SMART-ID

# SMART ID SCANNER

**Developed and designed by:**

**Tapas Sahu**

**GitHub:**

https://github.com/Tapas14102000

**GitHub project link:**

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner

# 1. INTRODUCTION

Came across a banner or any picture with text or a visiting card? Would you like to store the information in that image as text for future use? It is possible by extracting the text from the image. This application helps you to do that and stores it in a very secured environment and manages the data with maximum efficiency.

# 2. LITERATURE SURVEY

## 2.1 Existing Problem

In this modern era, everyone is so much busy that they don't want to put efforts on writing down important facts, information that they get from surroundings like mostly many students nowadays due to pandemic are attending classes on online platforms, they generally take screenshots of the lectures so that they can use those for their future references. This screenshots not only consumes much larger spaces in your system but rather it slows down the performance of your system. Larger the size of files, lesser will be performance.

## 2.2 Purposed solution

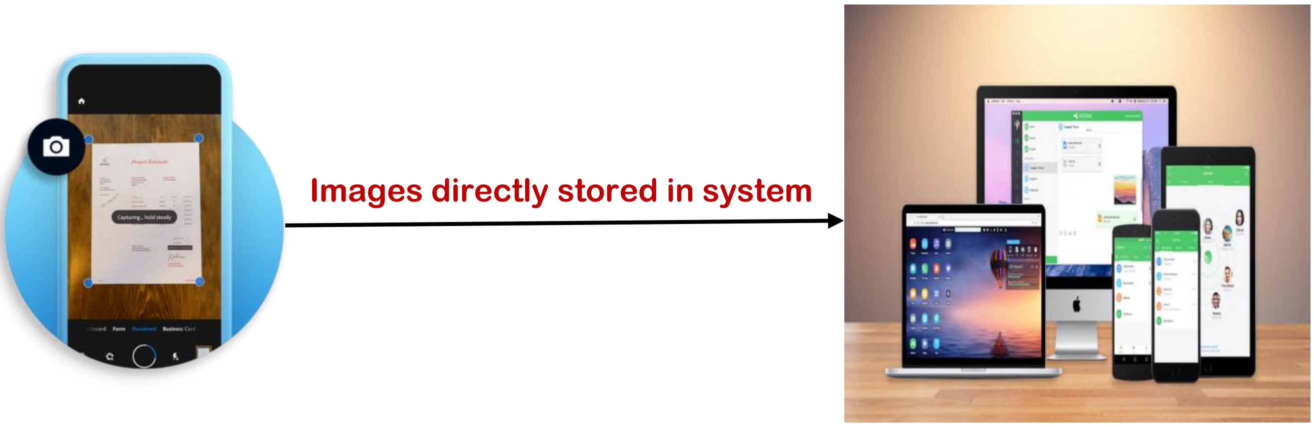
Just imagine if there was some application which can just extracts the text from images and stores them locally for your future use.

No more need of storing those images just upload the image in our application and we will provide you the text by extracting it from images.Which means, Space consumption minimized and your system is free from all these waste memory leakage.

