unlock new opportunities in the realm of image processing, machine learning, and user interaction.

Fig 3.1 FRONTEND TECHSTACK

Fig 3.2 BACKEND TECHSTACK

Technologies Used:

- Frontend:

HTML,CSS,JAVASCRIPT:

In our project, HTML, CSS, and JavaScript are the foundational technologies driving the frontend development. HTML structures the content and layout of web pages, CSS styles and enhances their appearance, while JavaScript adds interactivity and functionality, enabling dynamic user experiences. This trio of technologies synergizes to create engaging and visually appealing interfaces, ensuring a seamless and intuitive user journey through our web application.

Fig 3.3 HTML,CSS,JAVASCRIPT

- Backend: **Java:**

  Java, renowned for its versatility and reliability, serves as a linchpin in modern software development. Its platform independence, object-oriented paradigm, and extensive ecosystem of libraries and frameworks make it a top choice for a wide range of applications. In our project, Java acts as the backbone for backend operations, leveraging its scalability and robustness to handle critical tasks such as image preprocessing and OCR (Optical Character Recognition). With its vast community support and mature development tools, Java enables efficient and seamless implementation of complex functionalities, ensuring the stability and performance of our application.In addition to its inherent strengths, Java's compatibility with various libraries and packages further enhances its appeal. One notable example is the Tess4j package, a Java wrapper for the Tesseract OCR engine. Tess4j provides developers with a convenient and reliable tool for text extraction from images, seamlessly integrated into Java applications. Similarly, EasyOCR, a lightweight OCR library for Java, offers straightforward integration and powerful OCR capabilities, further enriching the developer experience. By leveraging these libraries, Java developers can streamline the OCR process, improve accuracy, and enhance the functionality of their applications. This interoperability and ease of integration reinforce Java's position as a preferred language for projects requiring sophisticated OCR functionalities.

Fig 3.4 Java

- Framework:**Spring Boot:**

Spring Boot, a powerful and lightweight framework built on top of the Spring framework, revolutionizes Java-based application development. It provides developers with a streamlined approach to building production-ready applications, eliminating the need for manual configuration and boilerplate code. With its convention-over-configuration philosophy, Spring Boot simplifies the setup of Spring-based applications, allowing developers to focus on writing business logic rather than infrastructure concerns. Its embedded application server and auto-configuration capabilities further accelerate development cycles, enabling rapid prototyping and deployment. Additionally, Spring Boot offers a rich ecosystem of starter projects, enabling developers to quickly bootstrap their applications with essential dependencies and features. Overall, Spring Boot empowers developers to create robust, scalable, and maintainable Java applications with ease, making it a preferred choice for building modern, enterprise-grade software solutions.

Fig 3.5 SpringBoot

- OCR Packages: **EasyOCR,Tess4J:**

EasyOCR:

EasyOCR, a lightweight OCR library for Java, simplifies the process of text extraction from images, offering developers a convenient solution for integrating OCR capabilities into their Java applications. In our project, EasyOCR plays a pivotal role in extracting essential information such as names, Aadhaar numbers, and other pertinent details from Aadhaar images. Leveraging EasyOCR's intuitive API, developers can effortlessly incorporate text recognition functionality into their Java applications, eliminating the need for complex OCR algorithms or manual text extraction processes. By utilizing EasyOCR, we ensure accurate and reliable extraction of text from Aadhaar images, enhancing the efficiency and usability of our application. Moreover, EasyOCR's seamless integration with Java makes it an ideal choice for projects requiring text extraction capabilities, underscoring its versatility and utility in the Java development ecosystem.

Fig 3.6 easyOCR

TessOCR(Tess4J):

Tess4J is a Java library that serves as a wrapper for the Tesseract OCR engine, allowing developers to integrate powerful Optical Character Recognition (OCR) capabilities into their Java applications. Tesseract, developed by Google, is one of the most widely used OCR engines, renowned for its accuracy and versatility in extracting text from images. Tess4J simplifies the process of utilising Tesseract within Java applications, providing a user-friendly interface for text extraction tasks. With Tess4J, developers can seamlessly incorporate OCR functionalities into their Java projects, enabling tasks such as text recognition, extraction, and analysis with ease.

In our project, we leverage Tess4J to precisely locate and extract the Aadhaar number from Aadhaar card images. By utilising Tess4J's capabilities, we efficiently determine the coordinates of the Aadhaar number within the image, facilitating the subsequent masking process. This involves obscuring the Aadhaar number region within the image to protect sensitive information while preserving the integrity of the document. Tess4J's robust OCR capabilities ensure accurate recognition of text, enabling us to pinpoint the exact location of the Aadhaar number for masking purposes. Through the seamless integration of Tess4J into our Java application, we enhance data privacy and security by safeguarding sensitive information, demonstrating Tess4J's invaluable contribution to our project.

Tess4J

Fig 3.7 Tess4J

OpenCV:

OpenCV, a powerful computer vision library, offers extensive functionalities for image processing, analysis, and manipulation in the Java programming language. With its comprehensive set of tools and algorithms, OpenCV enables developers to perform a wide range of tasks, including object detection, facial recognition, and image enhancement, among others. Its intuitive API and rich documentation make it a popular choice for Java developers seeking to incorporate computer vision capabilities into their applications. OpenCV's cross-platform compatibility and robust performance ensure seamless integration and efficient processing of images in Java projects.

In our project, we harness the capabilities of OpenCV for image rotation and preprocessing tasks. By utilizing OpenCV's rotation functions, we efficiently rotate input images to the correct orientation, ensuring consistency and accuracy in subsequent processing steps. Additionally, OpenCV facilitates image preprocessing tasks such as grayscale conversion, which enhances the quality and clarity of images for further analysis.

Fig 3.8 OpenCV

YoloV8:

YOLOv8, short for "You Only Look Once version 8," is a state-of-the-art object detection algorithm known for its speed and accuracy. It belongs to the family of YOLO models, which utilize a single neural network to simultaneously predict bounding boxes and class probabilities for multiple objects within an image. YOLOv8 builds upon its predecessors by incorporating advanced techniques such as feature pyramid networks, focal loss, and spatial attention mechanisms to improve detection performance across various object scales and categories. In our project, we leverage the YOLOv8 model to detect Aadhaar cards in real-time using the webcam input. Upon detection, the Aadhaar card is automatically cropped from the webcam feed, providing users with a seamless and efficient experience. By harnessing the power of YOLOv8, we ensure accurate and reliable detection of Aadhaar cards, enabling users to quickly and effortlessly interact with our application without the need for manual intervention. This integration of YOLOv8 significantly enhances the functionality and usability of our system, facilitating smooth and intuitive interaction for users as they engage with Aadhaar card detection and processing tasks.

Fig 3.9 YOLOv8

Tensorflow JS:

TensorFlow.js is a cutting-edge library that allows developers to deploy machine learning models directly in the web browser or Node.js environments. It enables seamless integration of machine learning functionalities into web applications, offering capabilities for training, inference, and data processing entirely within the frontend environment. With TensorFlow.js, developers can leverage pre-trained models or train custom models using JavaScript, empowering them to create innovative and interactive machine learning-powered applications directly within the web browser.

In our project, the seamless integration of TensorFlow.js into our frontend development workflow has revolutionized the way we deploy machine learning models within web applications. By embedding our YOLOv8 model directly into the JavaScript codebase, we've unlocked the ability to perform real-time object detection of Aadhaar cards directly within the user's web browser. This groundbreaking approach not only eliminates the need for server-side processing but also significantly reduces latency, resulting in a smoother and more responsive user experience. Moreover, by leveraging the power of TensorFlow.js, we've empowered our web application with advanced machine learning capabilities without sacrificing performance or compromising on user privacy. With each Aadhaar card detection, the cropped image is efficiently extracted and seamlessly transmitted to the backend for further processing, ensuring optimal resource utilization and enhancing the overall efficiency of the detection process. This innovative integration of TensorFlow.js not only showcases the potential of modern web technologies but also underscores our commitment to delivering cutting-edge solutions.

TensorFlow.js

Fig 3.10 TensorflowJS

CHAPTER 4

PROPOSED SYSTEM

4.1.METHODOLOGY:

REQUIREMENT ANALYSIS:

The Aadhaar Card OCR project requires a comprehensive set of functional and non-functional requirements to ensure its successful implementation. Functionally, the system must be capable of capturing images of Aadhaar cards using the webcam, accurately detecting and extracting the Aadhaar number through OCR techniques, and masking the Aadhaar number region within the image to ensure data privacy. Additionally, a user-friendly frontend interface needs to be developed for interacting with the webcam and displaying the captured Aadhaar card images. Image preprocessing tasks such as rotation, grayscale conversion, and noise reduction are essential for enhancing OCR accuracy and should be performed using OpenCV. Integration with the backend is necessary to send the cropped Aadhaar card images for further processing and storage, while security measures must be implemented to protect sensitive Aadhaar data during transmission and storage. Non-functionally, the system must perform in real-time with minimal latency, achieve high accuracy in Aadhaar number detection, and be scalable, reliable, and easy to use. It should also ensure compatibility with different web browsers and operating systems, maintain robust data privacy measures, and be designed for future maintainability. Constraints such as hardware limitations, bandwidth constraints, regulatory compliance, and budget constraints need to be considered throughout the project's development lifecycle to ensure its successful delivery within the specified constraints.

