A Project Report

On

"Document Validation System - ValidifyHUB"

Submitted to

Ericsson

In fulfilment of requirement for

Summer Industrial Internship

Submitted by

Uddipan Sarkar 21BCE0755,

3rd Year Student,

Vellore Institute of Technology, Vellore Campus

August 2023 – November 2023

CERTIFICATE

This is to certify that the project entitled "Document Validation System – ValidifyHUB" submitted by Mr. Uddipan Sarkar (3rd year student at Vellore Institute of Technology) for fulfilment of the Summer Industrial Internship at Ericsson is the authenticated work carried out by him and has undergone the requisite duration as prescribed by Ericsson for the internship.

Ms. Anamika Bannerjee – Project Director

Mr. Sunil Kumar Ray – Project Guide

Mr. Bidyut Das B – Project Guide

ACKNOWLEDGEMENT

I am grateful to Ms. Anamika Bannerjee (Project Director) for providing me with this opportunity, and who was a constant source of help and played a very instrumental role in the successful completion of the internship project.

I would like to thank Mr. Sunil Kumar Ray, and Mr. Bidyut Das B, who were my mentors during the internship. They were supportive throughout the internship and their constant guidance was of great help in the completion of the project.

I would also like to thank Ericsson for providing me with this incredible internship opportunity and for providing me with all the necessary tools for the successful completion of the project.

CONTENTS:

1. INTRODUCTION

1.1 MOTIVATION

This project is on the topic – "Document Validation". During our day-to-day lives, we come across several websites where we must upload documents of some kind to achieve our objectives or complete our tasks. Hence, a need arises to verify whether the correct document has been uploaded or not. The web application that has been designed will handle this task of document verification. It will not allow the user to upload any false or incorrect documents based on pre-defined criteria, thereby reducing the possibility of false or incorrect data.

1.2 PROBLEM STATEMENT

We live in the digital age. In such an age, it is very hard to be competitive without always keeping oneself updated about technology. In our regular lives, we encounter so many web applications where we must upload documents of some kind. These documents must be verified before inserting data into the database. While the available verification systems check for the file type that is being uploaded, they do not check the actual contents of the file. Here lies our problem statement, in solution to which we propose to build a document validation system.

Drawbacks of Present Systems:

1. They do not check the actual contents of the file.

2. They allow the user to upload any irrelevant file if the file type is accepted.

1.3 SCOPE AND OBJECTIVE

The scope of the project is to build a document validation web application that will read the contents of the file being uploaded and verify the validity of the file.

The objectives that are being achieved are –

1. The contents of the file are being verified.

2. It prevents the owner from uploading any file which does not meet the specific requirements of the file supposed to be uploaded.

The project provides a solution to the issue of document verification. It verifies and validates all files that are being uploaded. Only upon successful validation of the file, the data is uploaded into the database.

1.4 LIMITATIONS

For each file that must be uploaded, a set of pre-defined criteria must be set respectively by the administrator. There is yet no common policy for the different files. Separate rules must be created to cater to the diverse requirements of different files that might need to be uploaded.

3 sample documents have been included for the demonstration. Separate rules have been coded in for each of the 3 documents. Till now, we have not been able to find a way which encompasses the validation of all different kinds of files that might need to be uploaded.

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

None of the existing software services provide validation of the data which is present inside the document. The software only verifies the file type of the file that is to be submitted. This makes it possible for users to upload an irrelevant file with the correct file type, hence reducing the dependability of the system. The data is then extracted from the uploaded file and stored in the database.

The main drawbacks are –

1. User Interface – User interface can always be improved for the operators.

2. Document Validation – The data inside the file is not being verified by the software. Incorrect files may be uploaded.

3. Error Handling – Errors in the case of incorrect document submission are not pointed out and handled.

2.2 PROPOSED SYSTEM

The proposed software intends to rectify the drawbacks encountered in the existing software systems. The file type of the document, as well as the contents of the document are verified and validated to make sure that the correct document is being uploaded. In case of an incorrect submission, the proper error is pointed out by the software so that necessary rectifications can be made.

Advantages:

1. The content of the document is being verified and validated.

2. Error handling is done properly in case of incorrect submissions.

3. The User Interface has been improved.

3. SYSTEM ANALYSIS

3.1 THEORITICAL BACKGROUND OF THE SOFTWARE

1. Python programming language:

Python, a versatile and readable programming language, is renowned for its simplicity and broad applicability. Its clear syntax, emphasizing readability, appeals to developers, and its dynamic typing enables flexible and concise coding. Python's extensive standard library minimizes external dependencies, and its vibrant community supports a rich ecosystem of third-party libraries. Widely used in web development, data analysis, and AI, Python's cross-platform compatibility and support for multiple programming paradigms contribute to its popularity. Specialized frameworks like Django and Flask cater to distinct needs, reinforcing Python's position as a preferred language across diverse domains, driven by its ease of use and expansive community contributions.

Use case: The entire software has been designed using python as the programming language. All the different frameworks and libraries in python have been implemented to achieve the required functionalities.

2. Amazon Textract:

Amazon Textract is a fully managed optical character recognition (OCR) service by Amazon Web Services (AWS). It excels in extracting text, forms, and tables from scanned documents in various formats, such as PDFs. Leveraging machine learning algorithms, Textract intelligently analyses documents, identifying key elements and extracting structured data. Its versatility extends to diverse document types, enhancing efficiency in document processing workflows. Developers benefit from its easy integration with AWS services, allowing seamless incorporation into applications for tasks like content indexing, data analysis, and document management. Amazon Textract empowers businesses by automating document processing, reducing manual effort, and accelerating information extraction from a variety of documents, fostering improved productivity and data accessibility.

Use case: Amazon Textract has been used in the software to extract the data from the document for further processing to achieve required functionalities. Amazon Textract has been used by linking the AWS services with the system using python.

3. Flask:

Flask is a lightweight and flexible web framework for Python, designed to make web development simple and efficient. Known for its minimalist design, Flask provides essential tools for building web applications without imposing rigid structures. With a clear and intuitive routing system, developers can map URLs to functions easily, defining the behavior of different endpoints. Flask incorporates the Jinja2 templating engine for dynamic HTML

template creation, making it straightforward to generate dynamic content. While it comes with a built-in development server, Flask is often deployed with production-ready servers like Gunicorn. Its modularity allows the integration of extensions for additional functionalities, and its support for RESTful principles makes it ideal for developing APIs. Flask's simplicity, coupled with a vibrant community, has made it a popular choice for web development in Python.

Use case: The software application has been created and deployed locally on the system using the Flask framework within Python.

4. MySQL Database:

MySQL is an open-source relational database management system (RDBMS) widely used for efficiently storing and retrieving structured data. It is known for its robust performance, scalability, and reliability, making it a popular choice for various applications, from small-scale projects to large enterprise solutions. MySQL follows the SQL (Structured Query Language) standard, providing a powerful and comprehensive set of features for managing databases. It supports ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring data integrity. With features like transactions, indexes, and stored procedures, MySQL facilitates complex data operations. Its versatility extends to compatibility with different programming languages and operating systems. MySQL is favored for its ease of use, strong community support, and seamless integration with popular web development frameworks and applications.

Use Case: The relevant information is filtered out from the data present in the document and then it is stored in a MySQL Database. The database is linked to my software via a python code.

5. Ajax:

AJAX (Asynchronous JavaScript and XML) is a web development technique that enables seamless, asynchronous data exchange between a web browser and a server. Instead of reloading the entire page, AJAX allows specific portions to be updated, enhancing user experience by providing dynamic, responsive content. This is achieved through the XMLHttpRequest object or the more modern Fetch API in JavaScript. AJAX is not limited to XML and can handle various data formats like JSON. Widely used in creating interactive and fast-loading web applications, AJAX has become a fundamental tool for developers seeking to improve the fluidity and efficiency of web pages by fetching and updating data in the background without requiring a full page reload.

Use Case: Ajax is used to handle multiple file uploads and submissions in an efficient manner in a single webpage, thereby increasing the usability of the software website.

3.2 SYSTEM REQUIREMENTS

The system specifications required have been mentioned below:

Windows:

Microsoft® Windows® 7/8/10 (64-bit)

4 GB RAM minimum, 8 GB RAM recommended.

2 GB of available disk space minimum,

4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)

1280 x 800 minimum screen resolution

Mac:

Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)

4 GB RAM minimum, 8 GB RAM recommended.

2 GB of available disk space minimum,

4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)

1280 x 800 minimum screen resolution

Linux:

GNOME or KDE desktop - Tested on gLinux based on Debian.