# 3. THEORITICAL ANALYSIS

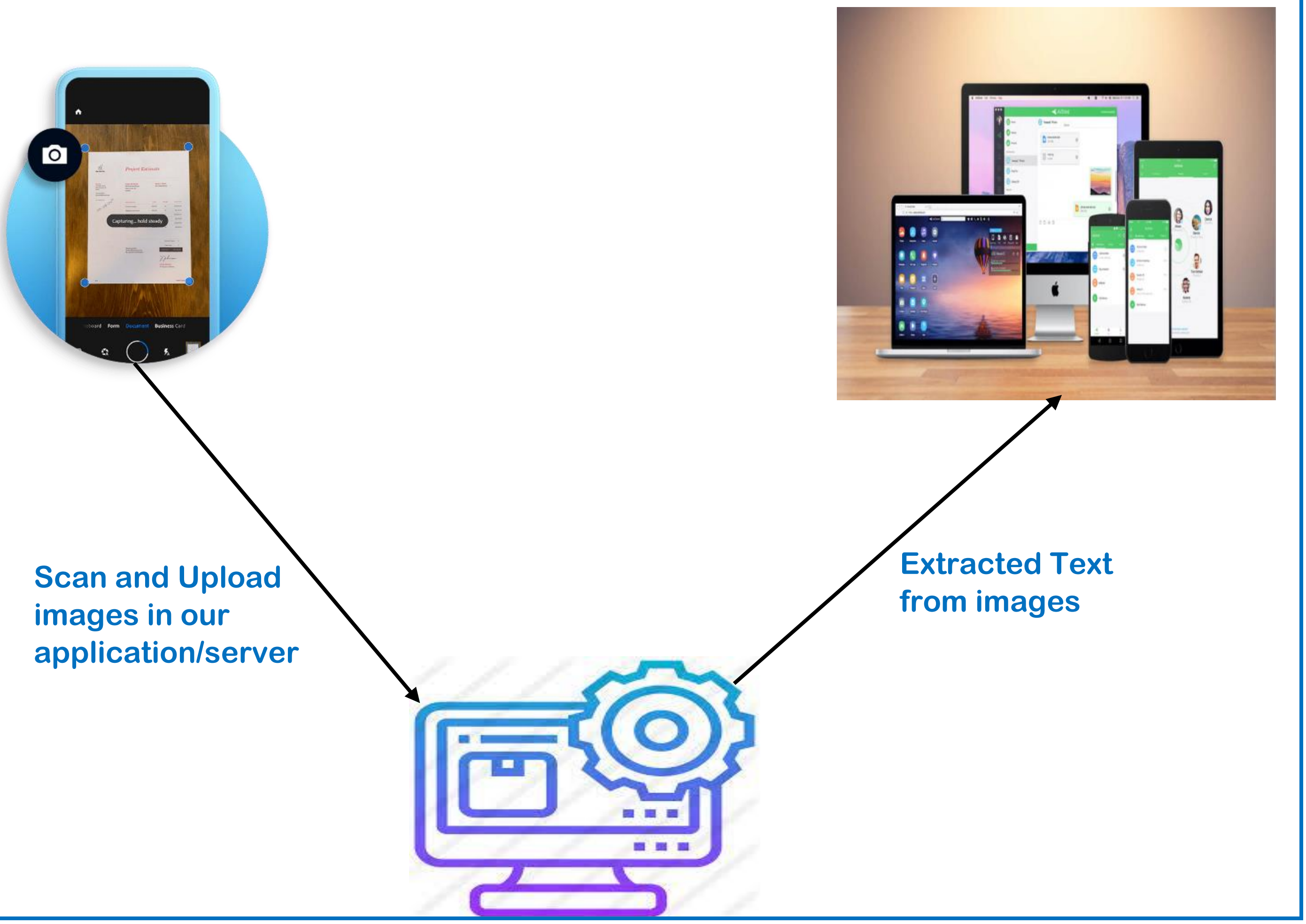
## 3.1 Block Diagram

### Traditional way:



Images directly stored in system

### Our Solution:



Scan and Upload images in our application/server

Extracted Text from images

## 3.2 Hardware/Software

### Server end is coded in:

- **Springboot**
- **Spring Security**
- **Hibernate**
- **Spring data JPA**
- **Remote MySQL**
- **Spring Mail**
- **Docker**
- **Tesseract OCR**
- **Thymeleaf**
- **Bootstrap, HTML, CSS, JS**

### Client end requires:

- **Browser with JS enabled**

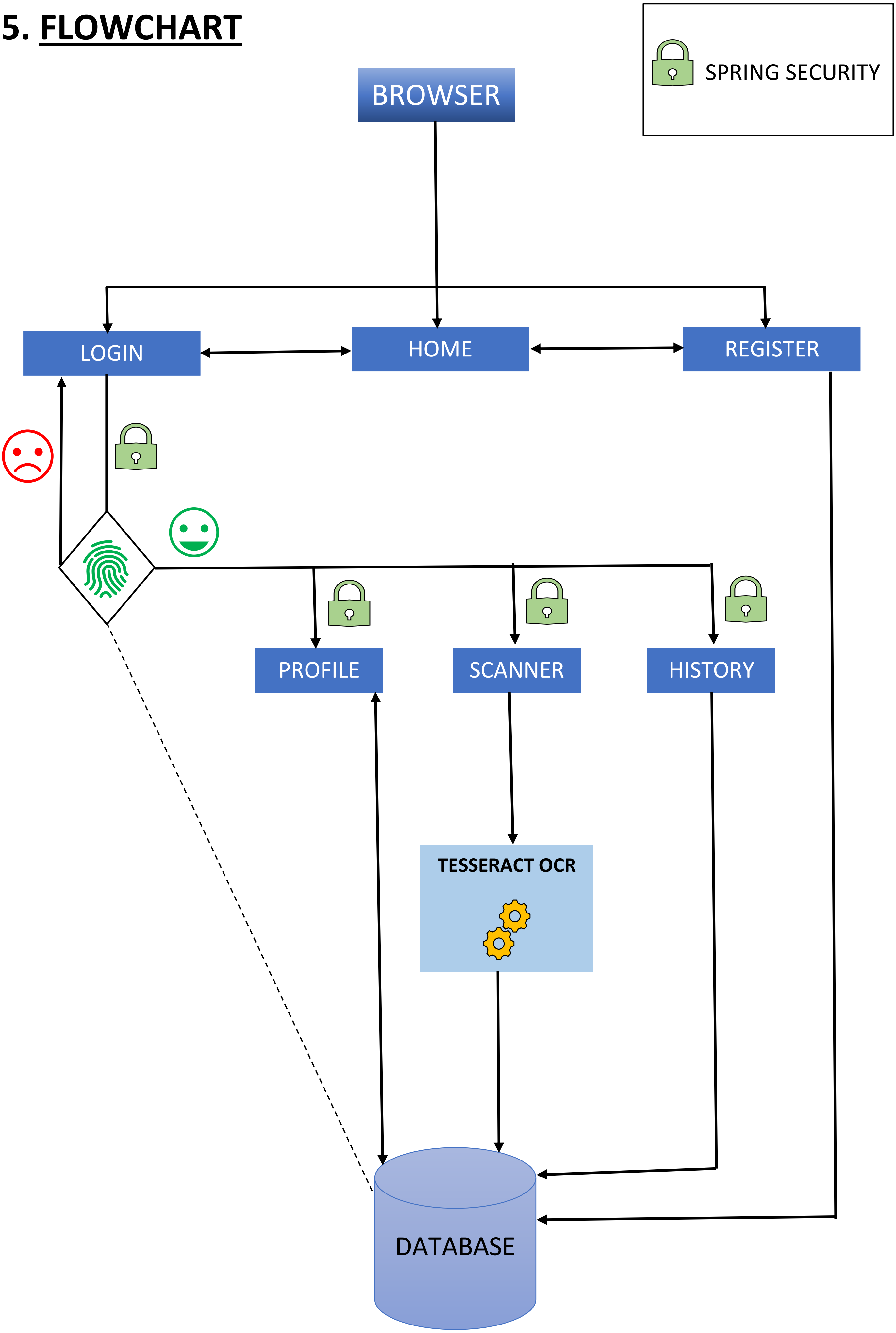
# 4. EXPERIMENTAL INVESTIGATIONS

The application is working absolutely fine and it is also optimized with respect to both time and space.