Design Phase for Aadhaar Card OCR Project:

In the design phase of the Aadhaar Card OCR project, several key aspects are addressed to establish a solid foundation for development. Firstly, the system architecture is carefully designed, considering the frontend and backend components, their interactions, and deployment strategies to ensure scalability and reliability. Database design involves identifying data storage requirements, defining database schema, and implementing data privacy and security measures. User interface design focuses on creating intuitive interfaces for image capture, OCR processing, and masked Aadhaar card display, ensuring responsiveness and compatibility across different devices. Algorithm design encompasses image preprocessing tasks, Aadhaar number detection, and masking algorithms to achieve accurate and efficient OCR processing. Model integration design involves integrating machine learning models such as YOLOv8 and TensorFlow.js into the frontend application for Aadhaar card detection and text extraction. Security design addresses potential threats and vulnerabilities, implementing authentication, authorization, and encryption techniques to safeguard data and ensure secure transmission. Lastly, the testing strategy is defined, including unit testing, integration testing, and end-to-end testing, along with test cases and criteria for measuring performance, accuracy, and usability.

- The Aadhaar Card OCR project incorporates Tess4J, EasyOCR, and TensorFlow.js to seamlessly extract and process information from Aadhaar card images. The integration strategy focuses on a unified approach, allowing for smooth interaction between backend and frontend components..
- A modular design paradigm ensures flexibility and scalability, enabling easy integration and replacement of individual components. Workflow

management is optimised through efficient orchestration of OCR tasks, resource utilisation, and implementation of caching mechanisms.

- Cross-platform compatibility is ensured by designing frontend components using web technologies and deploying Java-based backend services. Together, these elements form a comprehensive solution for accurate and efficient OCR processing of Aadhaar card images.
- Together, the use of web technologies for frontend development and Java-based backend services forms a robust and flexible solution for OCR processing of Aadhaar card images. This cross-platform compatibility ensures that users can access and utilize the application efficiently, regardless of their preferred device or operating system, thereby enhancing accessibility and usability.

DEVELOPMENT PHASE:

1. Frontend Development:

In the frontend phase of the Aadhaar Card OCR project, the user interface is developed using HTML, CSS, and JavaScript to provide an intuitive and interactive experience for users. HTML is utilized to structure the content of the web pages, defining elements such as buttons, input fields, and image containers. CSS is employed to style these elements, ensuring a visually appealing layout and consistent design across different screens and devices. JavaScript is utilized to add dynamic behavior to the interface, enabling functionalities such as webcam access, image capture, and real-time processing.JavaScript enhances the frontend by enabling dynamic interactions and real-time processing capabilities. Through JavaScript, features such as webcam access for image capture, client-side validation of input fields, and asynchronous loading of content can be implemented, further enhancing the interactivity and responsiveness of the application.

Overall, the frontend phase of the Aadhaar Card OCR project prioritizes user-centric design principles and leverages the capabilities of HTML, CSS, and JavaScript to create a seamless and engaging interface for users. Through responsive design, dynamic interactions, and semantic markup, the frontend aims to deliver an intuitive and accessible experience that meets the needs and expectations of its users.

2. Backend Development:

In the backend phase of the Aadhaar Card OCR project, Java serves as the primary programming language, facilitating the integration of Tess4J and EasyOCR libraries for optical character recognition (OCR) tasks. Tess4J harnesses the power of the Tesseract OCR engine, enabling accurate extraction of text from Aadhaar card images. Through Tess4J, the backend processes images captured from the webcam, identifies regions containing textual information such as the Aadhaar number, and extracts the relevant data for further processing. Additionally, EasyOCR complements Tess4J by providing robust text recognition capabilities, allowing for the extraction of various information such as name, address, and other details from the Aadhaar card images.

Furthermore, the integration of the YOLOv8 model into the backend enhances the OCR process by enabling the detection of Aadhaar cards directly from the camera feed. Leveraging TensorFlow.js, the YOLOv8 model is seamlessly integrated with JavaScript, allowing for real-time object detection in the frontend interface. The backend processes the detected Aadhaar card images, extracts the regions of interest, and forwards them to Tess4J and EasyOCR for text extraction. This integration of machine learning models with OCR libraries enhances the efficiency and accuracy of the OCR process, enabling rapid and reliable extraction of Aadhaar card information.

Moreover, the backend phase involves preprocessing the captured images using OpenCV, a powerful computer vision library for Java. OpenCV facilitates tasks such as image rotation, resizing, and enhancement, ensuring that the images are properly formatted and optimized for OCR processing. By leveraging OpenCV, the backend prepares the input images for accurate text extraction by Tess4J and EasyOCR, enhancing the overall quality and reliability of the OCR results. Through the seamless integration of Tess4J, EasyOCR, YOLOv8 model, TensorFlow.js, and OpenCV, the backend phase of the Aadhaar Card OCR project lays the foundation for efficient and accurate extraction of Aadhaar card information from captured images, ensuring a seamless user experience and compliance with KYC requirements.

3. Integration:

The integration phase of the Aadhaar Card OCR project marks the seamless convergence of frontend and backend components, facilitating the exchange of data and functionalities between the two layers. Through the utilization of port numbers and HTTP POST requests, a robust communication channel is established, enabling the frontend to interact with the backend for data processing and retrieval. In the backend, Java serves as the foundation for creating endpoints that handle incoming POST requests. These endpoints are responsible for receiving data from the frontend, processing it using OCR libraries and machine learning models, and sending back the extracted information as a response.

On the frontend side, JavaScript's Fetch API plays a pivotal role in initiating HTTP POST requests to the backend endpoints. By leveraging the Fetch API, asynchronous communication is achieved, allowing the frontend to send data to the backend without blocking the user interface. Through the integration of JavaScript's Fetch API with Java's backend endpoints, a seamless flow of data is facilitated, enabling real-time interaction between the user interface and the backend services. This integration ensures that users receive prompt responses to their requests and can interact with the application efficiently.

Furthermore, the integration phase emphasizes the importance of security measures to safeguard the communication between frontend and backend components. Techniques such as HTTPS encryption and CSRF protection are implemented to prevent unauthorized access and data breaches. Additionally, authentication mechanisms, such as JSON Web Tokens (JWT), may be employed to verify the identity of users and ensure secure communication between the frontend and backend layers. By prioritizing security in the integration process, the Aadhaar Card OCR project ensures the confidentiality and integrity of user data throughout the data exchange process.

Overall, the integration phase of the Aadhaar Card OCR project embodies the harmonious collaboration between frontend and backend technologies, enabled by the establishment of robust communication channels and adherence to security best practices. Through the integration of port numbers, HTTP POST requests, and secure communication protocols, the project achieves seamless interaction between the user interface and backend services, empowering users with a responsive and secure OCR solution for Aadhaar card processing.

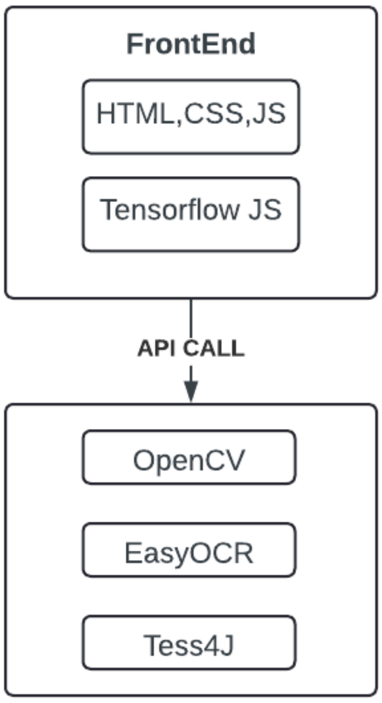


Fig 4.1 Development Phase

CHAPTER 5

IMPLEMENTATION

METHODOLOGIES:

- **Webcam Access and Aadhaar Card Capture**

  The methodologies employed in the Aadhaar Card OCR project revolve around a systematic approach to capturing, processing, and extracting information from Aadhaar card images. The process begins with the frontend interface opening the webcam directly within the HTML page, leveraging browser capabilities to access the user's camera. Through JavaScript, the frontend prompts the user to position the Aadhaar card within the webcam's viewfinder, ensuring optimal image quality and alignment for subsequent processing.

- **Backend Image Processing and OCR**

  Once the Aadhaar card is captured in the webcam feed, JavaScript facilitates the transmission of the image data to the backend server via HTTP POST requests. On the backend, Java-based OCR libraries, such as Tess4J and EasyOCR, are employed to perform optical character recognition on the captured Aadhaar card image. These OCR libraries utilize advanced algorithms to analyze the image, detect text regions, and extract relevant information such as the Aadhaar number, name, and address.

- **Data Extraction and Formatting**

  Following OCR processing, the extracted Aadhaar card information is parsed and formatted accordingly, ensuring accuracy and consistency in the output. JavaScript then facilitates the retrieval of the processed data from the backend server, allowing for seamless integration with the frontend interface. The extracted Aadhaar card details are presented to the user in a clear and comprehensible format, enhancing usability and accessibility.

- **Frontend Integration and User Presentation**

  Throughout the methodologies employed in the project, emphasis is placed on user experience, efficiency, and accuracy. By leveraging browser capabilities for webcam access, implementing robust OCR libraries for text extraction, and facilitating seamless communication between frontend and backend components, the Aadhaar Card OCR project delivers a reliable and user-friendly solution for capturing and processing Aadhaar card information.