64-bit distribution capable of running 32-bit applications

GNU C Library (glibc) 2.19 or later

4 GB RAM minimum, 8 GB RAM recommended.

2 GB of available disk space minimum,

4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)

1280 x 800 minimum screen resolution

Chrome OS:

8 GB RAM or more recommended

4 GB of available disk space minimum

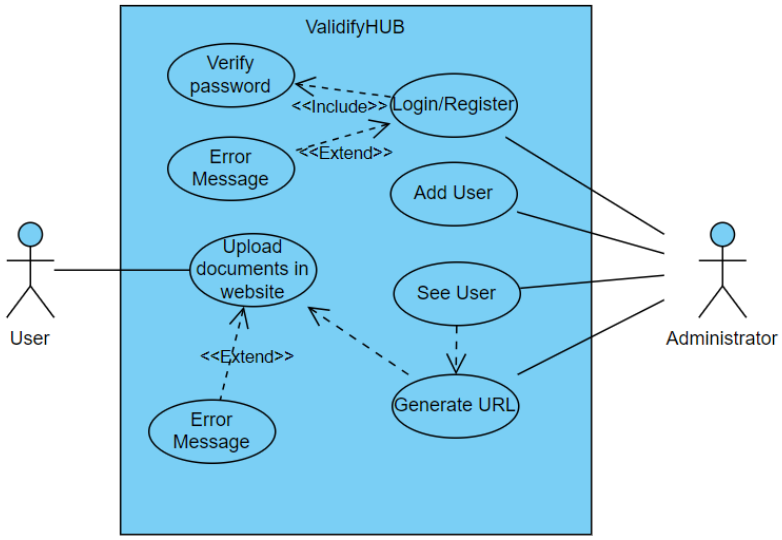
1280 x 800 minimum screen resolution

Intel i5 or higher (U series or higher) recommended.

4 SYSTEM DESIGN

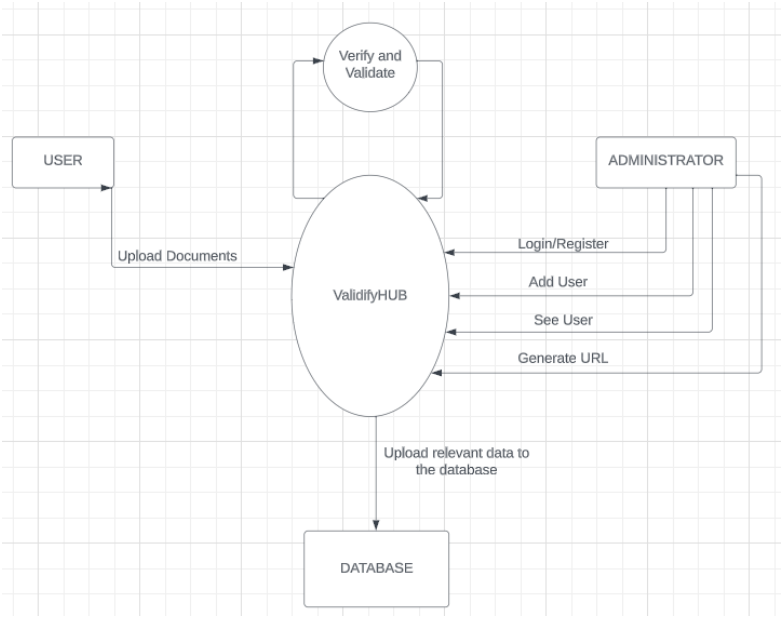
4.1 USE CASE

A UML use case diagram is the primary form of system (or software) requirement for a new software under development. The following use case diagram denotes the requirements to be achieved by the software.



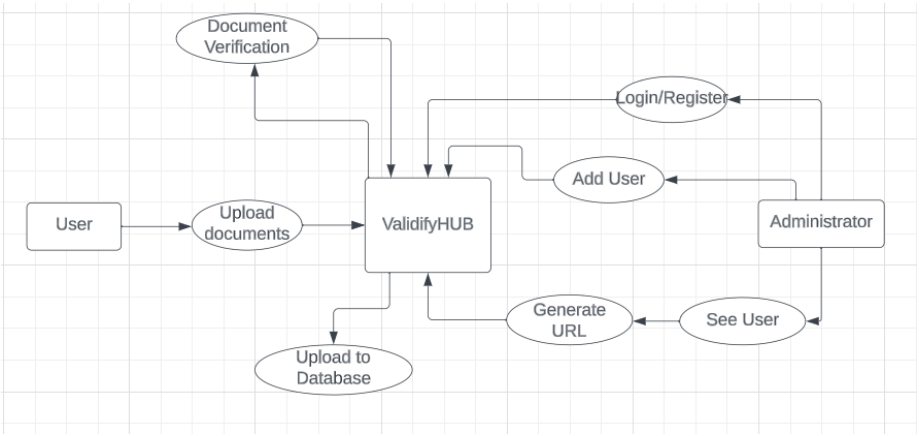
4.2 CONTEXT DIAGRAM

Context diagrams show the interactions between a system and other actors with which the system is designed to interface with. They clarify the context in which the system will be a part. The context diagram for the proposed software has been illustrated below.



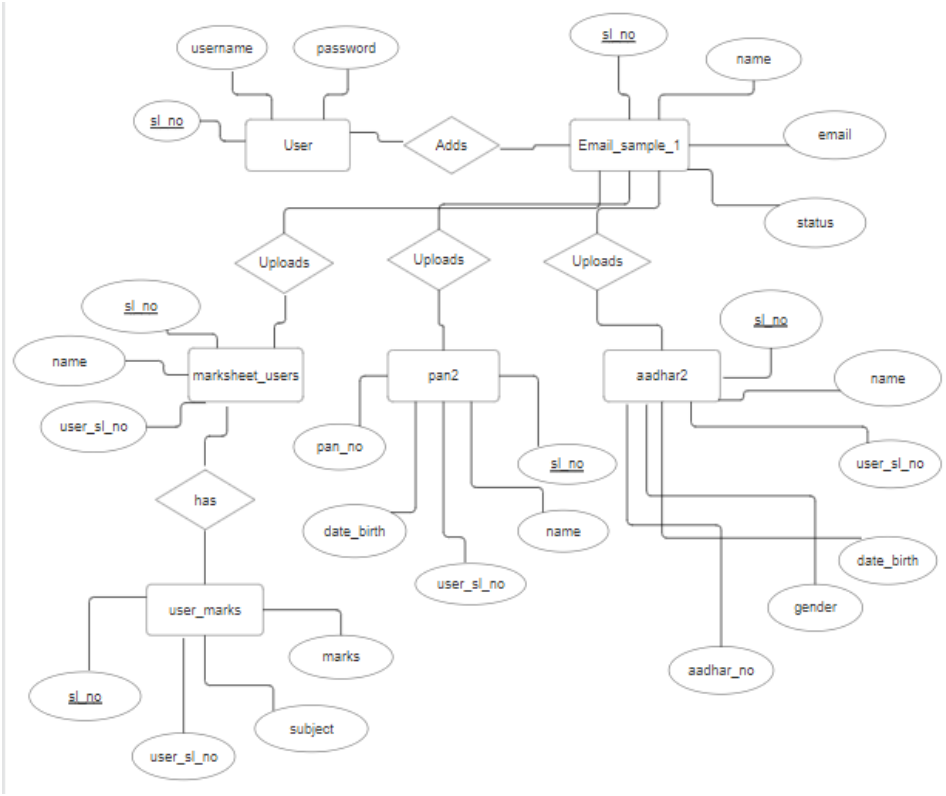
4.3 DATA FLOW DIAGRAM

A data flow diagram maps out the flow of information for any process or system. It uses defined symbols and texts to show data inputs, outputs, storage points and routes. They are used to analyse the software model.



