

FINAL REPORT:

Automatic Extraction of Medication Information from a Pill Bottle

Under the guidance of
Prof. Saurabh Srivastava
(Associate Professor)



**Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY
(INDIAN SCHOOL OF MINES)
DHANBAD**

By:

- 1. Avinesh Pratap Singh – 20JE0219**
- 2. Sanskar Bansal – 20JE0857**
- 3. Tirth Rami – 20JE0768**
- 4. Priyanshu Maurya – 20JE0728**
- 5. Asharam Meena – 20JE0202**

Contents

1- Requirement Analysis

2- Requirement Gathering

3- Proposed Solution and Workflow

4- System Design

5- Implementation

6- Testing

7- Results

Problem Statement:

A Label Parser app to parse the contents of medicine bottles to extract their name, composition and expiry date.

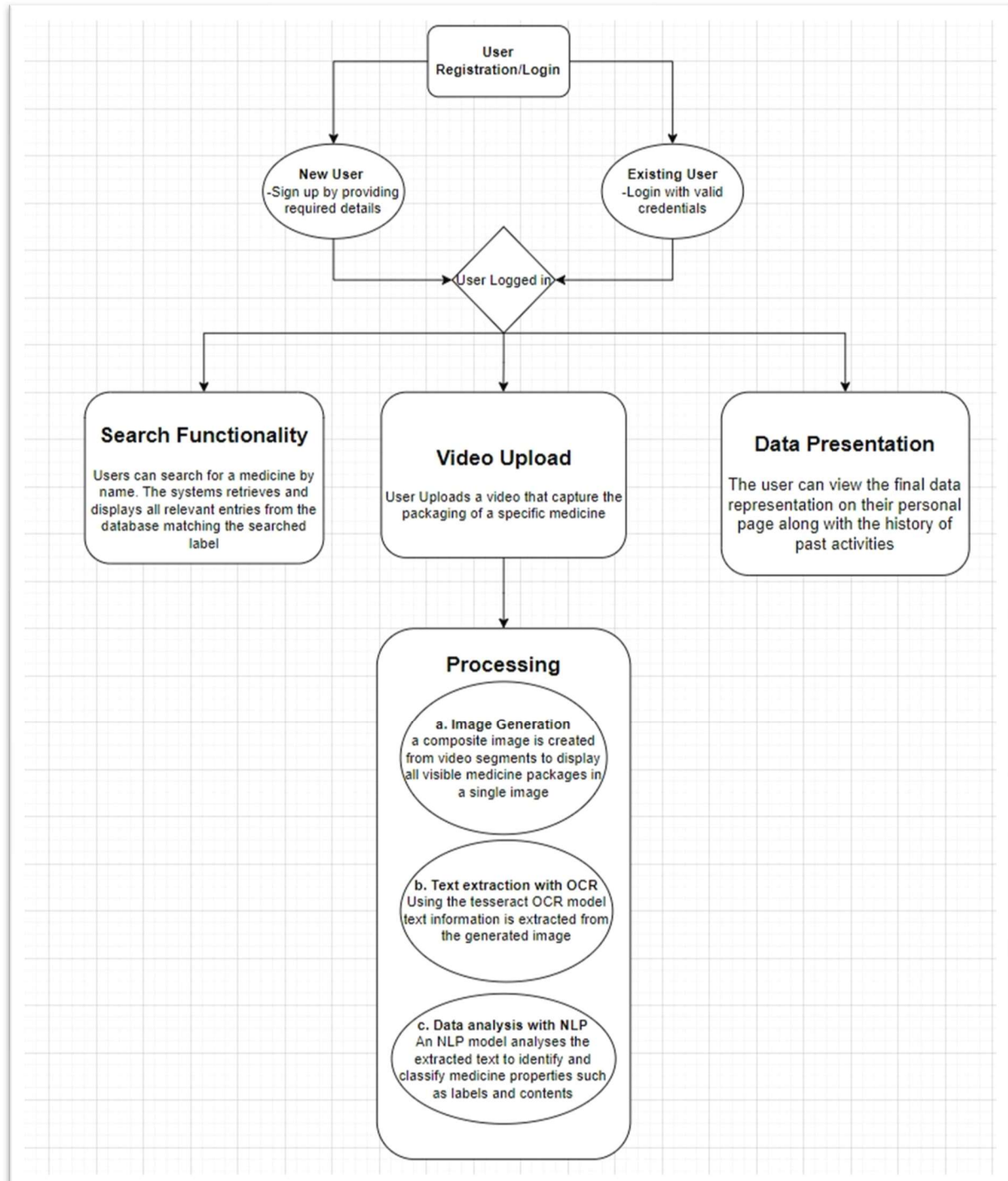
1. Requirement Analysis

- **OCR Model Implementation:** Develop an OCR model capable of accurately extracting text from images. The model should leverage Python programming language for its implementation.
- **Database Management:** Utilize MongoDB database for efficient storage and retrieval of extracted text data. MongoDB offers flexibility and scalability, making it suitable for handling diverse data types associated with OCR results.
- **Integration with Amazon AWS S3:** Employ Amazon AWS S3 as the file system for storing image files. Leveraging S3 ensures reliable and scalable storage infrastructure, facilitating seamless handling of large volumes of image data.