The results are having approx 98% accuracy if provided high pixeled clear images and the result quality decreases if images are not that clear as expected.

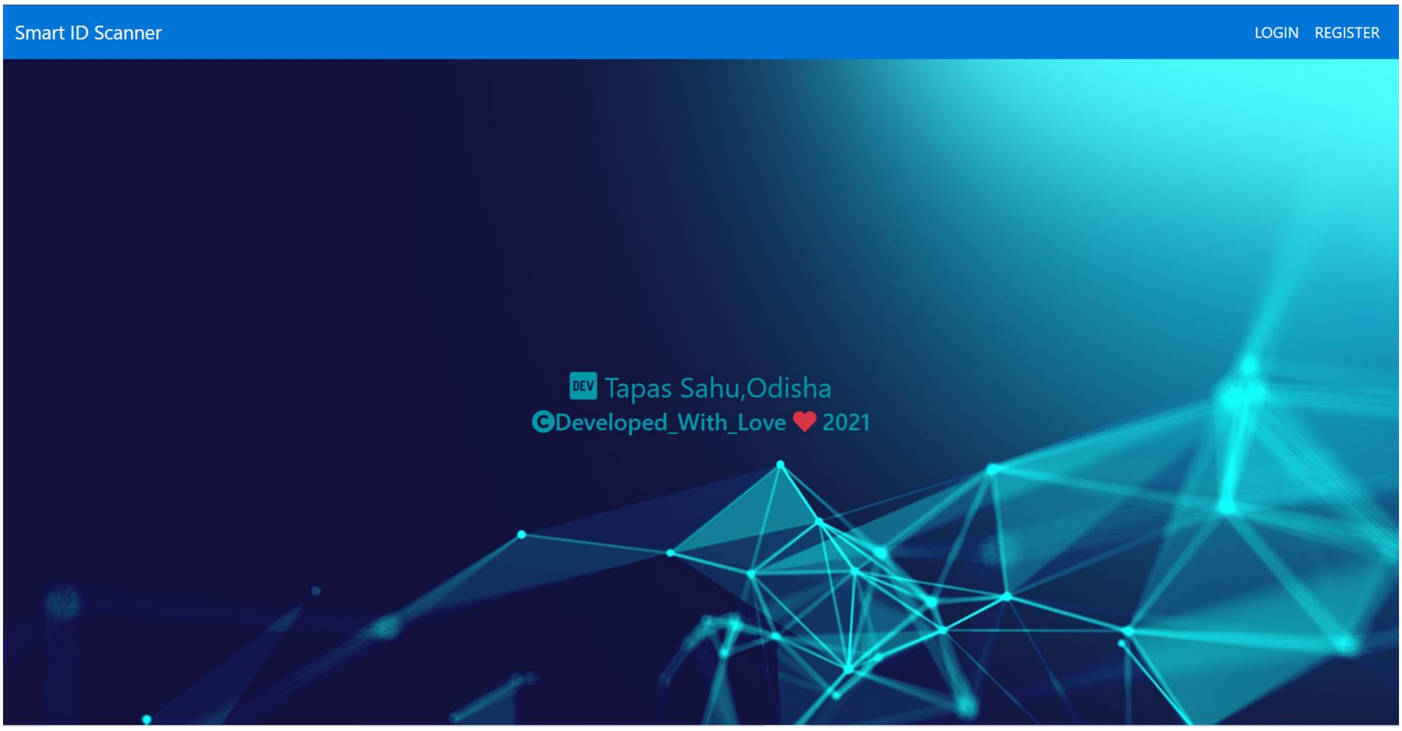
Store the text if and only if you are satisfied with the extracted result.