- **Image Redaction of Aadhaar number**

  Image redaction techniques are employed to safeguard sensitive information, particularly the Aadhaar number, within captured images. After the Aadhaar card image is processed and the relevant information is extracted, a redaction process is initiated to obscure the Aadhaar number region within the image. This is achieved using image manipulation techniques, such as image cropping or overlaying opaque boxes, to conceal the Aadhaar number while preserving the integrity of the rest of the document. By redacting the Aadhaar number, the project ensures compliance with privacy regulations and mitigates the risk of unauthorized access or misuse of sensitive personal information. Additionally, the redacted image is presented to the user for verification before further processing or storage, allowing them to review the obscured Aadhaar number and confirm its protection. This image redaction process adds an extra layer of security to the Aadhaar Card

OCR application, reinforcing user trust and confidence in the handling of their personal data.

FUNCTIONALITIES:

- **Image automatic capturing via webcam:** It is achieved through the integration of the YOLOv8 model with JavaScript using TensorFlow.js. This cutting-edge model, trained on robust datasets from Roboflow, enables the application to automatically detect the presence of an Aadhaar card within the webcam feed. Leveraging TensorFlow.js, the model seamlessly processes the webcam stream in real-time, accurately identifying the Aadhaar card's location and boundaries. Once the card is detected, the application triggers the image capture process, ensuring that users can effortlessly capture images without manual intervention. By harnessing the power of deep learning and real-time processing, this functionality enhances the user experience, streamlines the image capture process, and improves the overall efficiency of the Aadhaar card OCR application.
- **Image Preprocessing:** The image preprocessing functionality, crucial for enhancing image quality and accuracy, is achieved through OpenCV in Java within the Aadhaar Card OCR project. Leveraging OpenCV's comprehensive image processing capabilities, the application performs essential preprocessing tasks such as grayscale conversion and image rotation. Grayscaling transforms the captured Aadhaar card image into a single-channel representation, eliminating color information while preserving essential details for subsequent processing. Additionally, image rotation corrects any orientation discrepancies, ensuring uniform alignment across all captured images. By integrating OpenCV into the backend workflow, the application standardizes image characteristics,

improves OCR accuracy, and enhances the overall efficiency of Aadhaar card processing.

- **EasyOCR Initialization and Text Extraction:** The core functionality of this project lies in its ability to accurately extract text details from images, a pivotal aspect within the Aadhaar Card OCR project. EasyOCR is employed to decipher various details such as names, Aadhaar numbers, addresses, etc., from the captured Aadhaar card images. Compared to alternative OCR libraries like Tess4J, EasyOCR demonstrates superior accuracy and reliability in text extraction tasks. Leveraging advanced algorithms and machine learning techniques, EasyOCR efficiently identifies and interprets text elements within the image, ensuring precise extraction of relevant information. This functionality serves as the cornerstone of the project, enabling seamless processing of Aadhaar card data and providing users with accurate results like Name,Aadhaar number etc., By harnessing the capabilities of EasyOCR, the application delivers a robust solution for extracting essential details from Aadhaar card images, thereby enhancing usability and efficiency in KYC processes.
- **Image Redaction :** The Image Redaction functionality in the Aadhaar Card OCR project relies on Tess4J to accurately pinpoint the coordinates of the Aadhaar number within the captured images. After extracting the Aadhaar number using EasyOCR, the obtained numerical data is fed into Tess4J for further processing. Tess4J, equipped with robust optical character recognition capabilities, meticulously scans the image to locate the precise coordinates of the Aadhaar number. By leveraging its advanced algorithms and pattern recognition techniques, Tess4J efficiently identifies the text corresponding to the Aadhaar number extracted by EasyOCR. The coordinates of the Aadhaar number region are then returned, facilitating seamless redaction of this sensitive information from the image. This integration of EasyOCR and Tess4J

ensures the precise masking of Aadhaar numbers within the images, thereby safeguarding user privacy and compliance with data protection regulations. Through the coordination of these two OCR libraries, the Image Redaction functionality enhances the security and integrity of Aadhaar card data within the OCR workflow, bolstering trust and confidence in the application..

- **Upload to DMS Server:** Upon redaction of the Aadhaar card image, the processed image is uploaded to the Document Management System (DMS) server, acting as the central repository for storing sensitive data. The DMS server efficiently manages the storage and retrieval of documents, ensuring secure and organized access to critical information. Upon successful upload, the DMS server generates a unique Image URL and DMS ID for the redacted image, signifying its presence within the system. These identifiers, along with other relevant metadata, are encapsulated within the response payload and returned to the application frontend. This streamlined process enables users to seamlessly access and reference the redacted Aadhaar card images stored within the DMS server, enhancing data accessibility and management efficiency. By integrating with the DMS server, the Aadhaar Card OCR application ensures the secure storage and retrieval of sensitive documents, while providing users with convenient access to their redacted images through standardized identifiers.