- **Python Libraries:** Utilize relevant Python libraries for image processing, text extraction, and integration with MongoDB and Amazon AWS S3. These libraries will augment the OCR model's functionality and streamline its integration with other components.
- **Security Implementation:** Implement token-based authentication using Python JWT (JSON Web Tokens) library to ensure secure access to the OCR system. JWT offers a robust mechanism for generating and validating access tokens, enhancing system security.

2. Requirement Gathering

- **Photographs:** We require high-quality images capturing various aspects of medicine bottles, including different angles, lighting conditions, and types of packaging. Images should focus on the labels and text present on the bottles, ensuring clarity and legibility.
- **Videos:** Short video clips showcasing the manipulation or rotation of medicine bottles will provide additional insights into how the labels are presented from different perspectives. These videos will supplement the static images and help refine our parsing algorithm's ability to handle dynamic scenarios.

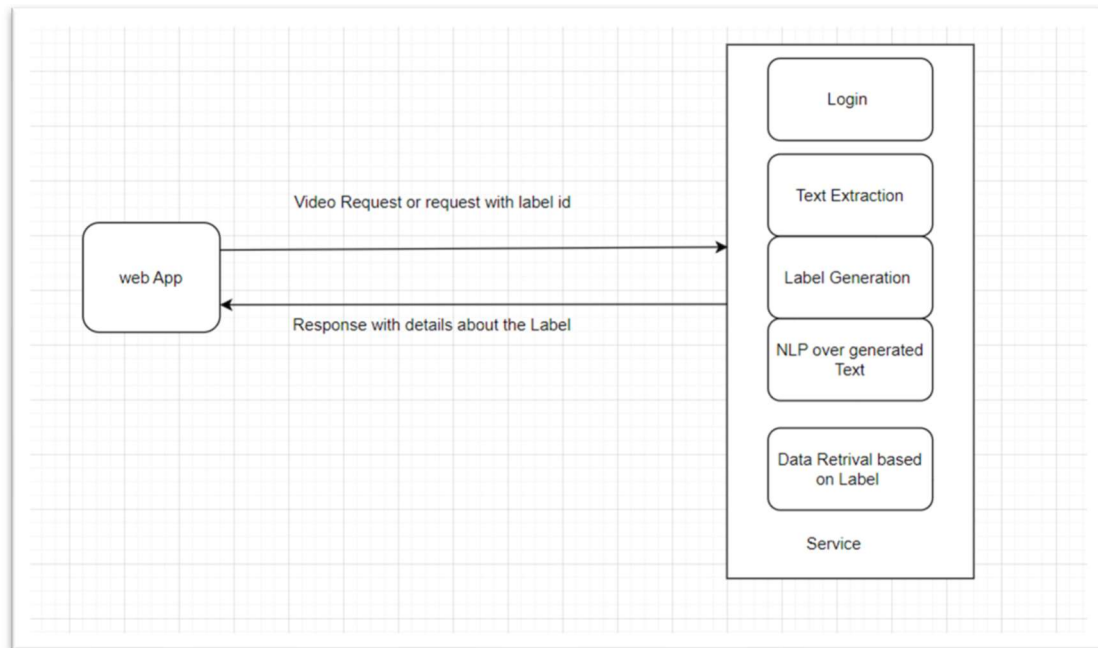
3. Proposed Solution and Workflow:



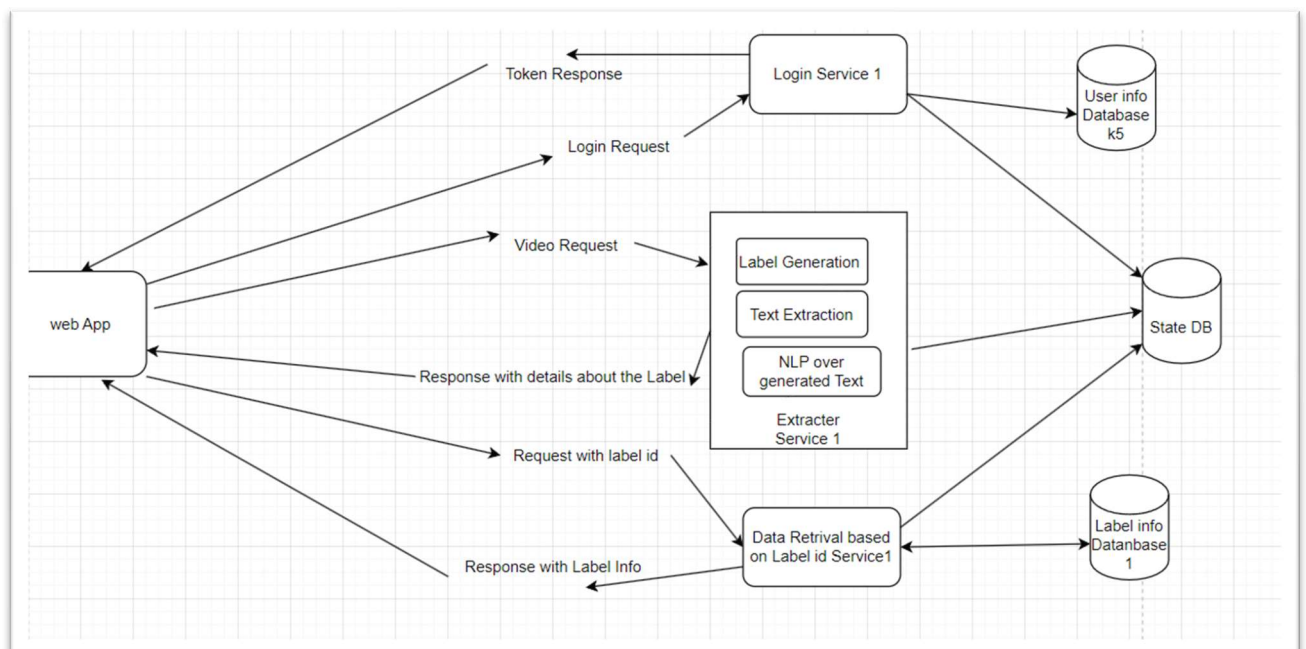
4. System Design:

4.1 High Level System Design:

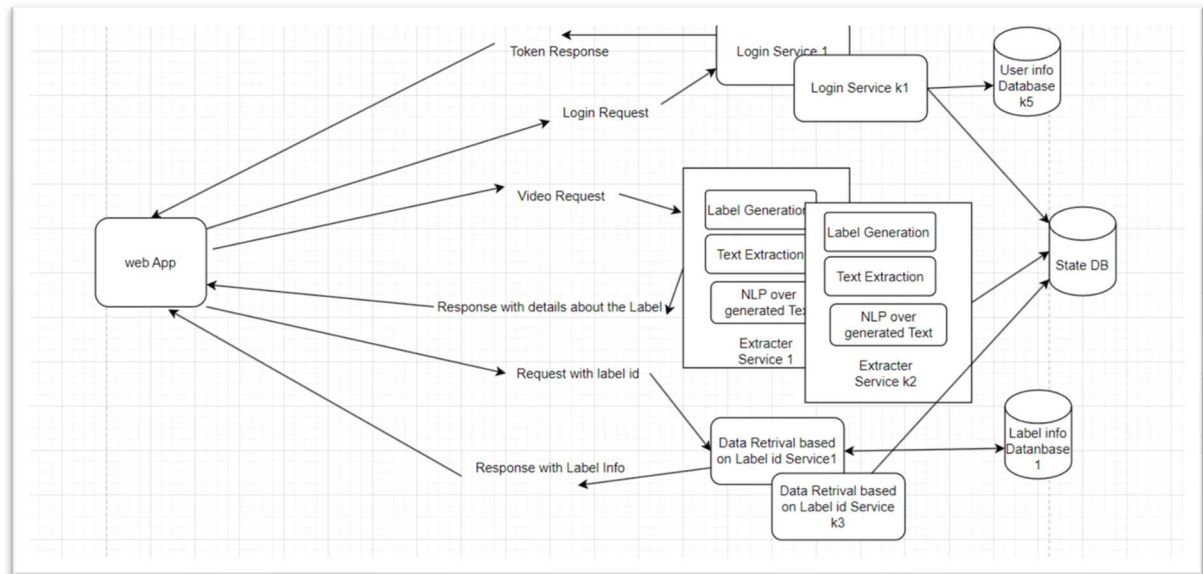
- System design has been done keeping in mind some of the quality attributes