# 5. FLOWCHART


SPRING SECURITY
BROWSER
LOGIN
HOME
REGISTER
PROFILE
SCANNER
HISTORY
TESSERACT OCR
DATABASE

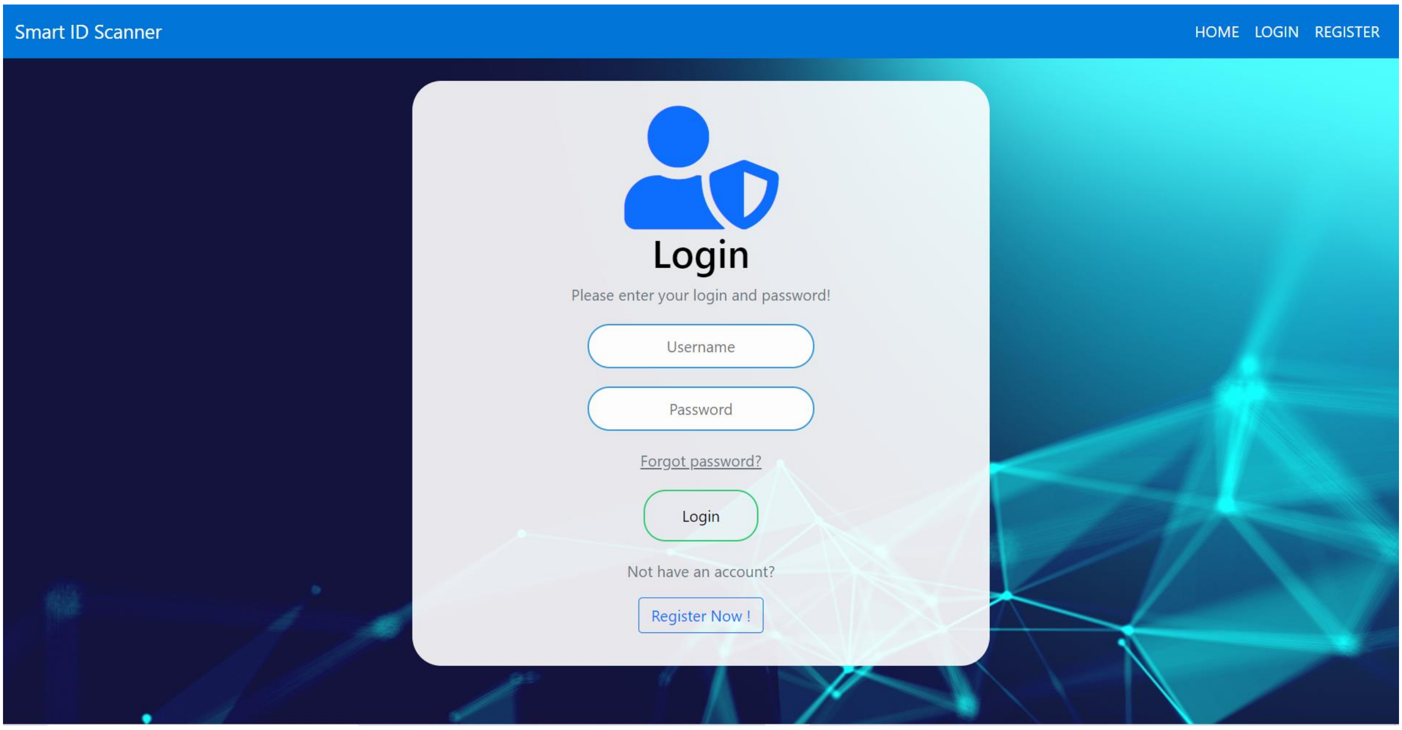
# 6. RESULT

## HOME PAGE



DEV Tapas Sahu,Odisha
©Developed_With_Love ❤ 2021

## LOGIN PAGE

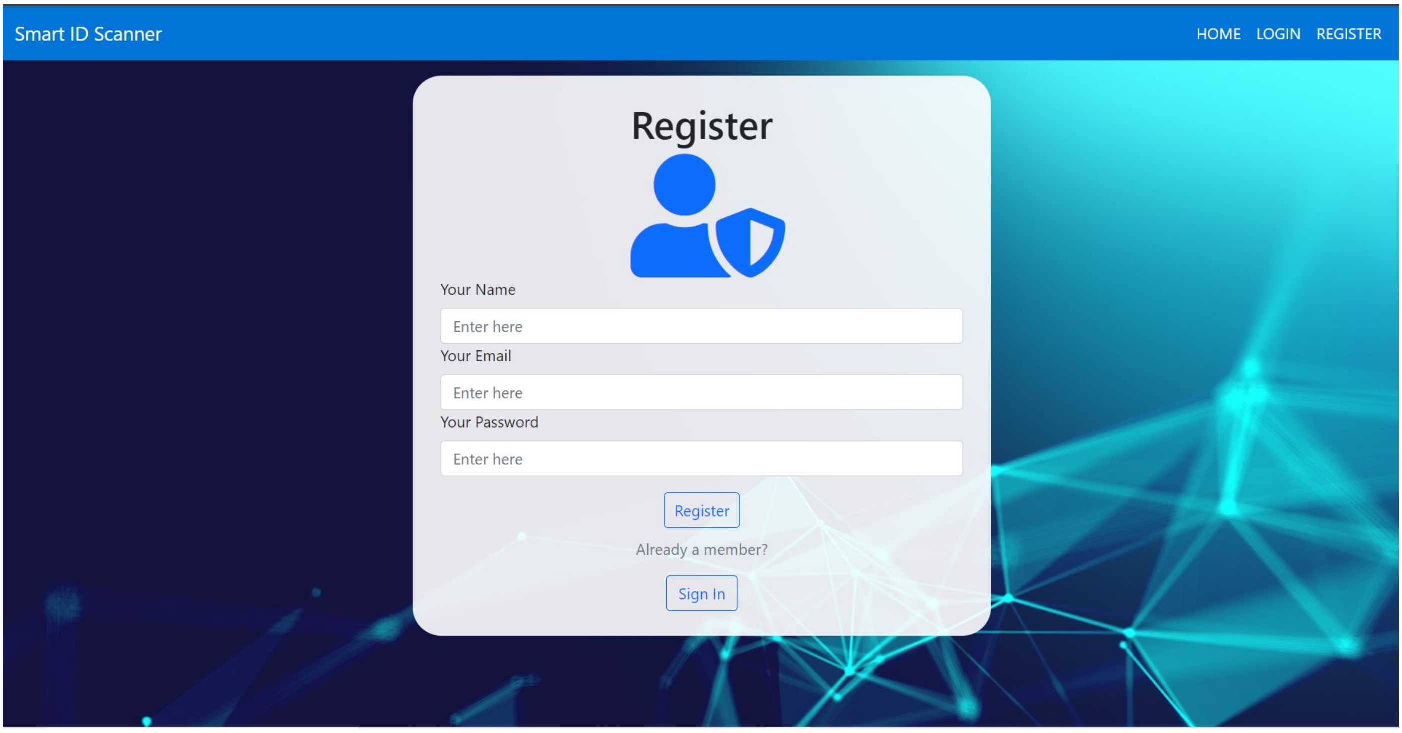


Login

Please enter your login and password!