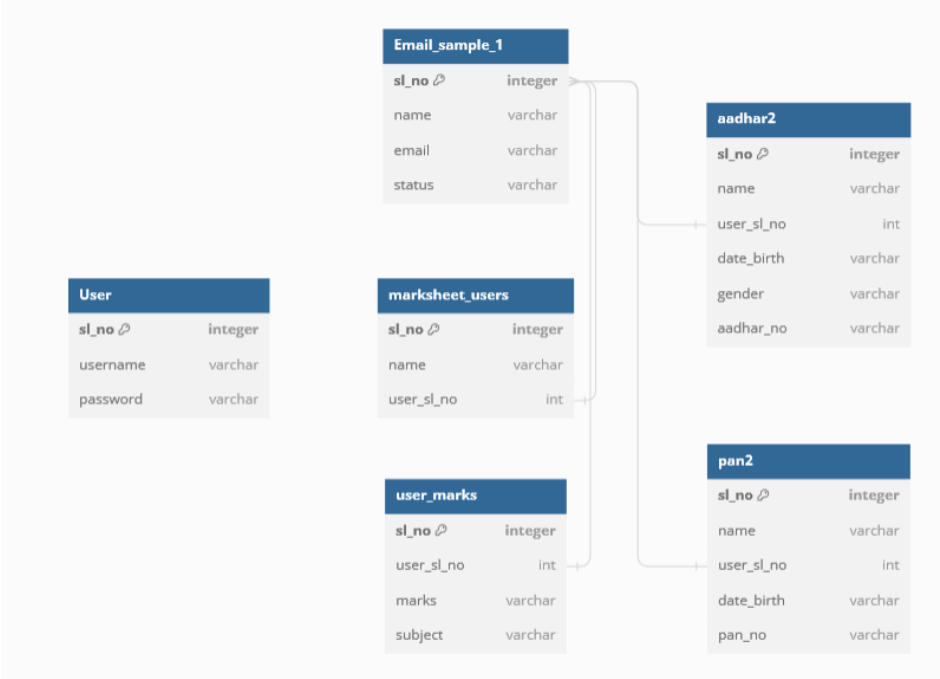
4.4 ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship Diagram displays the relationship between the various entity sets stored in the database. It helps to explain the logical structure of the database. The purpose of this diagram is to represent the entity framework infrastructure.



4.5 DATABASE SCHEMA

A database schema refers to the logical and visual configuration of the entire relational database. The database objects are often grouped and displayed as tables, functions, and relations. A schema describes the organization and storage of data in a database and defines the relationship between various tables.



5. FUNCTIONS AND TESTING

5.1 KEY FUNCTIONS

1. Register a new administrative user.
2. Login for the administrative user with credentials.
3. Add User option for the administrative user.
4. See User option for the administrative user.
5. Generate URL option for each candidate user.
6. Upload documents option for the candidate users.

5.2 TEST CASES AND REPORT

Sl. No.	Testcase Name	Expected Output	Actual Output	Result	Priority
1.	Register a new administrative user.	New administrative user should be registered.	Working as expected.	Successful	High
2.	Login for the administrative user with credentials.	Administrative User should be logged in.	Working as expected.	Successful	High
3.	Add User option for the administrative user.	New candidate user should be added to the list.	Working as expected.	Successful	High
4.	See User option for the administrative user.	All the candidate users should be listed.	Working as expected.	Successful	High
5.	Upload documents option for the candidate users.	The candidate users should be able to successfully upload correct documents. (Errors should be displayed)	Working as expected.	Successful	High

6. IMPLEMENTATION AND RESULTS

6.1 PROJECT LIFECYCLE

The Project Lifecycle is a sequential process consisting of distinct phases: initiation, analysis, planning, design, execution, testing, and closure.

1. **Initiation:** This marks the project's beginning and involves activities like feasibility studies, scoping, stakeholder identification, creating a statement of work, and establishing a timeline.
2. **Analysis:** In this phase, a detailed analysis is conducted based on project requirements. For mapping development, existing mappings are backtracked to identify dependencies.
3. **Planning:** Once selected, the project moves into the planning stage. Here, a comprehensive project plan is created, detailing tasks, schedules, resources, and project constraints.
4. **Design Phase:** This involves preparing a rough design for the mapping, deciding on transformations, and incorporating Firebase database elements.
5. **Execution:** The actual work takes place in this phase. If working alone, as in this case, meticulous task management is crucial to ensure a smooth workflow. Monitoring and controlling are integral to address issues promptly.
6. **Testing:** Various testing types are conducted:
    - *Unit Testing:* Scrutinizing the smallest testable parts of the application during development to ensure proper operation.
    - *Integration Testing:* Testing combined units as a group to expose faults in their interaction.
    - *User Acceptance Testing (UAT):* Testing by end users to verify and accept the software system before production. This includes validation of the end-to-end flow and is carried out in a separate environment with production-like data setup.
7. **Project Closure:** This marks the completion of coding and testing. Documentation is handed over to the maintenance team, and the project's performance is analyzed to determine goal achievement, including task completion, adherence to timelines, and staying within budget constraints for organizational projects.

6.2 PROJECT CODES

Python Codes:

1. extract_module.py:

```
from flask import request
import extract_module
import logic_module_2
import database_module_2

def upload_and_process(pdf_file, process_function, send_function, sl_no):
    if pdf_file:
        status, extracted_text = extract_module.upload_and_extract_text(pdf_file)
        if status == 'FAILED' or status == 'PARTIAL_SUCCESS':
            return 'Extraction Failed'
        else:
            result = process_function(extracted_text)
            if result[0] == 'FAILED':
                return 'Incorrect PDF'
            else:
                if send_function:
                    send_function(result, sl_no)
                return 'Successfully Uploaded'
    return 'No File Uploaded'

def process_aadhar(extracted_text):
    return logic_module_2.aadhar_details(extracted_text)

def process_pan(extracted_text):
    return logic_module_2.pan_details(extracted_text)

def process_marksheet(extracted_text):
    return logic_module_2.marksheet_details(extracted_text)

def send_to_database_aadhar(result, sl_no):
    name, dob, gender, num = result[1], result[2], result[3], result[4]
    database_module_2.send_to_database_aadhar(name, dob, gender, num, sl_no)

def send_to_database_pan(result, sl_no):
    name, dob, pan = result[1], result[2], result[3]
    database_module_2.send_to_database_pan(name, dob, pan, sl_no)

def send_to_database_marksheet(result, sl_no):
    name = result[1]
    subjects_and_marks = result[2]
    database_module_2.send_to_database_marksheet(name, subjects_and_marks, sl_no)
```

2. logic_module_2.py:

```
import re

def aadhar_details(extracted_text):
        pattern = r'\d{4} \d{4} \d{4}'
        has_12_digit_number = re.search(pattern, extracted_text)

        if has_12_digit_number:
                status = 'SUCCEEDED'
                num = has_12_digit_number.group()
                date_pattern = r'\d{2}/\d{2}/\d{4}'
                dob_match = re.search(date_pattern, extracted_text)

                lines = extracted_text.split('\n')
                dob = dob_match.group()
                name_index = None
                for i, line in enumerate(lines):
                    if re.search(date_pattern, line) and i > 0:
                        name_index = i-1
                        break

                name = lines[name_index]
                gender_pattern = r'\b(MALE|FEMALE)\b'
                gender_match = re.search(gender_pattern, extracted_text, re.IGNORECASE)
                gender = gender_match.group().upper()
                return status, name, dob, gender, num
        else:
                status = 'FAILED'
                return status,"","","",""

def pan_details(extracted_text):
        pattern = r'[A-Z]{5}\d{4}[A-Z]{1}'
        has_pan = re.search(pattern, extracted_text)

        if has_pan:
                status = 'SUCCEEDED'
                pan = has_pan.group()
                date_pattern = r'\b(\d{2})[-/](\d{2})[-/](\d{4})\b'
                date_match = re.search(date_pattern, extracted_text)
                day, month, year = date_match.groups()
                dob = f"{day}/{month}/{year}"

                lines = extracted_text.split('\n')
                name_index = None
                i=0
                for i, line in enumerate(lines):
                    if re.search('NAME', line) or re.search('Name',line):
                        name_index=i+1
                        break
```

```
            name = lines[name_index]
            return status, name, dob, pan
    else:
            status = 'FAILED'
            return status,"","",""



def marksheet_details(extracted_text):
    pattern_1 = r'COUNCIL FOR THE INDIAN SCHOOL CERTIFICATE
EXAMINATIONS'
    pattern_3 = r'\b(\d{7})\b'
    pattern_2 = r'CENTRAL BOARD OF SECONDARY EDUCATION'
    pattern_4 = r'\b(\d{8})\b'

    match_1 = re.search(pattern_1, extracted_text)
    match_3 = re.search(pattern_3, extracted_text)
    match_2 = re.search(pattern_2, extracted_text)
    match_4 = re.search(pattern_4, extracted_text)

    if match_1 and match_3:
        status = 'SUCCEEDED'
        # 1. Extract Name
        name_pattern = r'Name ([A-Z\s]+)'
        name_match = re.search(name_pattern, extracted_text)
        name = name_match.group(1).strip()

        # 2. Find the index of the line containing the word "ENGLISH"

        english_line_number = None
        lines = extracted_text.split('\n')
        for i, line in enumerate(lines):
            if 'ENGLISH' in line:
                english_line_number = i
                break

        # 3. Extract subjects and marks
        subjects_and_marks = []
        for i in range(english_line_number, english_line_number+12, 2):
            subject_line = lines[i].strip()
            marks_line = lines[i + 1].strip() if i + 1 < len(lines) else None

            subjects_and_marks.append({
                'subject': subject_line,
                'marks': marks_line
            })
```

```
            return status, name, subjects_and_marks

        elif match_2 and match_4:
            status = 'SUCCEEDED'
            # 1. Extract Name
            name_pattern = r'This is to certify that ([A-Z\s]+)'
            name_match = re.search(name_pattern, extracted_text)
            name = name_match.group(1).strip()
            name = name[:-1]

            # 2. Find the index of the line containing the word "ENGLISH"
            english_line_number = None
            lines = extracted_text.split('\n')
            for i, line in enumerate(lines):
                if 'ENGLISH CORE' in line:
                    english_line_number = i
                    break

            # 3. Extract subjects and marks
            subjects_and_marks = []
            for i in range(english_line_number, english_line_number+42, 7):
                subject_line = lines[i].strip()
                marks_line = lines[i + 3].strip() if i + 3 < len(lines) else None

                subjects_and_marks.append({
                    'subject': subject_line,
                    'marks': marks_line
                })

            return status, name, subjects_and_marks

        else:
            status = 'FAILED'
            return status, "", ""
```

