Leider ziemlich unbrauchbar. Die komplette OpenPDF Bibliothek aus dem Umfeld von LibreOffice, komplett falsch eingebunden. PDFs, werden leider dadurch keine erzeugt, Quellcode nicht komplierbar. Eine Dummy Adresse + Dummy Rechnung, die in der Datenbank gespeichert werden, stellt das „Main“ Programm da. Es fehlt jeglicher Hinweis, zu einem GUI für die Bearbeitung von Adressen und Rechnungen (Eingabe, etc.) . Das war alles nix! Tipp OpenPDF, wenn man das nicht kennt, Quellcode, der eventuell für die Erstellung eines PDFs mit OpenPDF, wenn die korrekten Importe von OpenPDF (Beispiele auf der Githubseite müssen zwingend angeschaut werden. Gemini - 12b, Google, lokal, schwierig für Codierung. Der Funktion, Emails per Sendmail dann zu versenden, wurde kommentarlos ignoriert.

To build a production-ready application with these requirements, you will need a few external libraries. Since you are using **Java 24**, we will utilize **Records** for clean data modeling and modern **Try-with-Resources** for database management.

Prerequisites

You will need to include the following dependencies in your pom.xml (Maven) or build.gradle (Gradle):

1. **SQLite JDBC Driver**: To connect to the SQLite database.
2. **OpenPDF**: A common library for generating PDF files in Java (since standard Java doesn't have a built-in PDF writer).

---

1. The Data Models

We use Java **Records** here. They are perfect for "Data Transfer Objects" (DTOs) because they are immutable and concise.

```
// Customer.java
public record Customer(int id, String name, String email, String address) {}
```

```
// Invoice.java
import java.time.LocalDate;

public record Invoice(int id, int customerId, double amount, LocalDate date) {}
```

2. Database Manager

This class handles the SQLite connection and table creation.

```
// DatabaseManager.java
import java.sql.*;

public class DatabaseManager {
    private static final String URL = "jdbc:sqlite:invoices.db";
```

```
public static Connection getConnection() throws SQLException {
    return DriverManager.getConnection(URL);
}

public static void initializeDatabase() {
    String customerTable = """
        CREATE TABLE IF NOT EXISTS customers (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            email TEXT,
            address TEXT
        );
        """;

    String invoiceTable = """
        CREATE TABLE IF NOT EXISTS invoices (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            customer_id INTEGER,
            amount REAL,
            date TEXT,
            FOREIGN KEY (customer_id) REFERENCES customers(id)
        );
        """;

    try (Connection conn = getConnection();
         Statement stmt = conn.createStatement()) {
        stmt.execute(customerTable);
        stmt.execute(invoiceTable);
    } catch (SQLException e) {
        System.err.println("DB Initialization Error: " + e.getMessage());
    }
}

public static void addCustomer(String name, String email, String address) throws SQLException {
    String sql = "INSERT INTO customers(name, email, address) VALUES(?, ?, ?)";
    try (Connection conn = getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, name);
        pstmt.setString(2, email);
        pstmt.setString(3, address);
        pstmt.executeUpdate();
    }
}

public static Customer getCustomerById(int id) throws SQLException {
    String sql = "SELECT * FROM customers WHERE id = ?";
    try (Connection conn = getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
```

```
            pstmt.setInt(1, id);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                return new Customer(rs.getInt("id"), rs.getString("name"),
                                    rs.getString("email"), rs.getString("address"));
            }
        }
        return null;
    }
}
```

3. Invoice Service (PDF Generation)

This handles the logic of creating the PDF. *Note: This assumes you have the OpenPDF library in your classpath.*

```
// InvoiceService.java
import com.lowagie.text.*;
import com.lowagie.text.pdf.*;
import java.io.FileOutputStream;

public class InvoiceService {
    public static void generatePdfInvoice(Customer customer, double amount, String filename) {
        Document document = new Document();
        try {
            PdfWriter.getInstance(document, new FileOutputStream(filename));
            document.open();

            Font font = Font.Font_BOLD;
            document.add(new Paragraph("INVOICE", font));
            document.add(new Paragraph("-----------------------------------------"));
            document.add(new Paragraph("Customer: " + customer.name()));
            document.add(new Paragraph("Email: " + customer.email()));
            document.add(new Paragraph("Address: " + customer.address()));
            document.add(new Paragraph("-----------------------------------------"));
            document.add(new Paragraph("Amount Due: $" + amount));
            document.add(new Paragraph("Date: " + java.time.LocalDate.now()));

            document.close();
            System.out.println("Invoice saved as: " + filename);
        } catch (Exception e) {
            System.err.println("Error generating PDF: " + e.getMessage());
        }
    }
}
```

4. Main Application

The entry point to run the tool.

```
// Main.java
import java.sql.SQLException;

public class Main {
    public static void main(String[] args) {
        try {
            // 1. Initialize DB
            DatabaseManager.initializeDatabase();
            System.out.println("Database initialized.");

            // 2. Add a dummy customer
            DatabaseManager.addCustomer("John Doe", "john@example.com", "123 Java Lane, Tech
City");
            System.out.println("Customer added.");

            // 3. Retrieve customer (Assuming ID 1 for this example)
            Customer customer = DatabaseManager.getCustomerById(1);

            if (customer != null) {
                // 4. Generate Invoice
                double amount = 1500.50;
                String fileName = "Invoice_" + customer.id() + ".pdf";
                InvoiceService.generatePdfInvoice(customer, amount, fileName);
            } else {
                System.out.println("Customer not found.");
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

How to run this:

1. **Database**: The code will automatically create a file named invoices.db in your project folder the first time it runs.
2. **PDFs**: Ensure you have a library like OpenPDF added to your project; otherwise, the InvoiceService will throw a "Class Not Found" error.
3. **Java 24**: Since you are using Java 24, you can use the --enable-preview flag if you decide to use any specific preview features, though the code above uses standard stable features.

As your **AI Assistant**, I recommend starting by getting the SQLite portion working first, then adding the PDF library once the database logic is stable!