Not have an account?

# REGISTER PAGE



Register

Your Name

Enter here

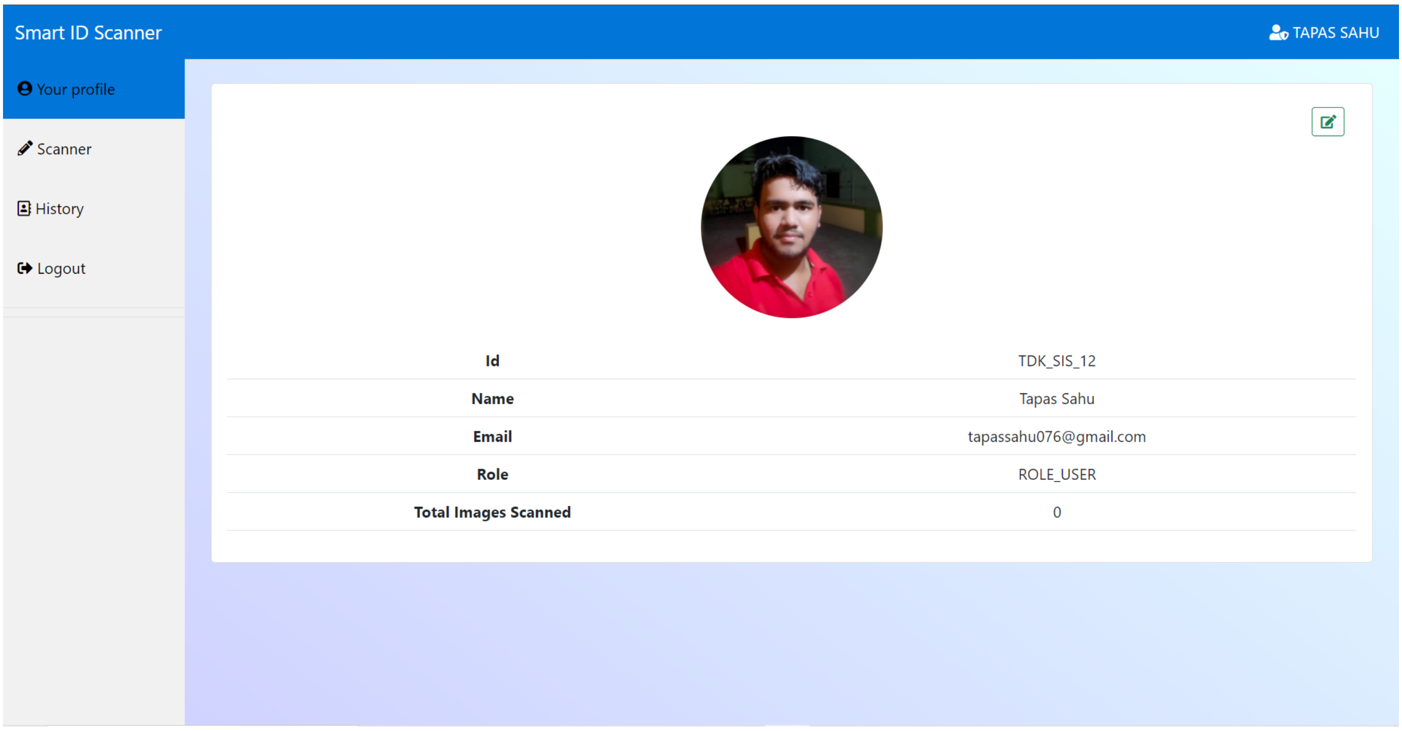
Your Email

Enter here

Your Password

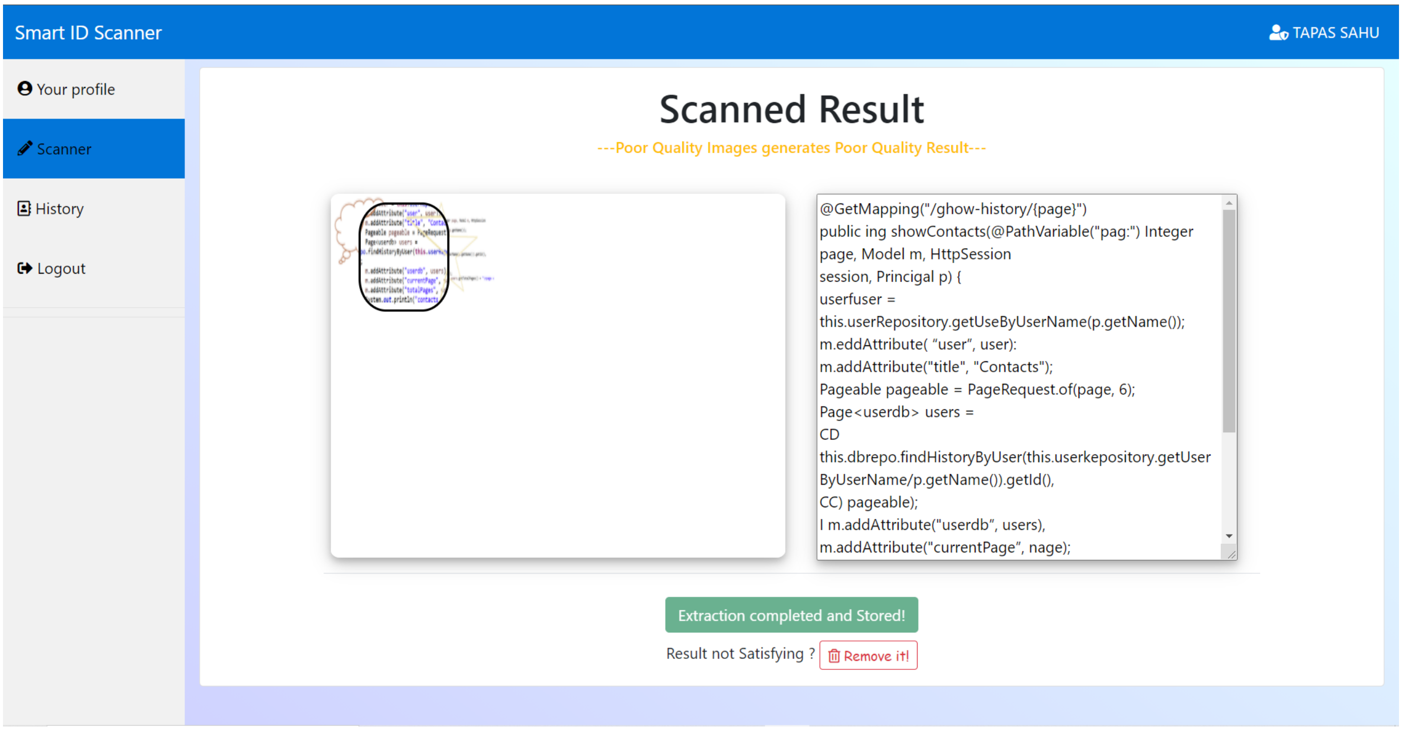
Enter here

# PROFILE PAGE



| | |
|---|---|
| **Id** | TDK_SIS_12 |
| **Name** | Tapas Sahu |
| **Email** | tapassahu076@gmail.com |
| **Role** | ROLE_USER |
| **Total Images Scanned** | 0 |

# SCANNER



## Scanned Result

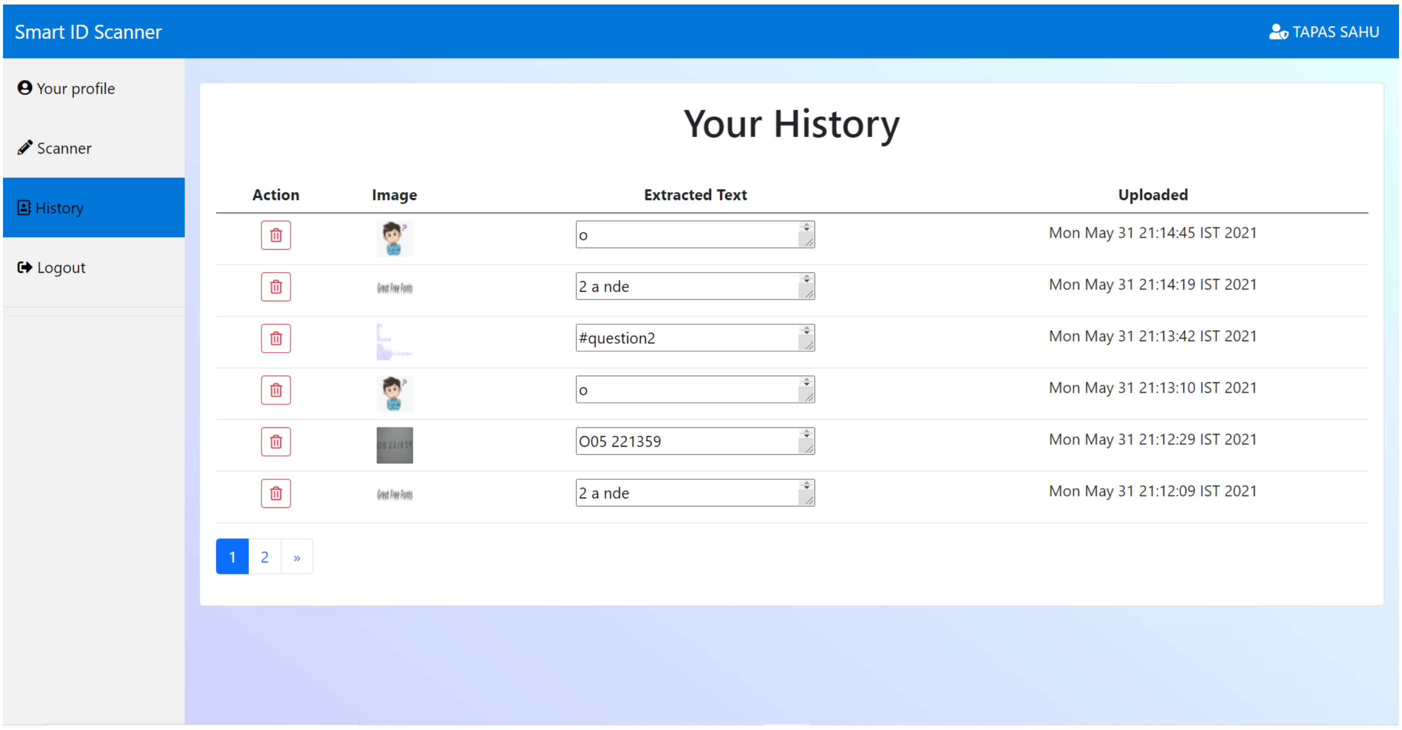
---Poor Quality Images generates Poor Quality Result---

```
@GetMapping("/ghow-history/{page}")
public ing showContacts(@PathVariable("pag:") Integer
page, Model m, HttpSession
session, Principal p) {
userfuser =
this.userRepository.getUseByUserName(p.getName());
m.eddAttribute( "user", user):
m.addAttribute("title", "Contacts");
Pageable pageable = PageRequest.of(page, 6);
Page<userdb> users =
CD
this.dbrepo.findHistoryByUser(this.userkepository.getUser
ByUserName/p.getName()).getId(),
CC) pageable);
I m.addAttribute("userdb", users),
m.addAttribute("currentPage", nage);
```

Extraction completed and Stored!