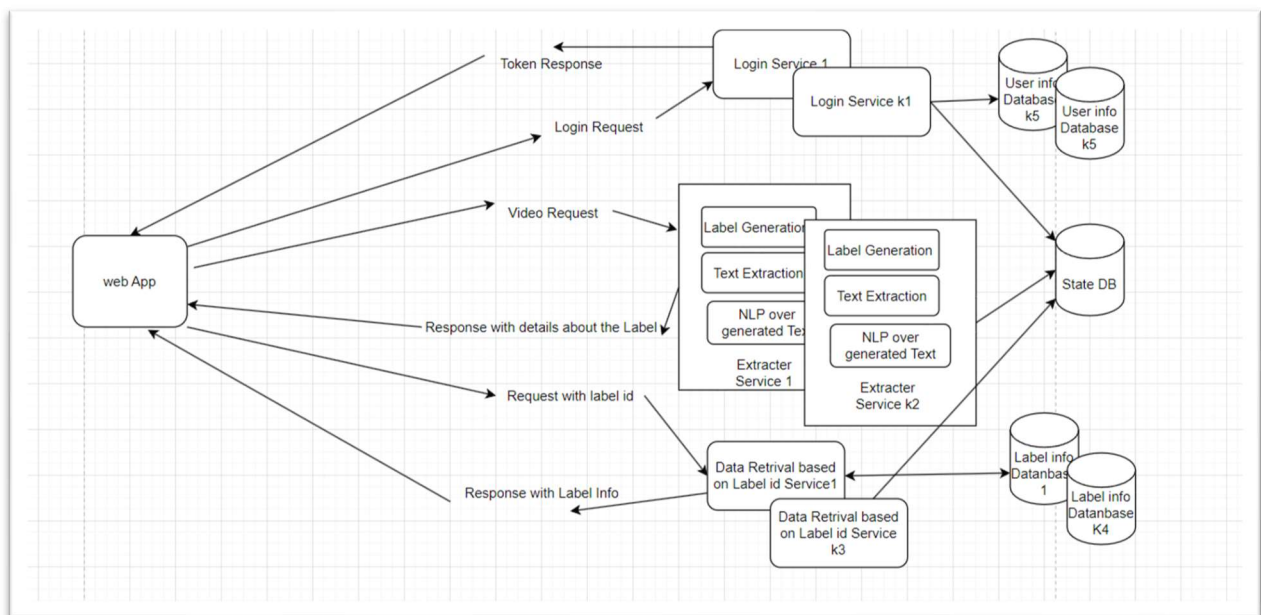
- **Splitting Services:** By separating the login, extraction, and data retrieval functionalities, you can optimize each service independently, leading to better performance.



