**Add Excel data to db:-**

**STEP1**.<!-- For Excel file parsing -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>5.2.5</version>

</dependency>

**STEP** 2:-

@Service

public class EmployeeService {

private final EmployeeRepository employeeRepository;

public EmployeeService(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

public void saveEmployeesFromExcel(MultipartFile file) {

try {

InputStream inputStream = file.getInputStream();

Workbook workbook = WorkbookFactory.create(inputStream);

Sheet sheet = workbook.getSheetAt(0);

List<Employee> employees = new ArrayList<>();

for (int i = 1; i <= sheet.getLastRowNum(); i++) { // skip header row

Row row = sheet.getRow(i);

if (row == null) continue;

Employee emp = new Employee();

emp.setName(row.getCell(0).getStringCellValue());

emp.setEmail(row.getCell(1).getStringCellValue());

emp.setDepartment(row.getCell(2).getStringCellValue());

employees.add(emp);

}

employeeRepository.saveAll(employees);

workbook.close();

} catch (Exception e) {

throw new RuntimeException("Failed to parse Excel file: " + e.getMessage());

}

}

}

**Add pdf data to db:-**

suppose this is pdf data :-

Name,Email,Department

John Doe,john@example.com,IT

Jane Roe,jane@example.com,HR

**Step 1:-** <!-- Apache PDFBox -->

<dependency>

<groupId>org.apache.pdfbox</groupId>

<artifactId>pdfbox</artifactId>

<version>2.0.29</version>

</dependency>

**Step 2:-**

@Service

public class PDFService {

private final EmployeeRepository employeeRepository;

public PDFService(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

public void extractAndSaveEmployees(MultipartFile file) throws IOException {

PDDocument document = PDDocument.load(file.getInputStream());

PDFTextStripper pdfStripper = new PDFTextStripper();

// Extract text

String text = pdfStripper.getText(document);

document.close();

// Use StringBuilder (optional)

StringBuilder sb = new StringBuilder();

sb.append(text);

// Split lines

String[] lines = sb.toString().split("\\r?\\n");

List<Employee> employees = new ArrayList<>();

for (String line : lines) {

// Skip empty lines

if (line.trim().isEmpty()) continue;

// Parse CSV-style line: Name,Email,Department

String[] data = line.split(",");

if (data.length != 3) continue;

Employee emp = new Employee();

emp.setName(data[0].trim());

emp.setEmail(data[1].trim());

emp.setDepartment(data[2].trim());

employees.add(emp);

}

employeeRepository.saveAll(employees);

}

}

**generate an Excel file**

**step 1 : -**

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>5.2.5</version>

</dependency>

**step 2: -**

@GetMapping("/download-excel")

public ResponseEntity<byte[]> downloadExcel() {

List<Employee> employees = employeeRepository.findAll();

byte[] excelData = excelService.generateEmployeeExcel(employees);

return ResponseEntity.ok()

.header(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=employees.xlsx")

.contentType(MediaType.APPLICATION\_OCTET\_STREAM)

.body(excelData);

}

**step 3:-**

@Service

public class ExcelService {

public byte[] generateEmployeeExcel(List<Employee> employees) {

try (Workbook workbook = new XSSFWorkbook()) {

Sheet sheet = workbook.createSheet("Employees");

// Header row

Row header = sheet.createRow(0);

header.createCell(0).setCellValue("ID");

header.createCell(1).setCellValue("Name");

header.createCell(2).setCellValue("Email");

header.createCell(3).setCellValue("Department");

// Data rows

int rowNum = 1;

for (Employee emp : employees) {

Row row = sheet.createRow(rowNum++);

row.createCell(0).setCellValue(emp.getId());

row.createCell(1).setCellValue(emp.getName());

row.createCell(2).setCellValue(emp.getEmail());

row.createCell(3).setCellValue(emp.getDepartment());

}

// Autosize columns

for (int i = 0; i < 4; i++) {

sheet.autoSizeColumn(i);

}

// Write to byte array

ByteArrayOutputStream bos = new ByteArrayOutputStream();

workbook.write(bos);

return bos.toByteArray();

} catch (Exception e) {

throw new RuntimeException("Failed to generate Excel: " + e.getMessage());