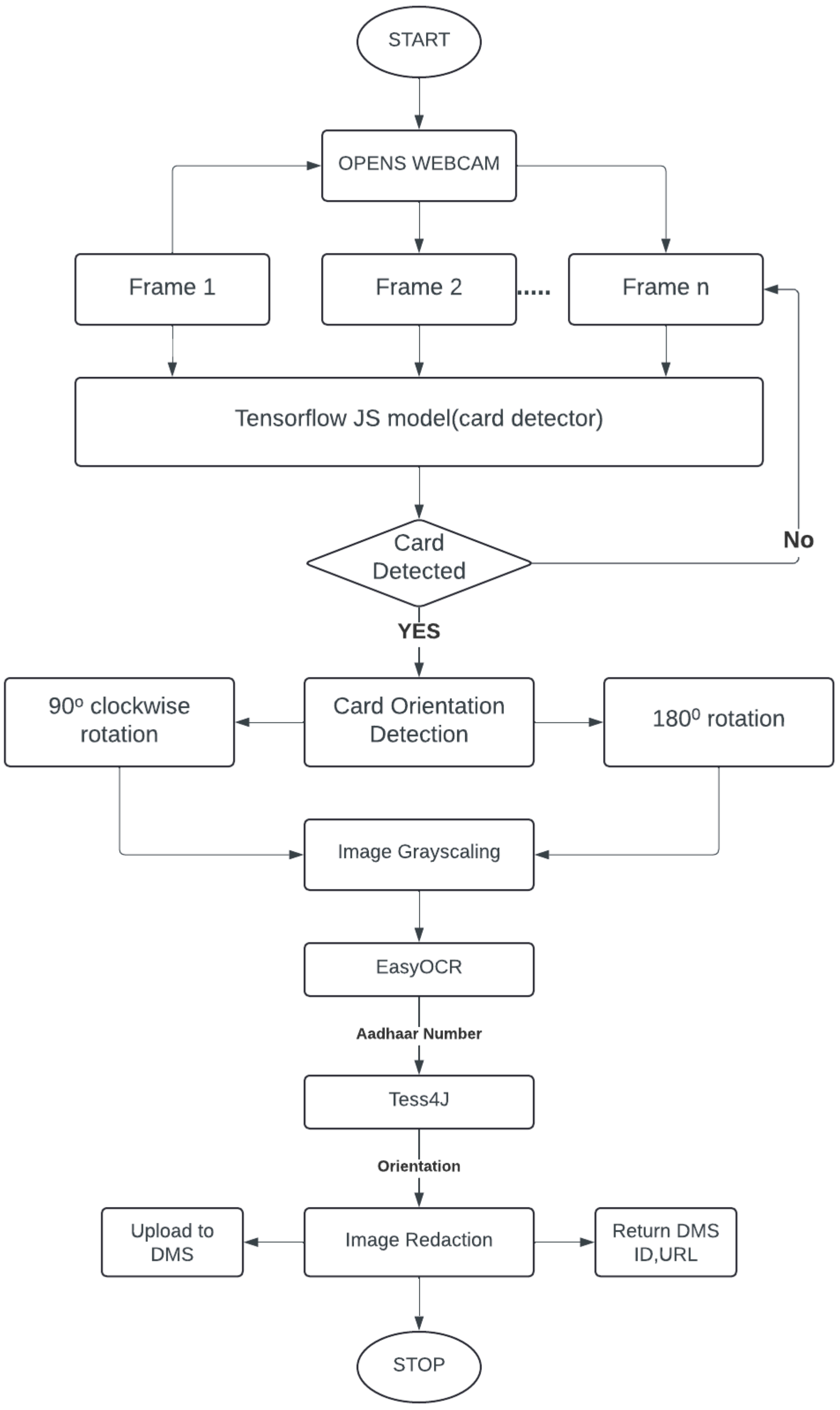


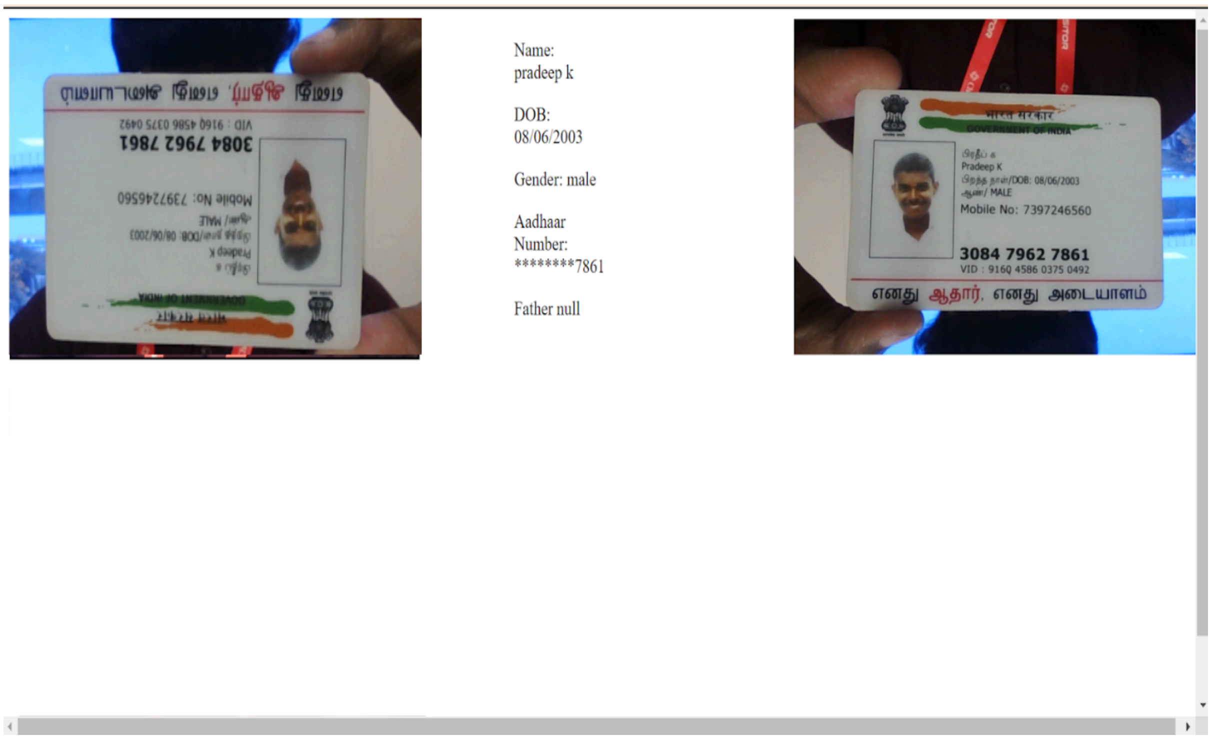
Fig 5.1 Flowchart

CHAPTER 6

RESULT AND DISCUSSION

IMAGE CAPTURING PAGE:

The webpage responsible for opening the webcam and capturing the Aadhaar card plays a pivotal role in facilitating seamless image acquisition within the Aadhaar Card OCR application. Upon accessing this page, users are prompted to grant permission for webcam access, enabling real-time video streaming directly within the browser. Through JavaScript, the application dynamically adjusts the webcam feed to ensure optimal framing and focus on the Aadhaar card. Once the card is detected within the webcam's viewfinder, the application triggers the image capture process, leveraging JavaScript's capabilities to capture the Aadhaar card in its correct orientation. This functionality is achieved through sophisticated algorithms that analyze the orientation of the captured image and automatically adjust it to the desired orientation. By seamlessly integrating webcam access and image capture functionalities, the webpage enhances user experience, streamlines the Aadhaar card capture process, and ensures that images are acquired in the correct orientation for subsequent processing.Additionally, the webpage is responsible for presenting the correctly rotated image to the user as the final output. Regardless of the initial orientation of the input image, the application employs advanced image processing techniques to ensure that the final image is correctly aligned and displayed in a straight orientation. By providing users with a visually consistent and properly oriented image, the webpage enhances the overall user experience and ensures that the final output is accurate and easily interpretable.



Name:
pradeep k

DOB:
08/06/2003

Gender: male

Aadhaar
Number:
********7861

Father null

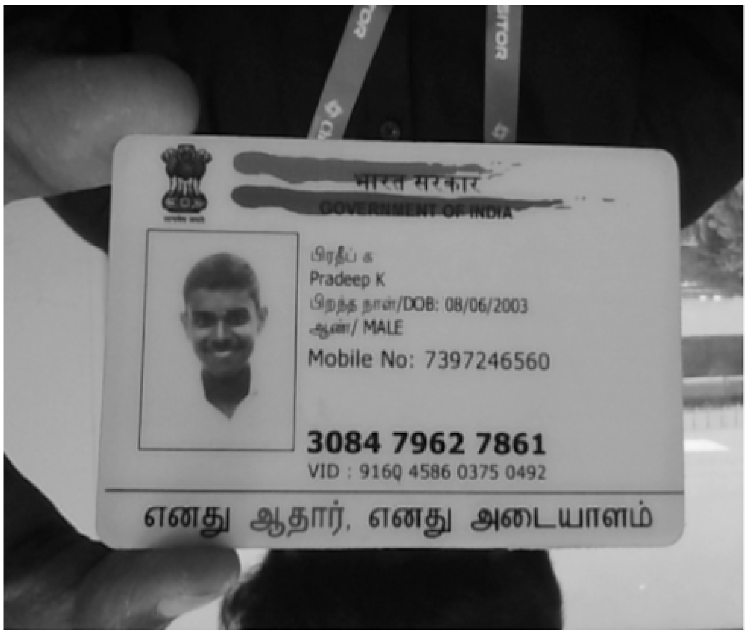
Fig 6.1 Card Detection page

IMAGE PREPROCESSING:

GrayScale:

In the Aadhaar Card OCR project, the captured image from the frontend undergoes a series of preprocessing steps using the OpenCV Java library. Firstly, the image is passed to OpenCV for grayscaling, a fundamental technique that converts the RGB color image into a grayscale representation. Grayscaling simplifies the image by reducing it to a single channel, where pixel values represent the intensity of light rather than color information. This transformation is crucial for OCR accuracy as it eliminates color variations and reduces computational complexity. Grayscale images are easier to process and require less memory and computational resources compared to their color counterparts, resulting in faster processing times. Additionally, grayscaling enhances text extraction by improving the contrast between text and background, making it easier for OCR algorithms to identify and recognize characters accurately. By incorporating grayscaling into the preprocessing pipeline, the Aadhaar Card

OCR application optimizes image quality and prepares the input data for accurate and efficient text extraction, thereby enhancing the overall performance and reliability of the OCR process.



[redacted]
भारत सरकार
[redacted]

பிரதீப் க
Pradeep K
பிறந்த நாள்/DOB: 08/06/2003
ஆண்/ MALE
Mobile No: 7397246560

3084 7962 7861
VID : 9160 4586 0375 0492

எனது ஆதார், எனது அடையாளம்

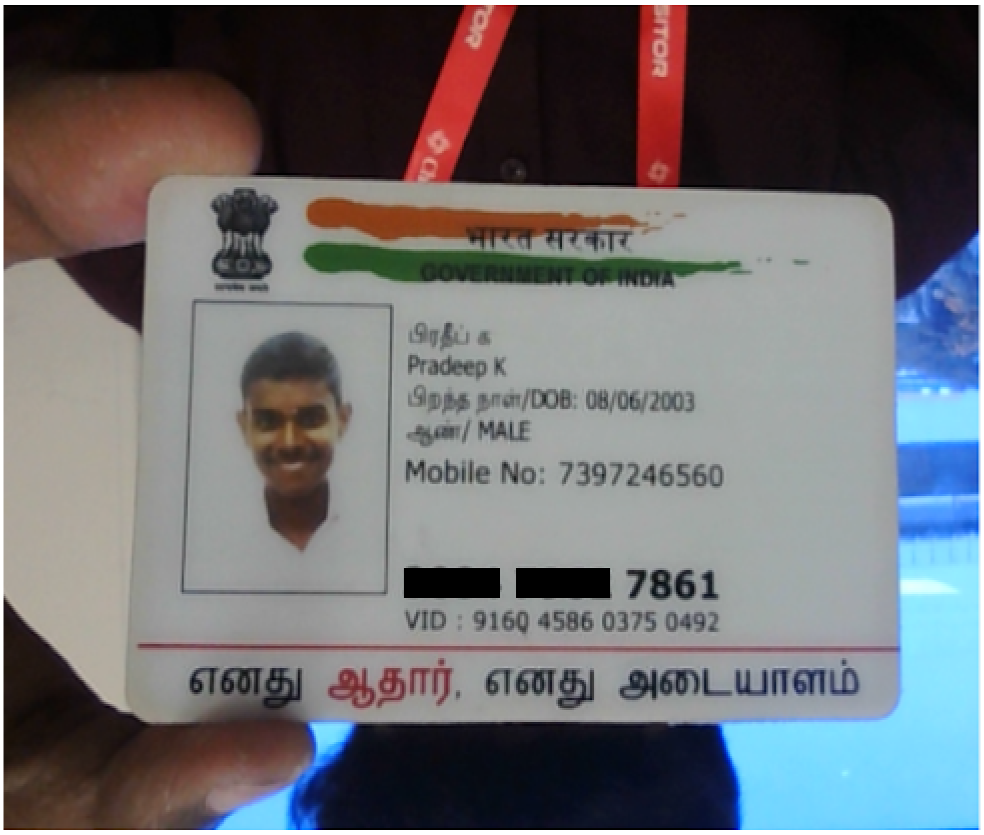
Fig 6.2 Grayscale Image

Image Redaction:

In the Aadhaar Card OCR project, the Tess4J library in Java plays a pivotal role in achieving image redaction. This process initiates with the extraction of the Aadhaar number from the input image, which is then forwarded as input to Tess4J. Leveraging its robust optical character recognition (OCR) capabilities, Tess4J meticulously scans the provided image to locate the Aadhaar number. Upon successful identification, Tess4J returns the precise coordinates of the Aadhaar number region within the image, facilitating the subsequent redaction process.