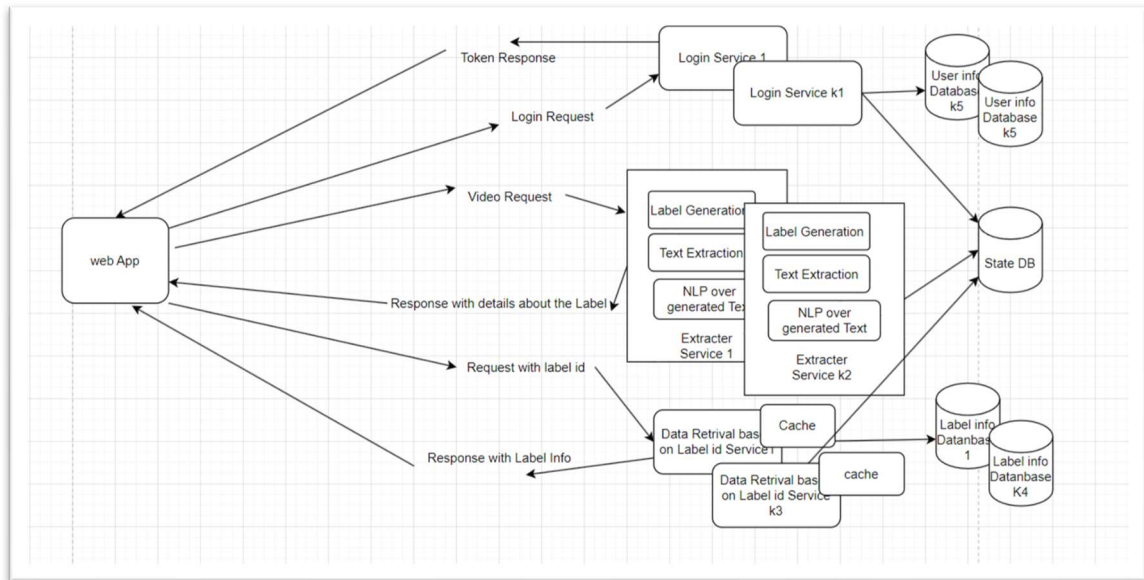
- **Multiple Servers:** Deploying multiple servers for each service can ensure that if one server fails, others can take over, thus improving availability.



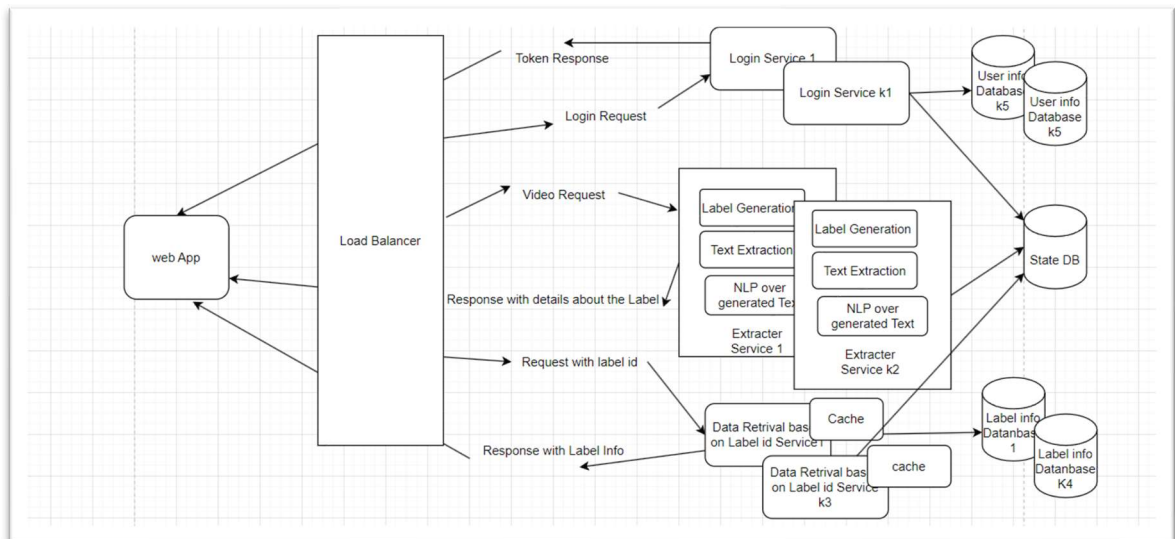
- **Database Replication:** Having multiple copies of the database can prevent data loss and ensure continuous data availability even in the event of a server failure.



- **Caching:** Implementing cache memory can significantly reduce data access times, enhancing the system's responsiveness.