3. database_module_2.py:

```
import mysql.connector

def send_to_database_aadhar(name, dob, gender, num, sl_no):
    host = "localhost"
    user = "Uddipan"
    password = "Dipto#1803"
    database = "ericsson_project"

    connection = mysql.connector.connect(
        host=host,
        user=user,
```

```
        password=password,
        database=database
    )

    if connection.is_connected():
        print("Connected")

    cursor = connection.cursor()

    insert_query = "INSERT INTO aadhar2 (name, user_sl_no, date_birth, gender, aadhar_no)
VALUES (%s, %s, %s, %s, %s)"
    data_to_insert = ( name, sl_no,  dob, gender, num)

    cursor.execute(insert_query, data_to_insert)
    connection.commit()
    cursor.close()
    connection.close()

def send_to_database_pan(name, dob, pan, sl_no):
    host = "localhost"
    user = "Uddipan"
    password = "Dipto#1803"
    database = "ericsson_project"

    connection = mysql.connector.connect(
        host=host,
        user=user,
        password=password,
        database=database
    )

    if connection.is_connected():
        print("Connected")

    cursor = connection.cursor()

    insert_query = "INSERT INTO pan2 (name, user_sl_no, date_birth, pan_no) VALUES
(%s, %s, %s, %s)"
    data_to_insert = ( name, sl_no, dob, pan)

    cursor.execute(insert_query, data_to_insert)
    connection.commit()
    cursor.close()
    connection.close()

def send_to_database_marksheet(name, subjects_and_marks, sl_no):
    host = "localhost"
    user = "Uddipan"
```

```
    password = "Dipto#1803"
    database = "ericsson_project"

    connection = mysql.connector.connect(
        host=host,
        user=user,
        password=password,
        database=database
    )

    if connection.is_connected():
        print("Connected")

    cursor = connection.cursor()
    insert_query = "INSERT INTO marksheet_users (name, user_sl_no) VALUES (%s, %s)"
    data_to_insert = (name, sl_no)
    cursor.execute(insert_query, data_to_insert)
    connection.commit()

    for entry in subjects_and_marks:
        subject = entry['subject']
        marks = entry['marks']

        # SQL query to insert values into the database
        insert_query = "INSERT INTO user_marks (user_sl_no, subject, marks) VALUES (%s,
%s, %s)"

        # Execute the query with the values
        cursor.execute(insert_query, (sl_no, subject, marks))
        connection.commit()

    cursor.close()
    connection.close()
```

4. common_module_2.py:

```
from flask import request
import extract_module
import logic_module_2
import database_module_2

def upload_and_process(pdf_file, process_function, send_function, sl_no):
    if pdf_file:
        status, extracted_text = extract_module.upload_and_extract_text(pdf_file)
        if status == 'FAILED' or status == 'PARTIAL_SUCCESS':
            return 'Extraction Failed'
        else:
```

```
            result = process_function(extracted_text)
            if result[0] == 'FAILED':
                return 'Incorrect PDF'
            else:
                if send_function:
                    send_function(result, sl_no)
                return 'Successfully Uploaded'
    return 'No File Uploaded'

def process_aadhar(extracted_text):
    return logic_module_2.aadhar_details(extracted_text)

def process_pan(extracted_text):
    return logic_module_2.pan_details(extracted_text)

def process_marksheet(extracted_text):
    return logic_module_2.marksheet_details(extracted_text)

def send_to_database_aadhar(result, sl_no):
    name, dob, gender, num = result[1], result[2], result[3], result[4]
    database_module_2.send_to_database_aadhar(name, dob, gender, num, sl_no)

def send_to_database_pan(result, sl_no):
    name, dob, pan = result[1], result[2], result[3]
    database_module_2.send_to_database_pan(name, dob, pan, sl_no)

def send_to_database_marksheet(result, sl_no):
    name = result[1]
    subjects_and_marks = result[2]
    database_module_2.send_to_database_marksheet(name, subjects_and_marks, sl_no)
```