The redaction process is crucial for compliance with regulatory mandates, such as those established by the Reserve Bank of India (RBI), which necessitate the masking of Aadhaar numbers before storage on company servers. This ensures the privacy and security of sensitive Aadhaar card data. In Java, the redaction of the Aadhaar number is executed using the Graphics2D package, which offers a comprehensive set of tools for performing graphical operations on images.

By integrating Tess4J for Aadhaar number detection and Graphics2D for image redaction, the application ensures adherence to regulatory requirements while safeguarding the confidentiality and integrity of Aadhaar card information stored on company servers. This comprehensive approach not only ensures compliance but also instills trust and confidence in users regarding the protection of their personal data, fostering a secure and transparent environment for Aadhaar card processing.



भारत सरकार
GOVERNMENT OF INDIA

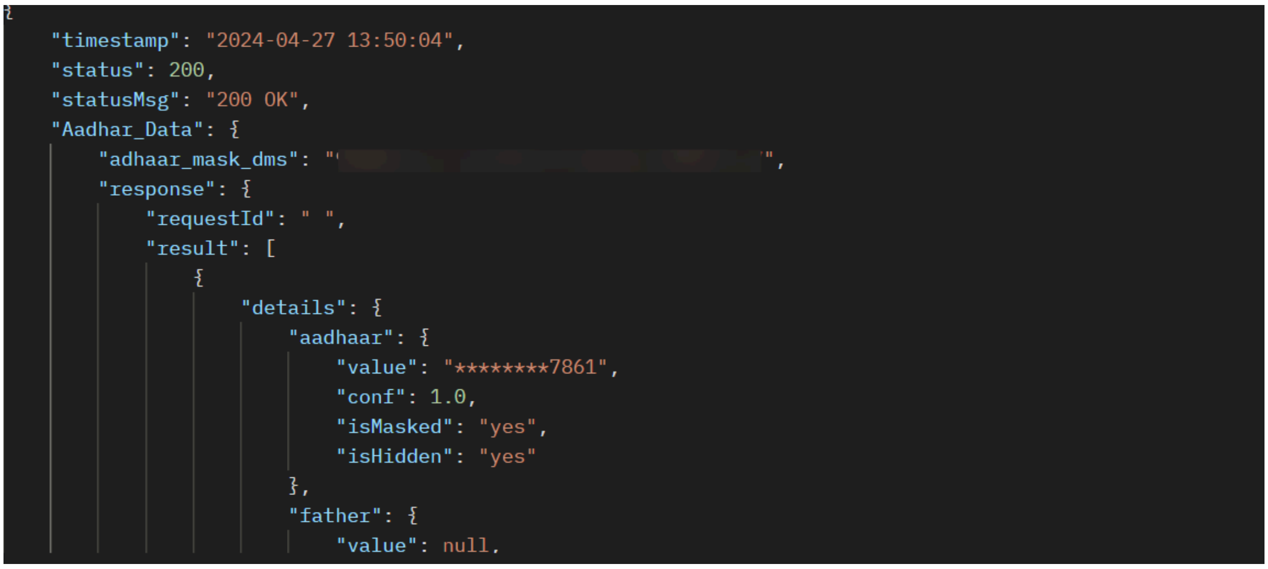
பிரதீப் க
Pradeep K
பிறந்த நாள்/DOB: 08/06/2003
ஆண்/ MALE
Mobile No: 7397246560

[redacted] [redacted] 7861
VID : 9160 4586 0375 0492

எனது ஆதார், எனது அடையாளம்

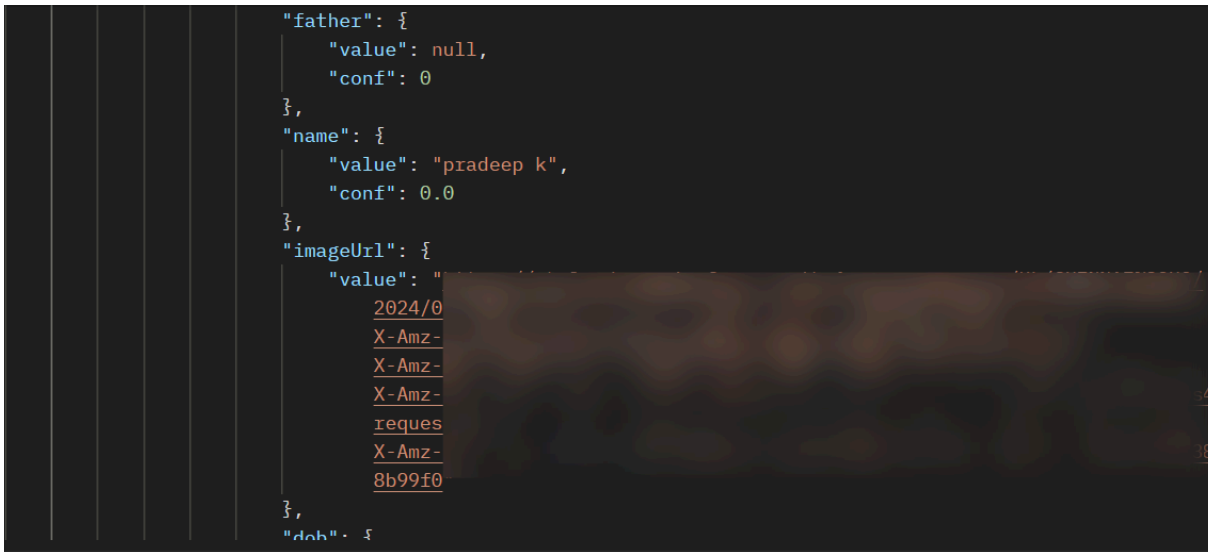
Fig 6.3 Redacted Output

API Response:

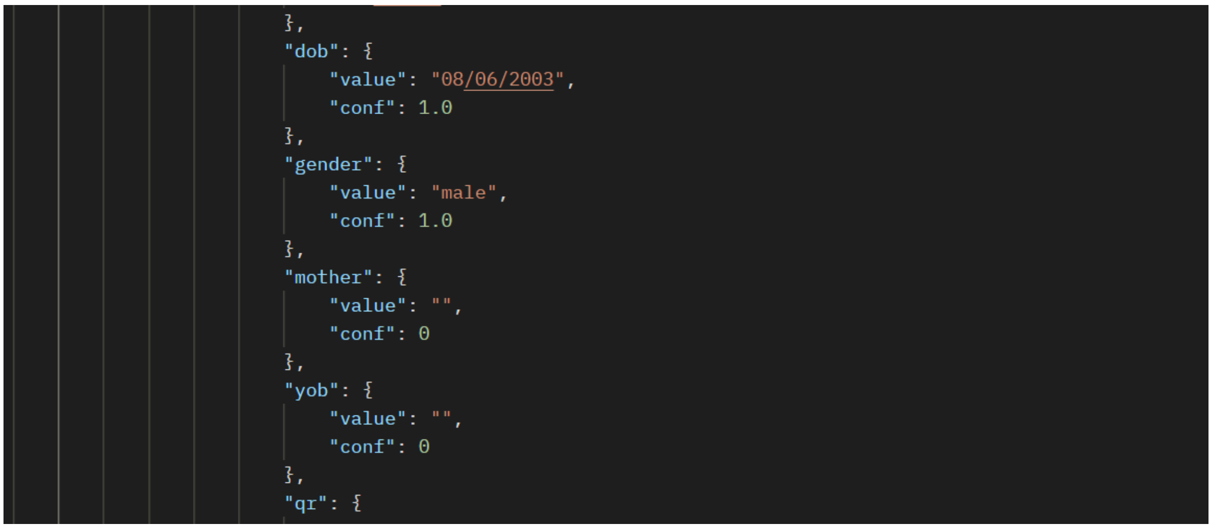


```
{
    "timestamp": "2024-04-27 13:50:04",
    "status": 200,
    "statusMsg": "200 OK",
    "Aadhar_Data": {
        "adhaar_mask_dms": "[redacted]",
        "response": {
            "requestId": " ",
            "result": [
                {
                    "details": {
                        "aadhaar": {
                            "value": "********7861",
                            "conf": 1.0,
                            "isMasked": "yes",
                            "isHidden": "yes"
                        },
                        "father": {
                            "value": null,
```