Result not Satisfying ? Remove it!

# HISTORY



## Your History

| Action | Image | Extracted Text | Uploaded |
|---|---|---|---|
| 🗑 | | o | Mon May 31 21:14:45 IST 2021 |
| 🗑 | | 2 a nde | Mon May 31 21:14:19 IST 2021 |
| 🗑 | | #question2 | Mon May 31 21:13:42 IST 2021 |
| 🗑 | | o | Mon May 31 21:13:10 IST 2021 |
| 🗑 | | 005 221359 | Mon May 31 21:12:29 IST 2021 |
| 🗑 | | 2 a nde | Mon May 31 21:12:09 IST 2021 |

## 7. ADVANTAGES AND DISADVANTAGES

**Advantages:**

1. Easily extract text from images and store it for future reference.
2. Your account is highly encrypted by spring security.
3. Better the pixel/clarity of the image, Better will be the result.
4. User Friendly UI design to give our user best experience.
5. Manage your data based upon your requirement.

**Disadvantages:**

1. Complex the image more time it takes to fetch text.
2. The application only predicts English alphabets (can be diversified in future upon user's requirement).
3. The accuracy of extraction is not that good (Tesseract OCR need to be updated) i.e., accuracy varies from 30-98%.

## 8. APPLICATIONS

**This application can be used in following areas:**

1. **Medical domains**

   Patients can upload their data by writing over a paper and the hospital staff can get the patients details from there with no use of pen and paper or manual inserting data in database.

   Results in no standing in a long queue for hours to get a ticket at hospital.

   (Most required in these pandemic times)

2. **Educational domains**

   Students can extract text from lecture notes to keep it for future reference.

   Education department can keep track of students details directly from the above-mentioned ways. (no-contact service)

   (Most required in these pandemic times)

3. **Military domains**

   **Military spy can keep important information from a piece of paper quicly when on a mission at enemy's country.**

4. **Shopping malls**

   **Elders generally like to keep notes of goods they will buy before going to market.**

   **Shopping mall staff can use our application to extract the required goods that their customer want by just uploading their item list.**

   **(Most required in these pandemic times)**

5. **Companies and different Organizations can use this kind of technology to keep track of its employee working hours . Employee will be given a smart card with a unique QR-Code that will be scanned by our application every time he/she come or leave the company at the end of the day.**

   **And in many more such areas we can utilize this technology to get maximum output.**

# 9. CONCLUSION

This technology will have a strong impact in this technologically changing world.

Every second millions of users are interacting with images that they only store for the information that is present over it. This not only leads to the wastage of memory but as well slows down their systems performance.

This technology can change the way that today's generation is seeing. Want information present in a text no need to store photos, just extract the text and store that.

But still this technology is needed to be updated more. As its accuracy is not that much acceptable.

Accuracy rate varies between 30-98% and is a bit slower.

# 10. BIBLIOGRAPHY

- https://docs.spring.io/spring-framework/docs/5.0.0.M1/spring-framework-reference/pdf/spring-framework-reference.pdf
- https://github.com/tesseract-ocr/
- https://docs.docker.com/
- https://www.youtube.com/watch?v=fufHSlovTE0
- https://www.youtube.com/watch?v=EhG5zuV7SUM
- https://www.youtube.com/watch?v=oK8FqPja1Ng
- https://www.youtube.com/watch?v=yhzBpRZSe0k&t=9869s

## Source code:

Controller –

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/tree/main/src/main/java/com/TapasCodes/controller

Configuration –

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/tree/main/src/main/java/com/TapasCodes/config

DAO –

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/tree/main/src/main/java/com/TapasCodes/dao