5. combined_7.py:

```
from flask import Flask, render_template, request, redirect, url_for, session, jsonify, flash
import mysql.connector
import os
import common_module_2

app = Flask(__name__)
app.secret_key = "abcd"

# MySQL Database Configuration
db_config = {
    "host": "localhost",
    "user": "Uddipan",
    "password": "Dipto#1803",
    "database": "ericsson_project"
}
```

```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        # Check if the username is already taken
        try:
            conn = mysql.connector.connect(**db_config)
            cursor = conn.cursor(dictionary=True)
            query = "SELECT username FROM user WHERE username = %s"
            cursor.execute(query, (username,))
            existing_user = cursor.fetchone()

            if existing_user:
                flash("Username is already taken. Please choose a different one.", 'danger')
            else:
                query = "INSERT INTO user (username, password) VALUES (%s, %s)"
                cursor.execute(query, (username, password))
                conn.commit()
                cursor.close()
                conn.close()
                flash("Registration successful! You can now log in.", 'success')
                return redirect(url_for('login'))
        except mysql.connector.Error as err:
            return f"Error: {err}"

    return render_template('register.html')

# Define a route for user login
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        try:
            conn = mysql.connector.connect(**db_config)
            cursor = conn.cursor(dictionary=True)
            query = "SELECT username, password FROM user WHERE username = %s"
            cursor.execute(query, (username,))
            user = cursor.fetchone()

            if user and user['password'] == password:
                session['logged_in'] = True
                session['username'] = user['username']
                flash("Login successful!", 'success')
```

```
            return redirect(url_for('admin_panel'))
        else:
            flash("Login failed. Please check your credentials.", 'danger')

    except mysql.connector.Error as err:
        return f"Error: {err}"

  return render_template('login.html')

@app.route("/admin_panel", methods=["GET", "POST"])
def admin_panel():
  if not session.get('logged_in'):
    return redirect(url_for('login'))
  users = []

  if request.method == "POST" and "see_users" in request.form:
    try:
      # Connect to the MySQL database
      conn = mysql.connector.connect(**db_config)
      cursor = conn.cursor(dictionary=True)

      # SQL query to fetch records from email_sample_1 table with initial status as 'not
verified'
      query = "SELECT sl_no, name, email, status FROM email_sample_1"
      cursor.execute(query)
      users = cursor.fetchall()

      # Close the database connection
      cursor.close()
      conn.close()

    except mysql.connector.Error as err:
      return f"Error: {err}"

  return render_template("admin_sample_1.html", users=users)

@app.route('/user_registration')
def user_registration():
  return render_template('add_user.html')

@app.route('/add_user', methods=['POST'])
def add_user():
  if request.method == 'POST':
    name = request.form['name']
    email = request.form['email']

    try:
      # Connect to the MySQL database
```

```
            conn = mysql.connector.connect(**db_config)
            cursor = conn.cursor()

            # Insert the user's name and email into table
            query = "INSERT INTO email_sample_1 (name, email, status) VALUES (%s, %s,
'not verified')"
            cursor.execute(query, (name, email))

            # Commit the changes to the database
            conn.commit()

            # Close the database connection
            cursor.close()
            conn.close()

            # Redirect to the admin panel
            return redirect(url_for('admin_panel'))

        except mysql.connector.Error as err:
            return f"Error: {err}"

    return "Invalid request"

@app.route("/generate_url/<int:sl_no>")
def generate_url(sl_no):
    # Store the user's sl_no and email in the session
    session['sl_no'] = sl_no

    # Generate a unique URL for the user
    unique_url = url_for("user_page", sl_no=sl_no, _external=True)

    # Return the generated URL as text
    return f"Generated URL for user with ID {sl_no} : <a
href='{unique_url}'>{unique_url}</a>"


@app.route('/user/<int:sl_no>')
def user_page(sl_no):
    return render_template('upload2_2.html', sl_no=sl_no)

@app.route('/upload_aadhar', methods=['POST'])
def upload_aadhar_file():
    pdf_file = request.files.get('pdf_file')
    sl_no = session.get('sl_no')

    if pdf_file:
        status_message = common_module_2.upload_and_process(pdf_file,
common_module_2.process_aadhar, common_module_2.send_to_database_aadhar, sl_no)
        if status_message == "Successfully Uploaded":
```

```
            session['aadhar_submitted'] = True
            error_message_aadhar = ""
        else:
            error_message_aadhar = f'Aadhar Error: {status_message}'
    else:
        # No file was uploaded, set an error message
        error_message_aadhar = "No Aadhar file uploaded"

    aadhar_submitted = not bool(error_message_aadhar)
    session['pan_submitted'] = session.get('pan_submitted', False)
    session['marksheet_submitted'] = session.get('marksheet_submitted', False)

    response_data = {
        'aadhar_submitted': aadhar_submitted,
        'pan_submitted': session.get('pan_submitted', False),
        'marksheet_submitted': session.get('marksheet_submitted', False),
        'error_message_aadhar': error_message_aadhar,
        'error_message_pan': session.get('error_message_pan', ''),
        'error_message_marksheet': session.get('error_message_marksheet', ''),
        'sl_no': sl_no
    }

    return jsonify(response_data)


@app.route('/upload_pan', methods=['POST'])
def upload_pan_file():
    pdf_file = request.files.get('pdf_file')
    sl_no = session.get('sl_no')

    if pdf_file:
        status_message = common_module_2.upload_and_process(pdf_file,
common_module_2.process_pan, common_module_2.send_to_database_pan, sl_no)
        if status_message == "Successfully Uploaded":
            session['pan_submitted'] = True
            error_message_pan = ""
        else:
            error_message_pan = f'PAN Error: {status_message}'
    else:
        # No file was uploaded, set an error message
        error_message_pan = "No PAN file uploaded"

    pan_submitted = not bool(error_message_pan)
    session['aadhar_submitted'] = session.get('aadhar_submitted', False)
    session['marksheet_submitted'] = session.get('marksheet_submitted', False)

    response_data = {
        'aadhar_submitted': session.get('aadhar_submitted', False),
        'pan_submitted': pan_submitted,
```

```
        'marksheet_submitted': session.get('marksheet_submitted', False),
        'error_message_aadhar': session.get('error_message_aadhar', ''),
        'error_message_marksheet': session.get('error_message_marksheet',''),
        'error_message_pan': error_message_pan,
        'sl_no': sl_no
    }

    return jsonify(response_data)

@app.route('/upload_marksheet', methods=['POST'])
def upload_marksheet_file():
    pdf_file = request.files.get('pdf_file')
    sl_no = session.get('sl_no')

    if pdf_file:
        status_message = common_module_2.upload_and_process(pdf_file,
common_module_2.process_marksheet, common_module_2.send_to_database_marksheet,
sl_no)
        if status_message == "Successfully Uploaded":
            session['marksheet_submitted'] = True
            error_message_marksheet = ""
        else:
            error_message_marksheet = f'Marksheet Error: {status_message}'
    else:
        # No file was uploaded, set an error message
        error_message_marksheet = "No Marksheet file uploaded"

    marksheet_submitted = not bool(error_message_marksheet)
    session['aadhar_submitted'] = session.get('aadhar_submitted', False)
    session['pan_submitted'] = session.get('pan_submitted', False)

    response_data = {
        'aadhar_submitted': session.get('aadhar_submitted', False),
        'marksheet_submitted': marksheet_submitted,
        'pan_submitted': session.get('pan_submitted', False),
        'error_message_aadhar': session.get('error_message_aadhar', ''),
        'error_message_pan': session.get('error_message_pan',''),
        'error_message_marksheet': error_message_marksheet,
        'sl_no': sl_no
    }

    return jsonify(response_data)


@app.route('/update_status', methods=['POST'])
def update_status():
    if session.get('aadhar_submitted') and session.get('pan_submitted') and
session.get('marksheet_submitted'):
        try:
```

```
            # Connect to the MySQL database
            conn = mysql.connector.connect(**db_config)
            cursor = conn.cursor()

            sl_no = session.get('sl_no')

            # Update the status of the user with the given sl_no to 'verified'
            query = "UPDATE email_sample_1 SET status = 'verified' WHERE sl_no = %s"
            cursor.execute(query, (sl_no,))

            # Commit the changes to the database
            conn.commit()

            # Close the database connection
            cursor.close()
            conn.close()

            # Reset the session flags
            session.pop('aadhar_submitted', None)
            session.pop('pan_submitted', None)
            session.pop('marksheet_submitted', None)

        except mysql.connector.Error as err:
            return f"Error: {err}"

    return redirect(url_for('admin_panel'))

if __name__ == '__main__':
    app.run(debug=True)
```

Front-End Codes:

Webpage codes –

1. add_user.html:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      background-image: url('{{ url_for('static', filename='bgp.jpg') }}');
      background-size: cover;
      background-repeat: no-repeat;
      background-color: #f5f5f5;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    .box {
      width: 90%;
      max-width: 700px;
      background-color: white;
      border: 1px solid black;
      border-radius: 10px;
      padding: 20px;
      display: flex;
      flex-direction: column;
      align-items: center;
    }

    .title {
      font-size: 24px;
      margin-bottom: 20px;
    }

    .form-container {
      display: flex;
      flex-direction: column;
      gap: 10px;
    }

    .form-container input[type="text"] {
      padding: 10px;
```

```
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        .form-container input[type="submit"] {
            background-color: #2d814b;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <div class="box">
        <h2 class="title">User Registration</h2>
        <form action="/add_user" method="post" class="form-container">
            <input type="text" name="name" placeholder="Name" required>
            <input type="text" name="email" placeholder="Email" required>
            <input type="submit" value="Add User">
        </form>
    </div>
</body>
</html>
```

2. admin_sample_1.html:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            background-image: url('{{ url_for('static', filename='bgp.jpg') }}');
            background-size: cover;
            background-repeat: no-repeat;
            background-color: #f5f5f5;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
```

```
.box2 {
    width: 90%;
    max-width: 700px;
    height: 310px;
    background-color: #163e5a;
    border: 1px solid black;
    border-radius: 10px;
    padding: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.box {
    width: 100%;
    background-color: white;
    border: 1px solid black;
    border-radius: 10px;
    padding: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.title {
    font-size: 24px;
    margin-bottom: 20px;
}

.button-container {
    margin-bottom: 20px;
    display: flex;
    justify-content: center;
    width: 100%;
}

.button-container input[type="submit"],
.add-user-button {
    background-color: #2d814b;
    color: white;
    padding: 10px 15px;
margin: 0 10px;
margin-left:20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
```

```
        table {
            width: 100%;
            border-collapse: collapse;
        }

        th, td {
            border: 1px solid #ddd;
            padding: 10px;
            text-align: left;
        }

        th {
            background-color: #163e5a; /* Light blue background color for column headings */
            color: white;
        }
    </style>
</head>
<body>
    <div class="box2">
        <div class="box">
            <h2 class="title">ADMIN PANEL</h2>
            <div class="button-container">
                <form action="/admin_panel" method="post">
                    <input type="submit" name="see_users" value="See Users">
                </form>
                <button class="add-user-button" onclick="location.href='{{
url_for('user_registration') }}'">Add User</button>
            </div>
            <table>
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Email</th>
                    <th>Action</th>
                    <th>Status</th>
                </tr>
                {% for user in users %}
                    <tr>
                        <td>{{ user.sl_no }}</td>
                        <td>{{ user.name }}</td>
                        <td>{{ user.email }}</td>
                        <td>
                            {% if user.status == "verified" %}
                                Action completed
                            {% else %}
                                <span id="url-container-{{ user.sl_no }}"></span>
```

```
                        <a href="#" class="generate-link" onclick="generateURL('{{
url_for('generate_url', sl_no=user.sl_no) }}', '{{ user.sl_no }}')">Generate URL</a>
                    {% endif %}
                </td>
                <td>{{ user.status }}</td>
            </tr>
        {% endfor %}
    </table>

</div>


</div>
<script>
    function generateURL(url, sl_no) {
        var urlContainer = document.getElementById('url-container-' + sl_no);
        var request = new XMLHttpRequest();

        request.onreadystatechange = function() {
            if (request.readyState === 4 && request.status === 200) {
                urlContainer.innerHTML = request.responseText;
            }
        };

        request.open('GET', url, true);
        request.send();
    }
</script>

</body>
</html>
```