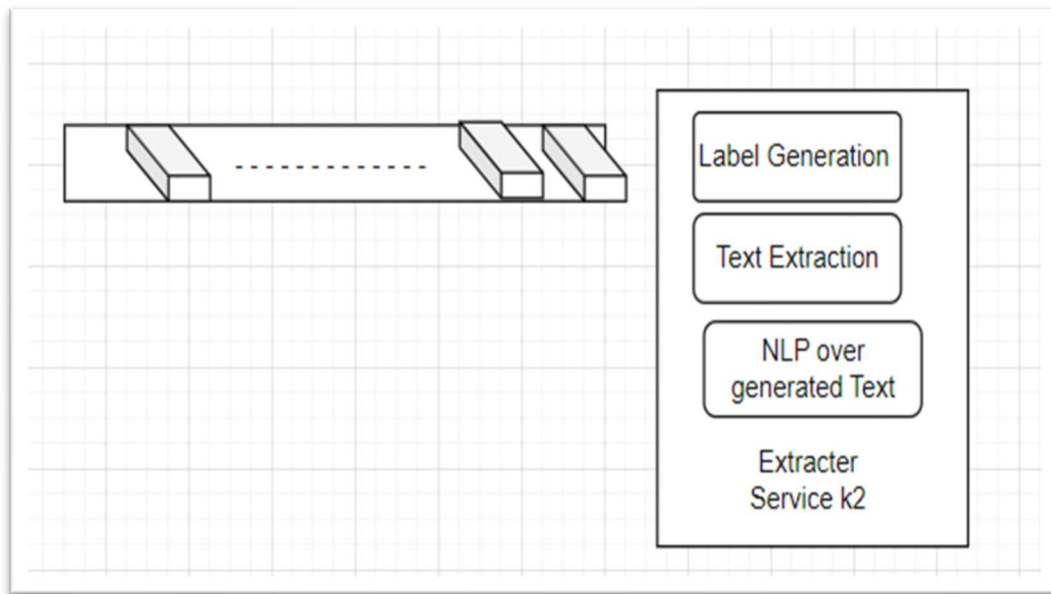


- **Load Balancing:** Introducing a load balancer can distribute incoming traffic across multiple servers, preventing any single server from becoming a bottleneck.

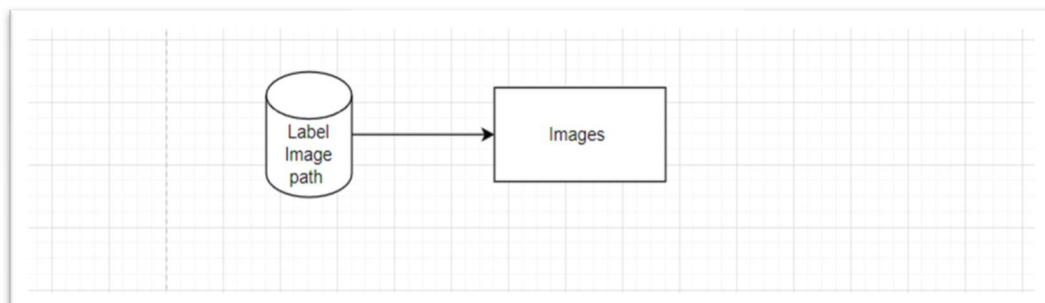


4.2 Low Level System Design:

- **Asynchronicity:** Handling video requests asynchronously is a wise choice since video processing can be resource-intensive and time-consuming.
- **Queue System:** Implementing a First-In-First-Out (FIFO) queue based on time will ensure that video requests are processed in the order they are received, which is fair and systematic.



- **Database for Paths:** Storing only the paths to images in the database is a good strategy to keep the database lightweight and **fast**.
- **File System for Images:** Keeping images in a file system is a standard practice that allows for **better manageability and scalability**.



5. Implementation:

5.1 User Authentication using PyJWT python library.

- [PyJWT](#) is a Python library that allows you to encode and decode JWTs. It provides functions for creating JWTs with custom claims, signing them using various algorithms (such as HS256), and verifying the integrity of received JWTs.
- The necessary user information is encoded into the JWT token itself, providing a lightweight and scalable authentication solution.
- Once the user is authenticated, you can use the extracted user identifier to authorize access to specific resources or perform certain actions.
- For example, you might check the user's permissions or roles stored in a database to determine if they have access to a particular endpoint or operation.

5.2 User Login/ Signup:

- we have used [react](#) for frontend development for login and signup page.
- The user credentials are saved in [MongoDB NoSQL](#) database.

5.3 Label Extraction: Creating a single image representation of the uploaded video that encapsulates its content using [OpenCV](#).

[Video Loading](#): The first step is to load the video file using OpenCV's video capture functionality. This allows you to access individual frames of the video.