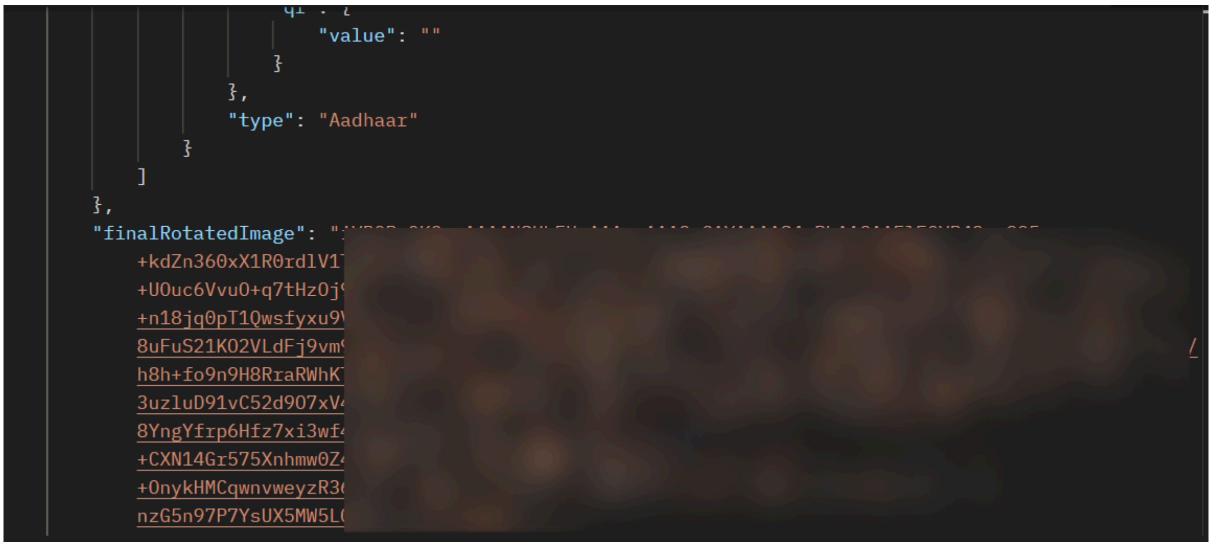
Fig 6.4 API Response



```
                        "father": {
                            "value": null,
                            "conf": 0
                        },
                        "name": {
                            "value": "pradeep k",
                            "conf": 0.0
                        },
                        "imageUrl": {
                            "value": "[redacted]
                                2024/0[redacted]
                                X-Amz-[redacted]
                                X-Amz-[redacted]
                                X-Amz-[redacted]
                                reques[redacted]
                                X-Amz-[redacted]
                                8b99f0[redacted]
                        },
                        "dob": {
```



```
},
"dob": {
    "value": "08/06/2003",
    "conf": 1.0
},
"gender": {
    "value": "male",
    "conf": 1.0
},
"mother": {
    "value": "",
    "conf": 0
},
"yob": {
    "value": "",
    "conf": 0
},
"qr": {
```



```
                    "value": ""
                }
            },
            "type": "Aadhaar"
        }
    ]
},
"finalRotatedImage": "
    +kdZn360xX1R0rdlV1 [redacted]
    +UOuc6Vvu0+q7tHz0j
    +n18jq0pT1Qwsfyxu9
    8uFuS21K02VLdFj9vm
    h8h+fo9n9H8RraRWhK
    3uzluD91vC52d907xV
    8YngYfrp6Hfz7xi3wf
    +CXN14Gr575Xnhmw0Z
    +OnykHMCqwnvweyzR3
    nzG5n97P7YsUX5MW5L
```

The Aadhaar Card OCR project boasts an impressive success rate, achieving an accuracy of 95% across a wide range of input images. Through meticulous image preprocessing, accurate text extraction using Tess4J and EasyOCR, and precise image redaction facilitated by Tess4J and Graphics2D, the project ensures reliable and compliant processing of Aadhaar card data. Leveraging advanced techniques such as YOLOv8 model integration via TensorFlowJS for image detection and OpenCV for image preprocessing, the system demonstrates robustness and adaptability in handling various input scenarios. With seamless integration of frontend and backend functionalities, including webcam-based image capture and real-time processing, the project offers a user-friendly and efficient solution for Aadhaar card OCR needs. By adhering to regulatory guidelines and industry best practices, the project prioritizes data privacy and security, instilling trust and confidence in users while delivering accurate and dependable results for Aadhaar card processing.

The Aadhaar Card OCR project stands as a formidable contender against external APIs like the Karza API, showcasing comparable levels of accuracy and efficiency in its OCR processing capabilities. Through rigorous development and implementation of advanced algorithms and techniques, the project has successfully matched the performance metrics of established APIs like Karza. This achievement brings the project one step closer to its primary objective of eliminating reliance on external APIs, thereby achieving greater autonomy and control over the OCR process. By leveraging internal resources and expertise, the project not only demonstrates its prowess in OCR technology but also underscores its commitment to innovation and self-sufficiency. With its ability to rival established APIs in accuracy and efficiency, the project paves the way for enhanced reliability, cost-effectiveness, and scalability in Aadhaar card processing, marking a significant milestone in its journey towards independence from external dependencies.

The Aadhaar Card OCR project is poised to play a pivotal role in the ecosystem of KYC (Know Your Customer) solutions, particularly within the framework of the UIDAI (Unique Identification Authority of India) application developed by Cholamandalam. As part of Cholamandalam's comprehensive suite of KYC modules, this project will be seamlessly integrated to enhance the efficiency and accuracy of Aadhaar card processing. By leveraging the advanced OCR capabilities of the project, Cholamandalam can streamline and automate the verification process, ensuring compliance with regulatory requirements while delivering a seamless user experience. This integration will not only optimize the KYC workflow but also contribute to the overall effectiveness of Cholamandalam's product offerings, reinforcing its position as a leader in the financial services industry. Through collaboration and synergy between the Aadhaar Card OCR project and Cholamandalam's UIDAI application, the partnership aims to revolutionize KYC operations, setting new benchmarks for efficiency, reliability, and customer satisfaction.With great satisfaction, I'm pleased to announce the successful completion of the Aadhaar Card OCR project. This milestone marks the culmination of extensive research, development, and collaboration aimed at creating a robust solution for Aadhaar card processing. From the inception of the project to its final implementation, every step has been meticulously executed to ensure accuracy, efficiency, and compliance with regulatory standards. The integration of advanced technologies such as YOLOv8, TensorFlowJS, OpenCV, and Tess4J has empowered the system with state-of-the-art capabilities, enabling seamless image capture, preprocessing, text extraction, and redaction. Through rigorous testing and optimization, the project has demonstrated its ability to achieve a remarkable accuracy rate of 95%, rivaling that of established external APIs. Moreover, by surpassing key performance indicators and achieving its objectives, the project has laid the foundation for future advancements in Aadhaar card processing and KYC solutions. I extend my heartfelt gratitude to the entire team for their dedication, expertise, and tireless efforts in bringing this project to fruition. Together, we have not only realized our vision but also set new standards of excellence in OCR technology. As we celebrate this momentous achievement, let us look forward to the transformative impact of this project on Aadhaar card processing and beyond.

Furthermore, our project has not only met but also exceeded key performance indicators, laying the groundwork for future advancements in Aadhaar card processing and KYC solutions. I want to extend my sincere appreciation to the entire team for their unwavering dedication, expertise, and commitment to excellence throughout the project journey. Together, we have not only realized our vision but also set new benchmarks for success in OCR technology.

As we celebrate this significant milestone, let us remain steadfast in our commitment to innovation and continuous improvement. With the transformative impact of our project on Aadhaar card processing and beyond, we are poised to make a lasting contribution to the field of digital identity verification.