3. login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/login_styles.css') }}">
</head>
<body>
    <div class="title">Validify<span>HUB</span></div>
    <div class="container">
```

```
        <h2>LOGIN</h2>
        <div class="form_container">
            <form method="POST" action="{{ url_for('login') }}">
                <input type="text" id="username" name="username" placeholder="Username"
required>
                <input type="password" id="password" name="password"
placeholder="Password" required>
                <input type="submit" value="Login">
            </form>
        </div>
        <p>Don't have an account? <a href="/register">Register</a></p>
    </div>
</body>
</html>
```

login_styles.css:

```
*

{
    margin: 0;
    padding: 0;
    font-family: 'Poppins', sans-serif;
    box-sizing: border-box;
}

body
{
    background-image: url('images/login_bg.png');
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center center;
    background-attachment: fixed;
    display: flex;
    justify-content: center;
    align-items: center;
}

.title {
    position: absolute;
    top: 15px;
    left: 20px;
    font-size: 3rem;
    font-weight: bold;
    color: #22092C;
}
span
```

```
{
    color: #872341;
}

h1
{
    color: #435585;
}

.container
{
    margin-top: 200px;
    background-color: rgba(255, 255, 255, 0.8);
    border-radius: 8px;
    padding: 20px;
    width: 400px;
    height: 350px;
    text-align: center;
}

h2
{
    color: #164863;
    font-size: 3rem;
}

.form_container
{
    margin-top: 40px;
}

form
{
    display: flex;
    flex-direction: column;
    align-items: center;
}

input
{
    width: 100%;
    padding: 10px;
    margin: 8px 0;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 4px;
}
```

```
input[type="submit"]
{
    background-color: #164863;
    color: #fff;
    cursor: pointer;
}

input[type="submit"]:hover
{
    background-color: #123456;
}

p
{
    margin-top: 10px;
    color: #555;
}

a
{
    color: #007bff;
    text-decoration: none;
}

a:hover
{
    text-decoration: underline;
}
```

4. register.html:

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/user_styles.css') }}">
</head>
<body>
    <div class="title">Validify<span>HUB</span></div>
    <div class="container">
        <h2>REGISTER</h2>
        <div class="form_container">
            <form method="POST" action="/register">
```

```
            <input type="text" name="username" placeholder="Username" required>
            <input type="password" name="password" placeholder="Password" required>
            <button type="submit">Register</button>
        </form>
        <p>Already have an account? <a href="/login">Login</a></p>
    </div>
  </div>
</body>
</html>
```

user_styles.css:

```
*
{
    margin: 0;
    padding: 0;
    font-family: 'Poppins', sans-serif;
    box-sizing: border-box;
}

body
{
    background-image: url('images/login_bg.png');
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center center;
    background-attachment: fixed;
    display: flex;
    justify-content: center;
    align-items: center;
}

.title {
    position: absolute;
    top: 15px;
    left: 20px;
    font-size: 3rem;
    font-weight: bold;
    color: #22092C;
}
span
{
    color: #872341;
}

h1
{
```

```
    color: #435585;
}

.container
{
    margin-top: 200px;
    background-color: rgba(255, 255, 255, 0.8);
    border-radius: 8px;
    padding: 20px;
    width: 400px;
    height: 350px;
    text-align: center;
}

h2
{
    color: #164863;
    font-size: 3rem;
}

.form_container
{
    margin-top: 40px;
}

form
{
    display: flex;
    flex-direction: column;
    align-items: center;
}

input
{
    width: 100%;
    padding: 10px;
    margin: 8px 0;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 4px;
}

input[type="submit"]
{
    background-color: #164863;
    color: #fff;
    cursor: pointer;
}
```

```
input[type="submit"]:hover
{
   background-color: #123456;
}
```

5.upload2_2.html:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <style>
      body {
         background-image: url('{{ url_for('static', filename='bgp.jpg') }}');
         background-size: cover;
         background-repeat: no-repeat;
         background-color: white;
         margin: 0;
         padding: 0;
         margin-top: 20px;
         display: flex;
         flex-direction: column;
         align-items: center;
         gap: 20px;
      }

      @media screen and (min-width: 768px) {
         .container {
            margin-top: 60px;
         }
      }

      .container {
         width: 80%;
         max-width: 600px;
         background-color: rgba(255, 255, 255, 0.8);
         border-radius: 10px;
         margin-top: 20px;
         padding: 20px;
         text-align: center;
      }

      .form-container {
         display: flex;
         flex-direction: column;
```

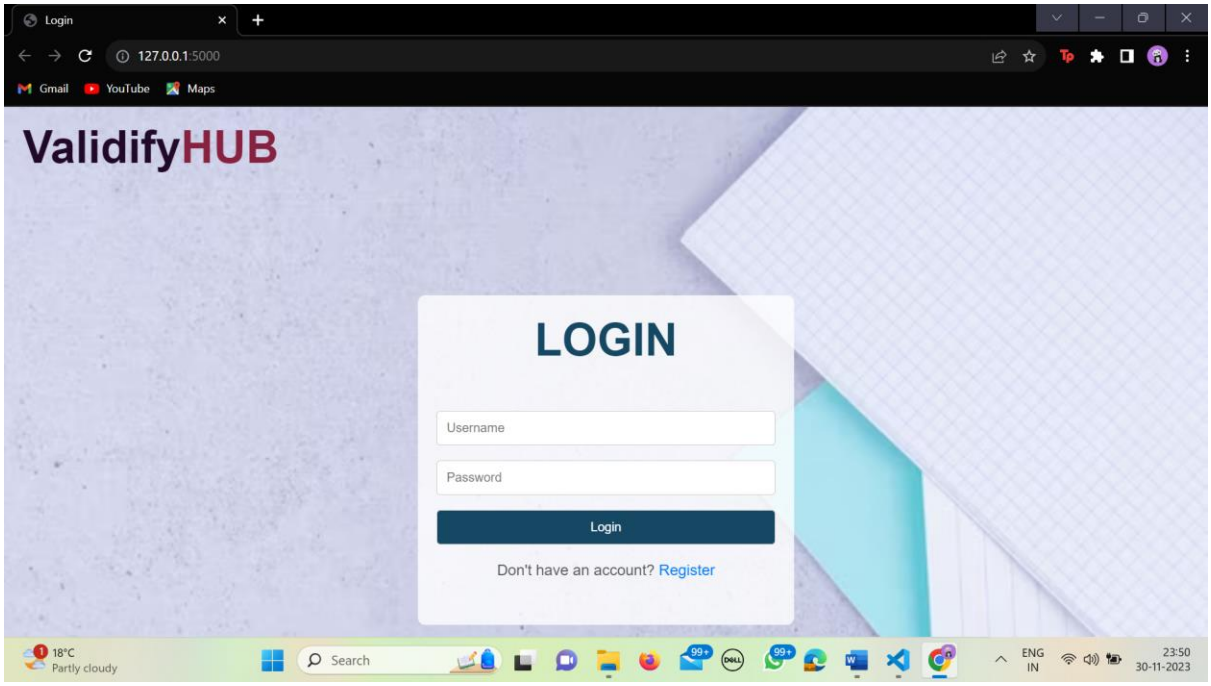
```
            gap: 10px;
        }