Entities –

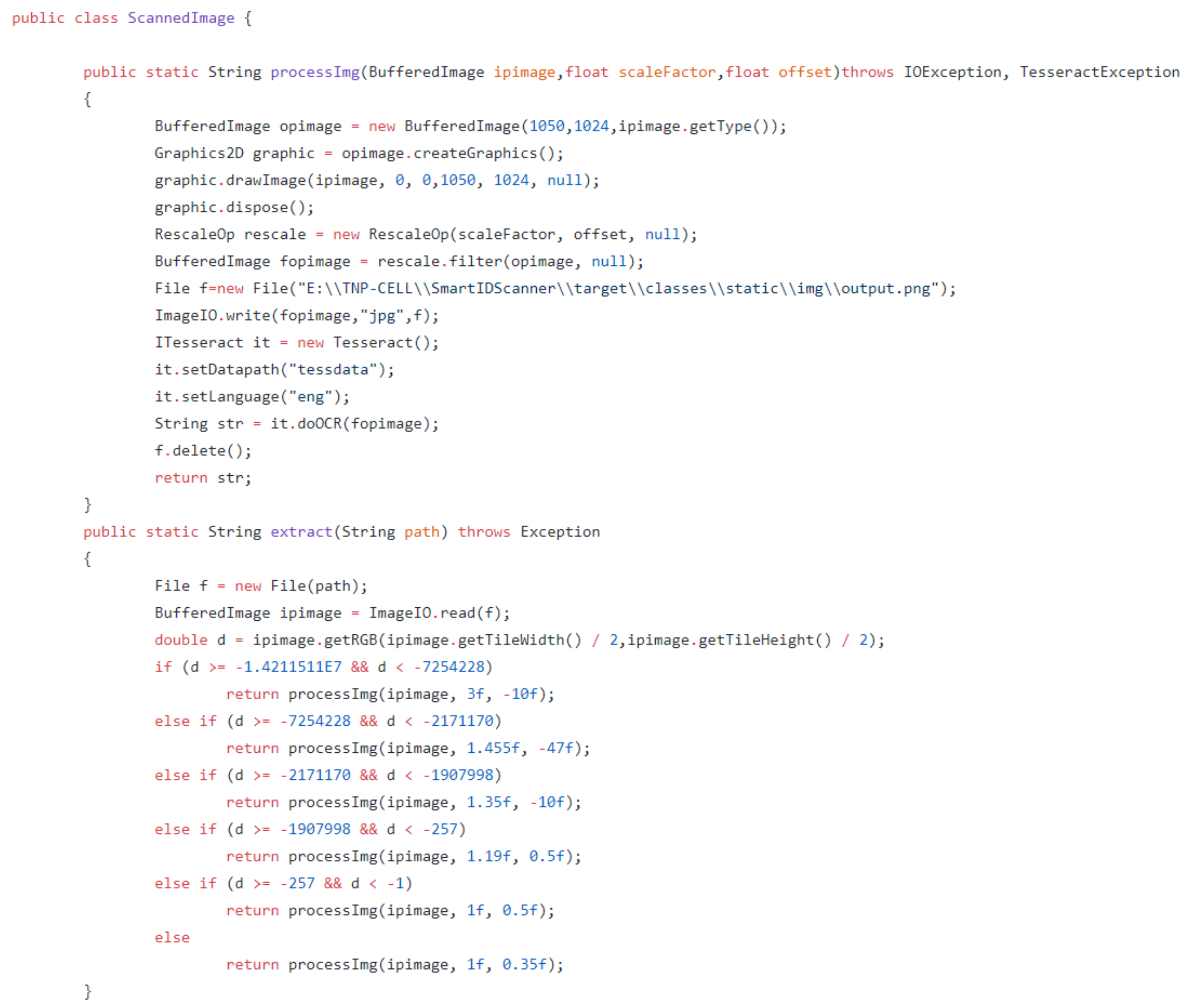
https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/tree/main/src/main/java/com/TapasCodes/entities

Helper –

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/tree/main/src/main/java/com/TapasCodes/helper

Scanned Image –

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/blob/main/src/main/java/com/TapasCodes/ScannedImage.java



```
public class ScannedImage {

        public static String processImg(BufferedImage ipimage,float scaleFactor,float offset)throws IOException, TesseractException
        {
                BufferedImage opimage = new BufferedImage(1050,1024,ipimage.getType());
                Graphics2D graphic = opimage.createGraphics();
                graphic.drawImage(ipimage, 0, 0,1050, 1024, null);
                graphic.dispose();
                RescaleOp rescale = new RescaleOp(scaleFactor, offset, null);
                BufferedImage fopimage = rescale.filter(opimage, null);
                File f=new File("E:\\TNP-CELL\\SmartIDScanner\\target\\classes\\static\\img\\output.png");
                ImageIO.write(fopimage,"jpg",f);
                ITesseract it = new Tesseract();
                it.setDatapath("tessdata");
                it.setLanguage("eng");
                String str = it.doOCR(fopimage);
                f.delete();
                return str;
        }
        public static String extract(String path) throws Exception
        {
                File f = new File(path);
                BufferedImage ipimage = ImageIO.read(f);
                double d = ipimage.getRGB(ipimage.getTileWidth() / 2,ipimage.getTileHeight() / 2);
                if (d >= -1.4211511E7 && d < -7254228)
                        return processImg(ipimage, 3f, -10f);
                else if (d >= -7254228 && d < -2171170)
                        return processImg(ipimage, 1.455f, -47f);
                else if (d >= -2171170 && d < -1907998)
                        return processImg(ipimage, 1.35f, -10f);
                else if (d >= -1907998 && d < -257)
                        return processImg(ipimage, 1.19f, 0.5f);
                else if (d >= -257 && d < -1)
                        return processImg(ipimage, 1f, 0.5f);
                else
                        return processImg(ipimage, 1f, 0.35f);
        }
```

Properties file –

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/blob/main/src/main/resources/application.properties

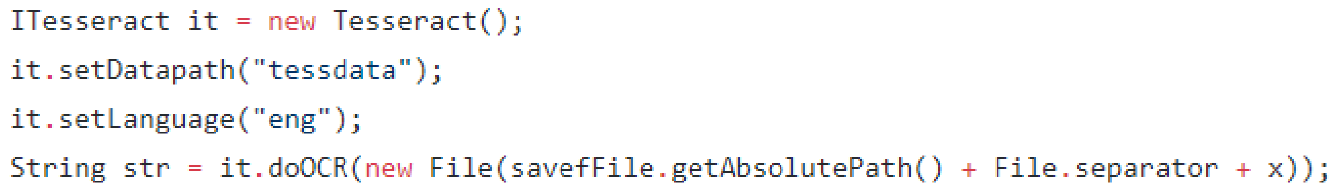
Templates –

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/tree/main/src/main/resources/templates

Static –

https://github.com/smartinternz02/SPS-10534-Smart-ID-Scanner/tree/main/src/main/resources/static

Tesseract OCR Code –



```
ITesseract it = new Tesseract();
it.setDatapath("tessdata");
it.setLanguage("eng");
String str = it.doOCR(new File(savefFile.getAbsolutePath() + File.separator + x));
```