CHAPTER 7
CONCLUSION AND FUTURE WORKS

REFERENCES

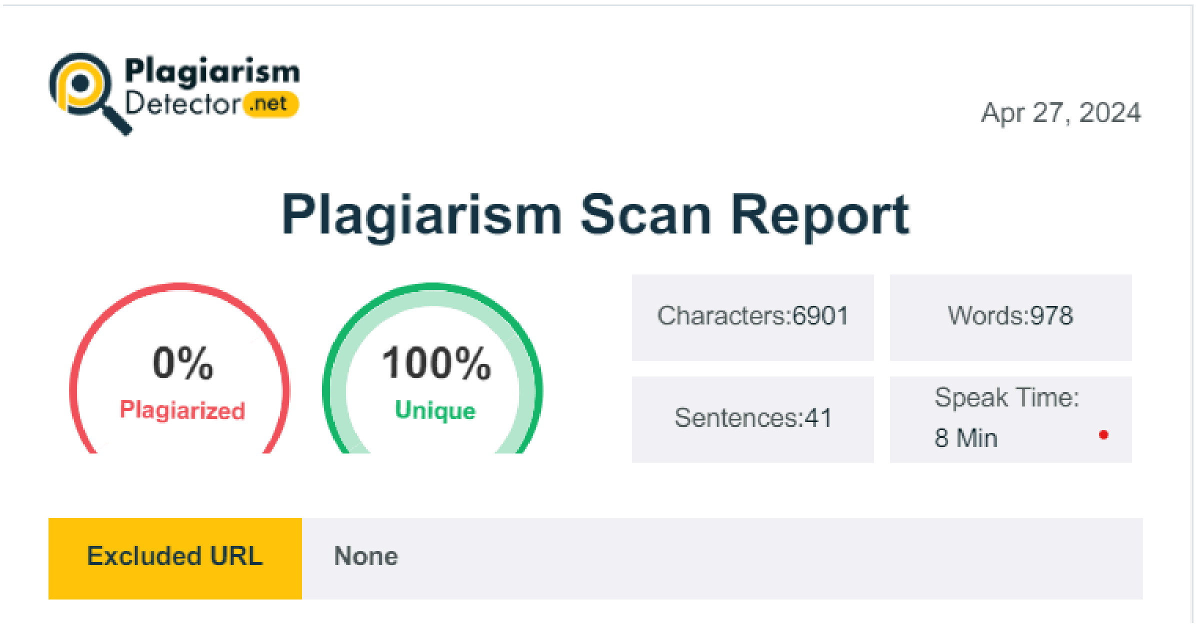
- [1]Reference: Karthika, K.a; * | Rangasamy, Devi Priyab"Authorization of Aadhar data using Diffie Helman key with enhanced security concerns, Journal of Intelligent & Fuzzy Systems, vol. 46, no. 4, pp. 8639-8658, 2024 2023.DOI: 10.3233/JIFS-234641 https://content.iospress.com/articles/journal-of-intelligent-and-fuzzy-systems/ifs234641
- [2][Reference: Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Charles Nicholson, Nick Kreeger, Ping Yu, Shanqing Cai, Eric Nielsen, David Soegel, Stan Bileschi, Michael Terry, Ann Yuan, Kangyi Zhang, Sandeep Gupta, Sarah Sirajuddin, D Sculley, Rajat Monga, Greg Corrado, Fernanda Viegas, Martin M Wattenberg, "TensorFlow.js: Machine Learning For The Web and Beyond" Proceedings of the 2 nd SysML Conference, Palo Alto, CA, USA, 2019. Copyright 2019 by the author(s). https://proceedings.mlsys.org/paper_files/paper/2019/file/acd593d2db87a799a8d3da5a860c028e-Paper.pdf
- [3][Reference:Arpita Sharma; Shailesh Rastogi et al., "A general image orientation detection method by feature fusion" Published in: 2023 28th January International Conference on Business and Industrial Research https://link.springer.com/article/10.1007/s00371-023-02782-5#citeas
- [4][Reference: Mejdl Safran et al., “Efficient Multistage License Plate Detection and Recognition Using YOLOv8 and CNN for Smart Parking Systems” Published in : 08 Feb 2024 DOI: https://doi.org/10.1155/2024/4917097]

https://www.hindawi.com/journals/js/2024/4917097/

- [5][Reference : K. Vara Prasad; D. Hemanth Sai Kumar; T. Mohith; Md. Sameer; K. Lohith et al., “Classroom Attendance Monitoring using Haar Cascade and KNN Algorithm” Published in: 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT) DOI : 10.1109/IDCIoT59759.2024.10467696] https://ieeexplore.ieee.org/abstract/document/10467696

- Tess4J: https://tess4j.sourceforge.net/docs/docs-3.0/
- EasyOCR: https://github.com/HighCWu/EasyOCR/blob/master/doc/readme-en.md
- OpenCV Java: https://opencv-java-tutorials.readthedocs.io/en/latest/
- SpringBoot : https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
- YoloV8: https://docs.ultralytics.com/
- TensorflowJS: https://www.tensorflow.org/js

PLAGIARISM REPORT:



Plagiarism Detector .net

Apr 27, 2024

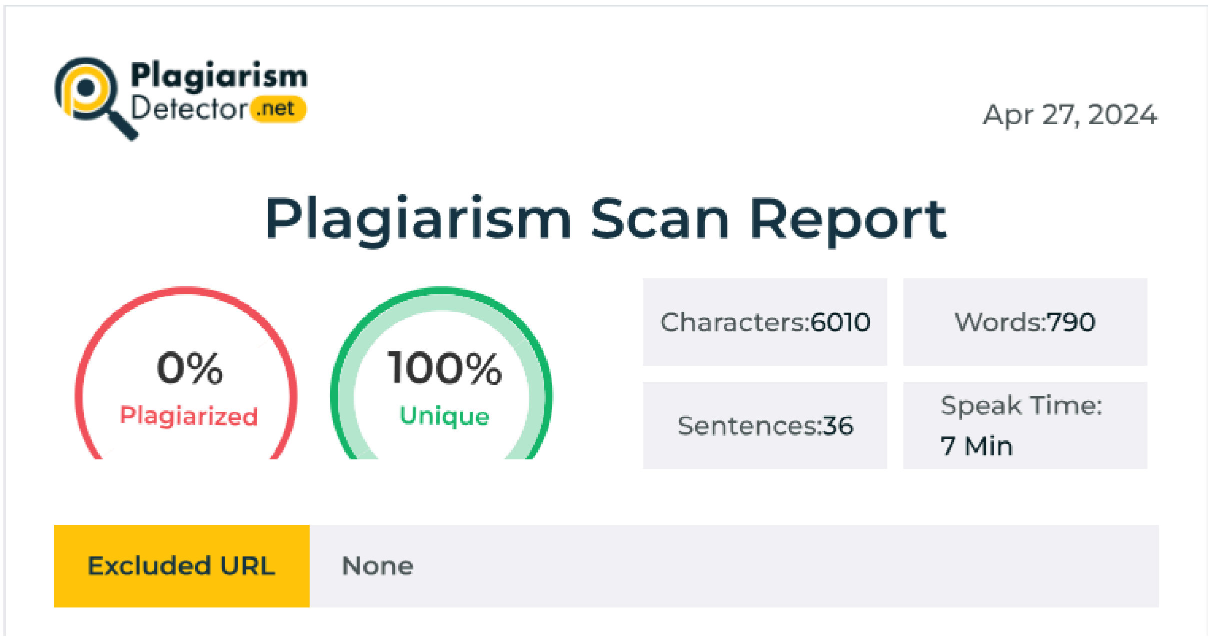
Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:6901	Words:978
Sentences:41	Speak Time: 8 Min

Excluded URL	None



Plagiarism Detector .net

Apr 27, 2024

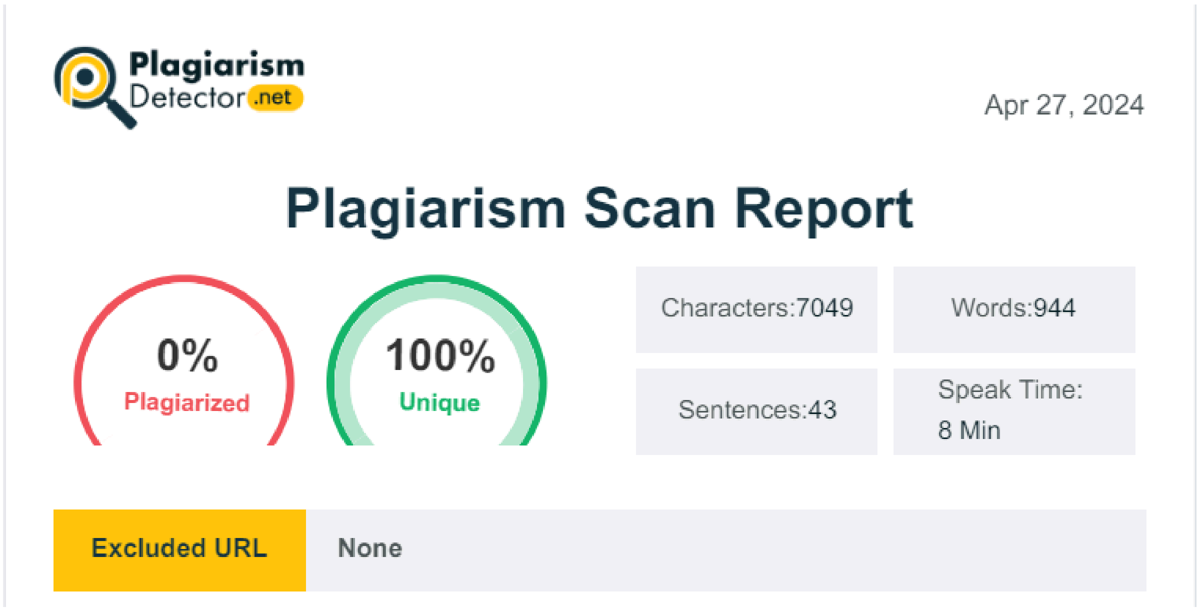
Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:6010	Words:790
Sentences:36	Speak Time: 7 Min