        .form-container input[type="file"] {
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        .form-container input[type="button"] {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        .status-message {
            color: green;
            font-weight: bold;
        }

        .error-message {
            color: red;
        }

        /* Loading spinner styles */
        .loading-spinner {
            display: none; /* Hidden by default */
            border: 4px solid rgba(255, 255, 255, 0.3);
            border-top: 4px solid #4CAF50;
            border-radius: 50%;
            width: 20px;
            height: 20px;
            animation: spin 2s linear infinite;
        }

        @keyframes spin {
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>User Page</h1>
        <p>User ID: {{ sl_no }}</p>
```

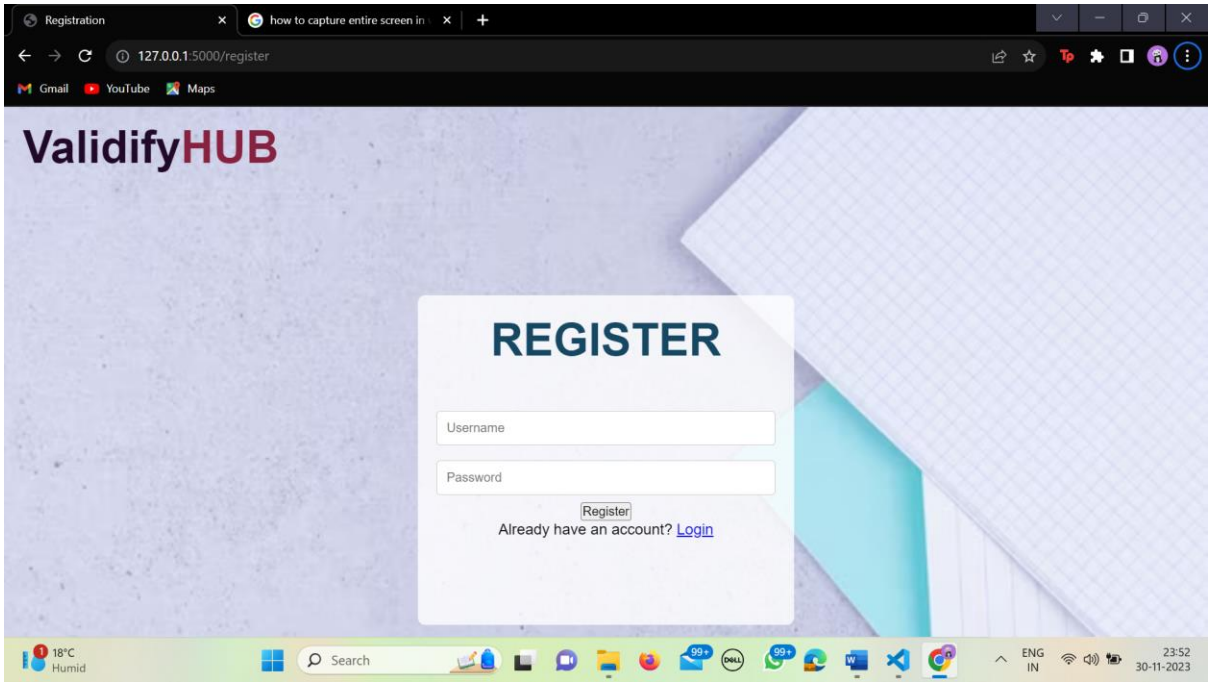
```
    <!-- Upload Aadhar Form -->
    <div class="form-container">
        <input type="file" id="aadhar-file" accept=".pdf">
        <input type="button" value="Upload Aadhar" onclick="uploadFile('aadhar')">
        <div id="aadhar-submitted-message" class="status-message"></div>
        <div id="aadhar-error-message" class="error-message"></div>
        <div id="aadhar-loading" class="loading-spinner"></div> <!-- Loading spinner -->
    </div>

    <!-- Upload PAN Form -->
    <div class="form-container">
        <input type="file" id="pan-file" accept=".pdf">
        <input type="button" value="Upload PAN" onclick="uploadFile('pan')">
        <div id="pan-submitted-message" class="status-message"></div>
        <div id="pan-error-message" class="error-message"></div>
        <div id="pan-loading" class="loading-spinner"></div> <!-- Loading spinner -->
    </div>

    <div class="form-container">
        <input type="file" id="marksheet-file" accept=".pdf">
        <input type="button" value="Upload Marksheet" onclick="uploadFile('marksheet')">
        <div id="marksheet-submitted-message" class="status-message"></div>
        <div id="marksheet-error-message" class="error-message"></div>
        <div id="marksheet-loading" class="loading-spinner"></div> <!-- Loading spinner --
>
    </div>

    <!-- Done Button -->
    <form id="done-form" action="/update_status" method="POST">
        <input type="submit" value="Done" id="done-button" disabled>
    </form>
</div>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
    function uploadFile(type) {
        var fileInput = document.getElementById(type + '-file');
        var file = fileInput.files[0];

        if (file) {
            var formData = new FormData();
            formData.append('pdf_file', file);

            // Show loading spinner
            $('#' + type + '-loading').show();

            $.ajax({
```

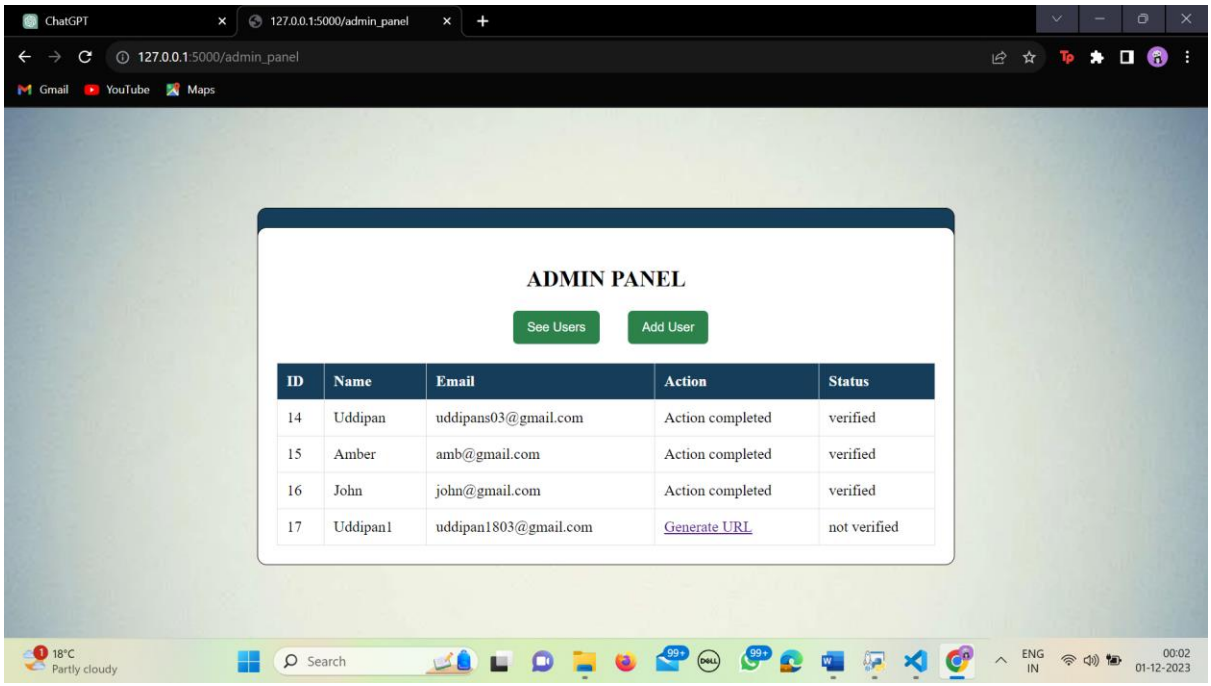
```
            type: 'POST',
            url: '/upload_' + type,
            data: formData,
            processData: false,
            contentType: false,
            beforeSend: function () {
                // Show loading spinner before the AJAX request is sent
            },
            success: function(response) {
                if (response['error_message_' + type]) {
                    $('#' + type + '-error-message').text(response['error_message_' + type]);
                    $('#' + type + '-submitted-message').text('');
                } else {
                    $('#' + type + '-submitted-message').text('Submitted');
                    $('#' + type + '-error-message').text('');
                }

                // Hide loading spinner on success
                $('#' + type + '-loading').hide();

                checkSubmissionStatus();
            },
            error: function(error) {
                $('#' + type + '-error-message').text('Error: ' + error.responseText);
                $('#' + type + '-submitted-message').text('');