[Frame Extraction](#): Once the video is loaded, you iterate through each frame of the video. OpenCV provides functions to read frames sequentially from the video file.

[Frame Processing](#): For each frame, you can perform any necessary processing or transformations. This could include resizing, cropping, adjusting brightness/contrast, or applying filters depending on the specific requirements of your application.

[Frame Stitching](#): As frames are processed, you combine them into a single larger image. This can be done by arranging the frames in a grid-like layout or by overlaying them on top of each other.

[Image Output](#): Once all frames have been stitched together, the resulting image can be saved to disk or displayed in a graphical user interface using OpenCV's image writing or display functions.

5.4 Text Extraction: Extraction of Text from the image using an OCR model called [Tesseract OCR](#).

- Tesseract is an open-source OCR engine maintained by Google. It is widely used for extracting text from images, documents, and other sources.
- Tesseract analyzes the image, identifies text regions, and attempts to recognize the characters within those regions.
- The OCR model processes the image using various techniques, such as character segmentation, feature extraction, and pattern recognition, to extract the text.
- Once the text extraction process is complete, Tesseract outputs the recognized text as a string or a structured format (e.g., plain text, JSON).

- The Tesseract OCR functionality is integrated into the application pipeline, typically through libraries or APIs.
- The application sends the composite image to the OCR model, receives the extracted text, and then utilizes this text for various purposes within the application.

5.5 Label Name Extraction: Extraction of Label Name from the Text generated in previous step.

- Integration of a Pre trained Natural Language Processing (NLP) models to extract medicine names and contents from the text extracted by the Tesseract OCR using [spaCy module](#).
- nlp is a language model object in spaCy, loaded with "[en_core_web_sm](#)", which is one of spaCy's English models. This model contains the binary weights, vocabulary, syntax, and entities necessary for running NLP tasks.
- "en_core_web_sm" is a small English model trained on written web text (blogs, news, comments), which includes part-of-speech tags, syntactic dependencies, and named entities.

5.6 storing label information: The information extracted in previous steps such as Label Name, content, image (we will store its path only) and the username of the user in [MongoDB NoSQL](#) database.

- The extracted medicine information obtained from the NLP Model is structured into JSON-like documents.
- The application that utilizes the medicine information stored in MongoDB can query the database to retrieve specific medicine details as needed.

5.7 Image Storage on Amazon AWS S3: The stitched image, which contains frames from the video, is saved on AWS S3. S3 is chosen for its scalability, reliability, and ease of use.

- By storing the image on S3, it can be accessed quickly and reliably from anywhere with an internet connection.

5.8 Search functionality Implementation: user can search labels previously added by other users or same user by their label names.

- First, we queried MongoDB database with the label name and then retrieve list of label information (content, image paths).
- In next steps we retrieve images from s3 buckets with the help given image path.

6. Results/Demo:


6.1 Login signup

Sign In
[Don't have an account? Sign Up](#)

6.2 Dashboard


Choose File No file chosen

Search



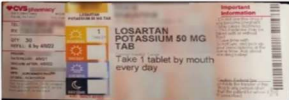
label1


Some extracted content here



label2

Some extracted content here



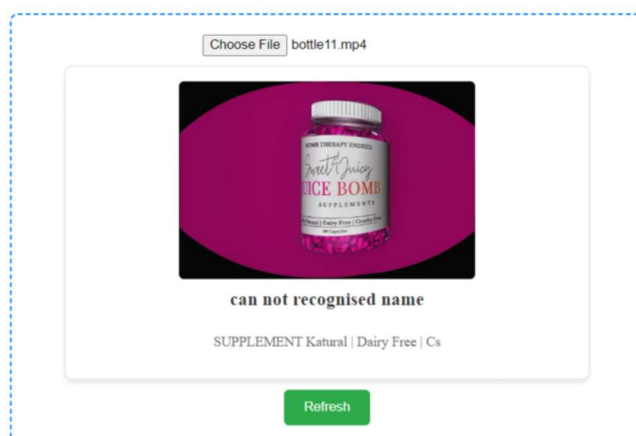


6.3 Uploading and processing video

Choose File bottle11.mp4

Processing...

6.4 final Result after processing



Search labels by name

Search

6.5 Search Functionality

label6

Search

Show All My Labels



label6

Some extracted content here



label6

Some extracted content here

GitHub Repository: <https://github.com/pratapavinesh/Label-Parser>