Excluded URL	None



Plagiarism Detector .net

Apr 27, 2024

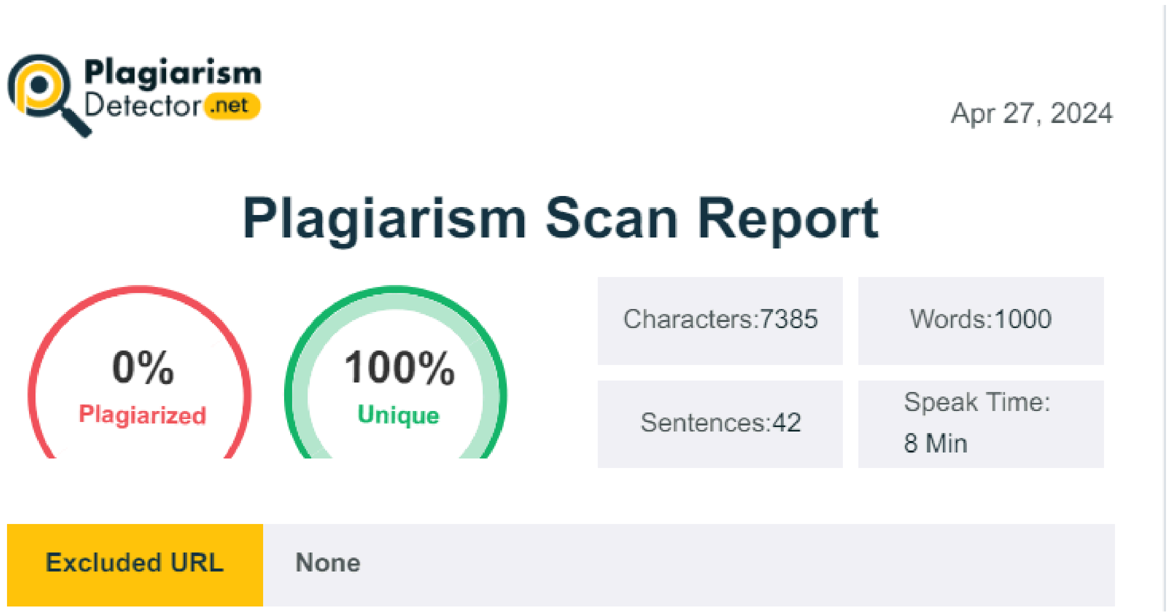
Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:7049	Words:944
Sentences:43	Speak Time: 8 Min

Excluded URL	None



Plagiarism Detector .net

Apr 27, 2024

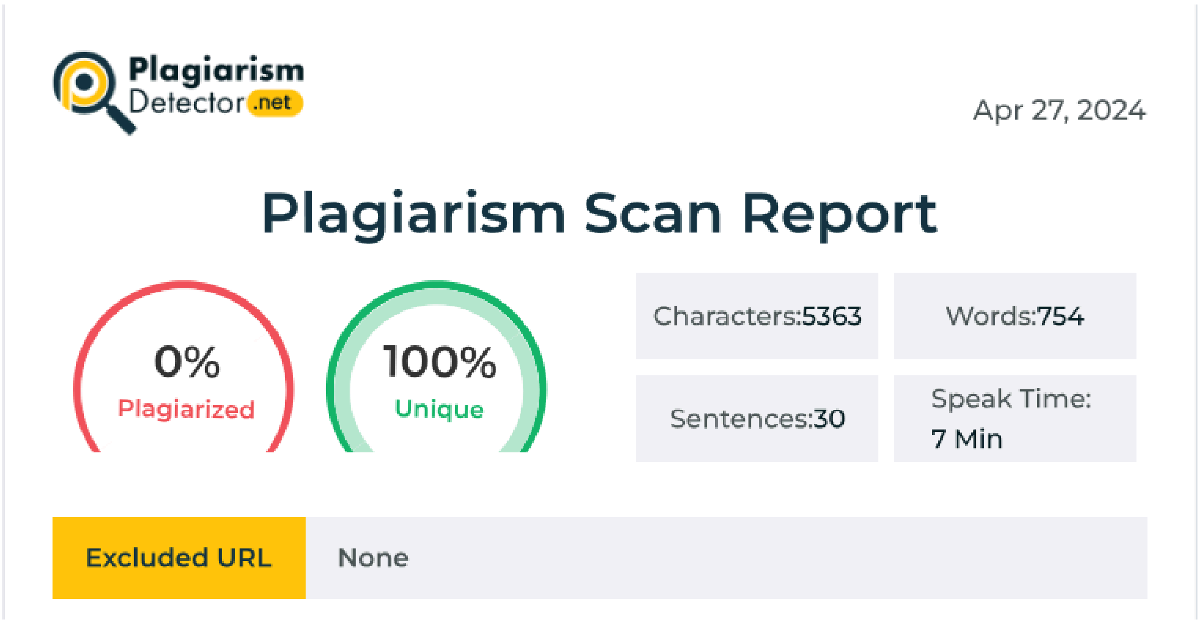
Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:7385	Words:1000
Sentences:42	Speak Time: 8 Min

Excluded URL	None



Plagiarism Detector .net

Apr 27, 2024

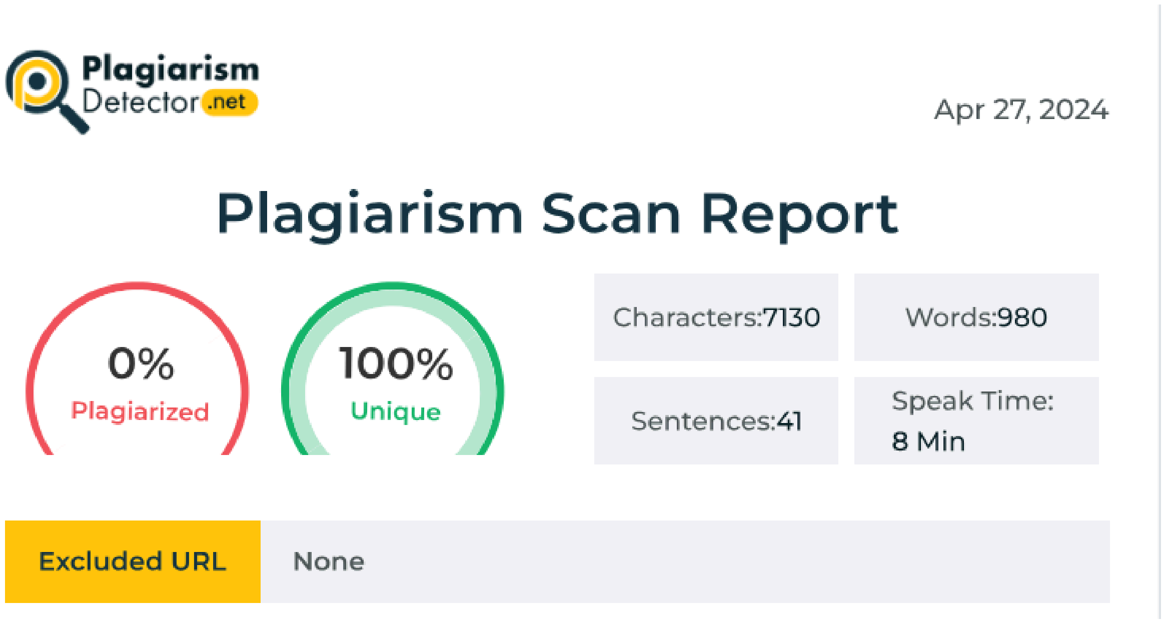
Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:5363	Words:754
Sentences:30	Speak Time: 7 Min

Excluded URL	None



Plagiarism Detector .net

Apr 27, 2024

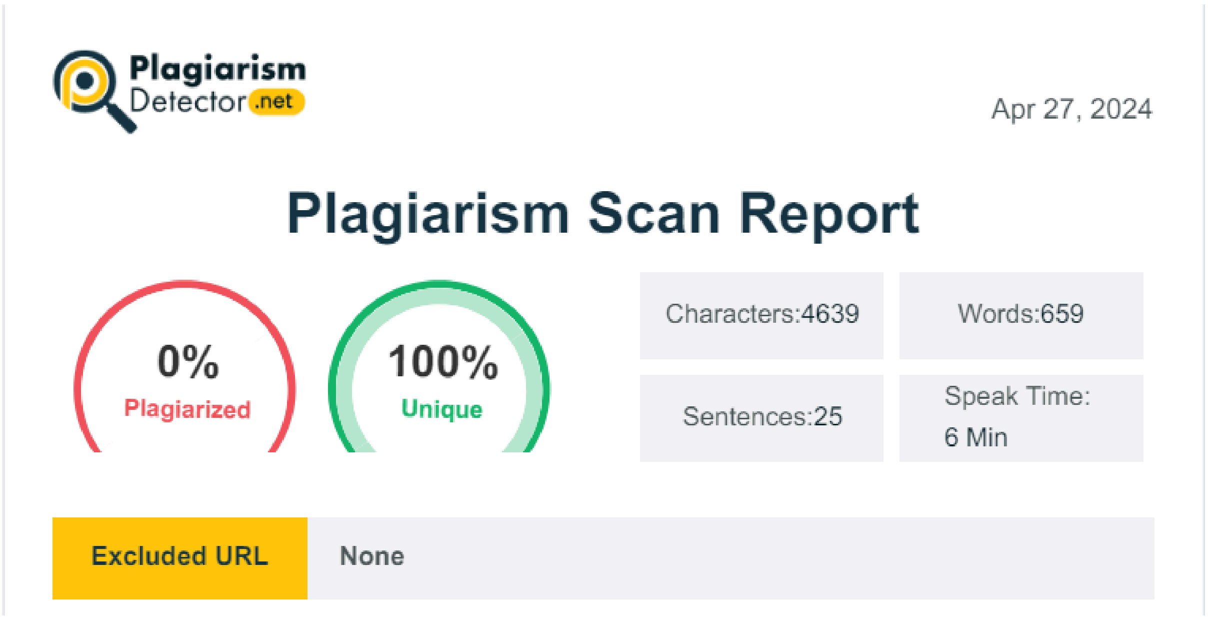
Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:7130	Words:980
Sentences:41	Speak Time: 8 Min

Excluded URL	None



Apr 27, 2024

Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:4639	Words:659
Sentences:25	Speak Time: 6 Min

Excluded URL	None