                // Hide loading spinner on error
                $('#' + type + '-loading').hide();
            },
            complete: function() {
                // Hide loading spinner when the request is complete
            }
        });
    } else {
        $('#' + type + '-error-message').text('No file uploaded');
        $('#' + type + '-submitted-message').text('');
    }
}

function checkSubmissionStatus() {
    var aadharSubmitted = $('#aadhar-submitted-message').text() === 'Submitted';
    var panSubmitted = $('#pan-submitted-message').text() === 'Submitted';
    var marksheetSubmitted = $('#marksheet-submitted-message').text() === 'Submitted';
    var doneButton = $('#done-button');

    if (aadharSubmitted && panSubmitted && marksheetSubmitted) {
        doneButton.prop('disabled', false);
    } else {
```

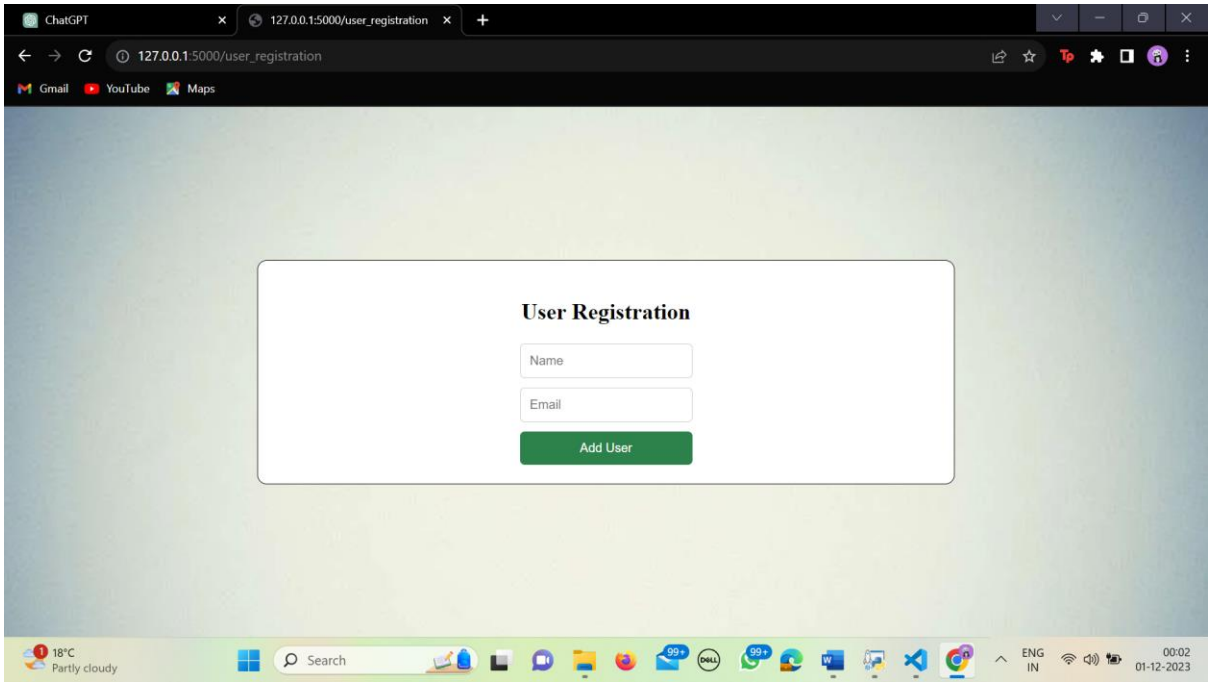
```
            doneButton.prop('disabled', true);
        }
    }

    $(document).ready(function() {
        checkSubmissionStatus();
    });
</script>
</body>
</html>
```

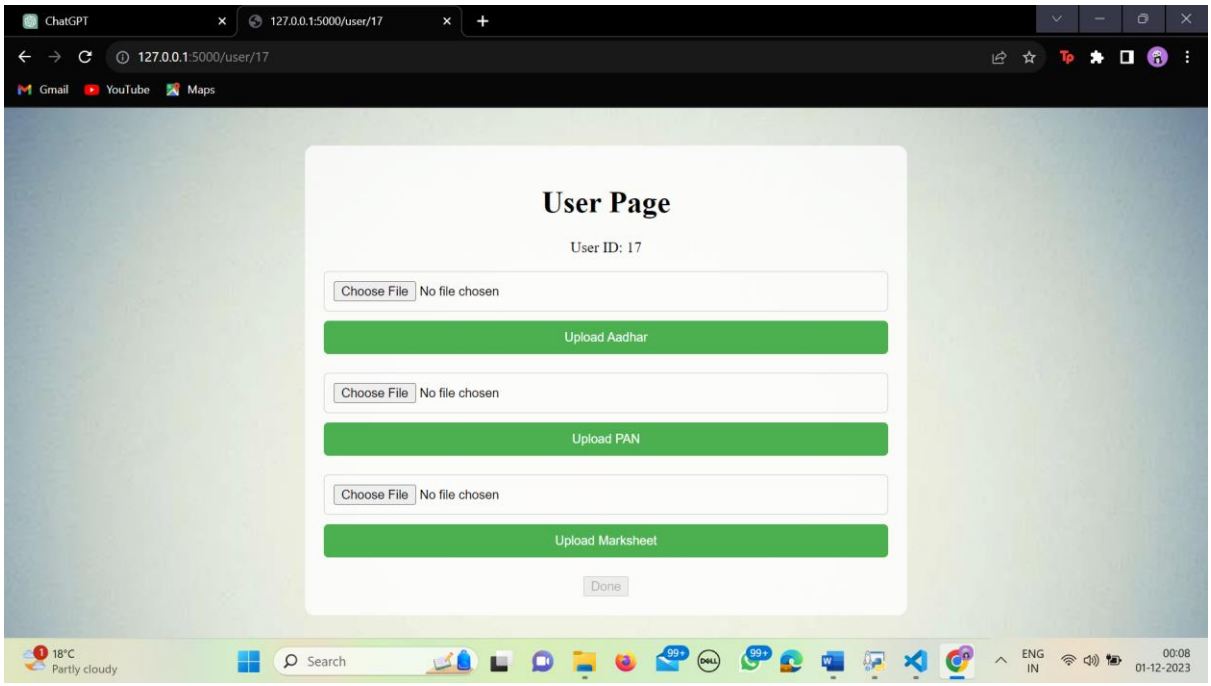
7. RESULTS AND CONCLUSION




Registration
how to capture entire screen in
127.0.0.1:5000/register
Gmail
YouTube
Maps
ValidifyHUB
REGISTER
Username
Password
Register
Already have an account? Login
18°C
Humid
Search
ENG
IN
23:52
30-11-2023


ChatGPT
127.0.0.1:5000/admin_panel
127.0.0.1:5000/admin_panel
Gmail
YouTube
Maps
ADMIN PANEL
See Users
Add User
ID
Name
Email
Action
Status
14
Uddipan
uddipans03@gmail.com
Action completed
verified
15
Amber
amb@gmail.com
Action completed
verified
16
John
john@gmail.com
Action completed
verified
17
Uddipan1
uddipan1803@gmail.com
Generate URL
not verified
18°C
Partly cloudy
Search
ENG
IN
00:02
01-12-2023


ChatGPT
127.0.0.1:5000/user_registration
127.0.0.1:5000/user_registration
Gmail
YouTube
Maps
User Registration
Name
Email
Add User
18°C
Partly cloudy
Search
ENG
IN
00:02
01-12-2023


ChatGPT
127.0.0.1:5000/user/17
127.0.0.1:5000/user/17
Gmail
YouTube
Maps
User Page
User ID: 17
Choose File
No file chosen
Upload Aadhar
Choose File
No file chosen
Upload PAN
Choose File
No file chosen
Upload Marksheet
Done
18°C
Partly cloudy
Search
ENG
IN
00:08
01-12-2023

This web application has been developed to verify and validate the document being uploaded by the users, to reduce occurrences of invalid data. It is designed to replace the existing verification systems where only the file type is being verified, not the data inside the file. This will also lead to more automation as the administrator will not have to physically verify the authenticity of the files. This also improves the time efficiency of the application.

In the modern world, where each and every task is now being done in the most optimized method possible, this is but a small step in increasing efficiency and reducing physical labor, hence increasing productivity and contributing to development.