}

}

}

**generate a PDF**

**step 1:-**

<dependency>

<groupId>com.itextpdf</groupId>

<artifactId>itext7-core</artifactId>

<version>7.2.5</version>

</dependency>

@GetMapping("/download-pdf")

public ResponseEntity<byte[]> downloadPDF() {

List<Employee> employees = employeeRepository.findAll();

byte[] pdfData = pdfGenerateService.generateEmployeePDF(employees);

return ResponseEntity.ok()

.header(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=employees.pdf")

.contentType(MediaType.APPLICATION\_PDF)

.body(pdfData);

}

**step3:-**

@Service

public class PDFGenerateService {

public byte[] generateEmployeePDF(List<Employee> employees) {

try {

Document document = new Document();

ByteArrayOutputStream baos = new ByteArrayOutputStream();

PdfWriter.getInstance(document, baos);

document.open();

document.add(new Paragraph("Employee List"));

document.add(new Paragraph(" ")); // empty line

for (Employee emp : employees) {

String line = emp.getId() + " | " + emp.getName() + " | " + emp.getEmail() + " | " + emp.getDepartment();

document.add(new Paragraph(line));

}

document.close();

return baos.toByteArray();

} catch (Exception e) {

throw new RuntimeException("Failed to generate PDF: " + e.getMessage());

}

}

}

**read data/text from an image**

**Note: You also need Tesseract OCR installed on your system or use the tessdata language files.**

**step 1:-**

<dependency>

<groupId>net.sourceforge.tess4j</groupId>

<artifactId>tess4j</artifactId>

<version>5.7.0</version>

</dependency>

**step 2:-**

@Service

public class ImageOCRService {

private final EmployeeRepository employeeRepository;

public ImageOCRService(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

public String extractAndSaveEmployees(MultipartFile file) {

try {

// Convert MultipartFile to File

File convFile = File.createTempFile("image", ".tmp");

try (FileOutputStream fos = new FileOutputStream(convFile)) {

fos.write(file.getBytes());

}

// Initialize Tesseract

ITesseract tesseract = new Tesseract();

tesseract.setDatapath("C:\\Program Files (x86)\\Tesseract-OCR\\tessdata");

tesseract.setLanguage("eng");

// Extract text

String text = tesseract.doOCR(convFile);

// Cleanup temp file

convFile.delete();

// Parse text and save to DB

List<Employee> employees = new ArrayList<>();

String[] lines = text.split("\\r?\\n");

for (String line : lines) {

if (line.trim().isEmpty()) continue;

String[] data = line.split(",");

if (data.length != 3) continue;

Employee emp = new Employee();

emp.setName(data[0].trim());

emp.setEmail(data[1].trim());

emp.setDepartment(data[2].trim());

employees.add(emp);

}

employeeRepository.saveAll(employees);

return text;

} catch (Exception e) {

throw new RuntimeException("Failed to process image: " + e.getMessage());

}

}

}

**generate an image from data**

**step 1:-**

@GetMapping("/download-image")

public ResponseEntity<byte[]> downloadImage() {

List<Employee> employees = employeeRepository.findAll();

byte[] imageData = imageGenerateService.generateEmployeeImage(employees);

return ResponseEntity.ok()

.header(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=employees.png")

.contentType(MediaType.IMAGE\_PNG)

.body(imageData);

}

}

**step 2:-**

import org.springframework.stereotype.Service;

import javax.imageio.ImageIO;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;

import java.util.List;

@Service

public class ImageGenerateService {

public byte[] generateEmployeeImage(List<Employee> employees) {

try {

int width = 600;

int height = 50 + employees.size() \* 30; // height based on number of employees

// Create a BufferedImage

BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

Graphics2D g = image.createGraphics();

// Background color

g.setColor(Color.WHITE);

g.fillRect(0, 0, width, height);

// Text color & font

g.setColor(Color.BLACK);

g.setFont(new Font("Arial", Font.BOLD, 16));

int y = 30;

g.drawString("Employee List", 20, y);

y += 20;

g.setFont(new Font("Arial", Font.PLAIN, 14));

for (Employee emp : employees) {

String line = emp.getId() + " | " + emp.getName() + " | " + emp.getEmail() + " | " + emp.getDepartment();

g.drawString(line, 20, y);

y += 25;

}

g.dispose();

// Convert to byte array

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ImageIO.write(image, "png", baos);

return baos.toByteArray();

} catch (Exception e) {

throw new RuntimeException("Failed to generate image: " + e.getMessage());

}

}

}