Configure DB

Install Prisma Dependencies for SQLite

```
npm install prisma @types/node @types/better-sqlite3 -D
```

```
npm install @prisma/client @prisma/adapter-better-sqlite3 dotenv
```

Initialize Prisma ORM project

```
npx prisma init --datasource-provider sqlite --output ../generated/prisma
```

1. Open the `.env` file and update the db file name to `sample.db`
2. Open the `prisma/schema.prisma` file and add the following code at the end of the file.

```
model Customer {
    id          String   @id @default(uuid())
    name        String
    email       String   @unique
    phone       String
    dateOfBirth DateTime @default(now())
    address     String?
    createdAt   DateTime @default(now())

    // Relations
    loans     Loan[]
}

model Loan {
    id           String   @id @default(uuid())
    customerId   String
    loanType     LoanType
    amount       Float
    interestRate Float
    status       LoanStatus @default(ACTIVE)
    startDate    DateTime
    endDate      DateTime

    // Relations
    customer Customer @relation(fields: [customerId], references: [id])
    payments Payment[]
}

model Payment {
    id          String   @id @default(uuid())
    loanId      String
    amountPaid  Float
    paymentDate DateTime @default(now())
    method      PaymentMethod

    // Relations
    loan Loan @relation(fields: [loanId], references: [id])
}

// Enums for controlled values
enum LoanType {
    HOME
    CAR
    PERSONAL
    BUSINESS
}

enum LoanStatus {
    ACTIVE
    CLOSED
    DEFAULTED
}

enum PaymentMethod {
    CASH
    CARD
    BANKTRANSFER
    UPI
}
```

3. Create and apply the migration

```
npx prisma migrate dev --name init
```

4. Generate the db

```
npx prisma generate
```

5. Create `lib/db.ts` file and add the following code to it.

```
import "dotenv/config";
import { PrismaBetterSqlite3 } from "@prisma/adapter-better-sqlite3";
import { PrismaClient } from "../generated/prisma/client";
```

```
const connectionString = `${process.env.DATABASE_URL}`;

const adapter = new PrismaBetterSqlite3({ url: connectionString });

declare global {
// Prevent multiple instances of Prisma Client in development
// (Next.js hot-reloading can cause this issue)
var prisma: PrismaClient | undefined;
}

export const db = global.prisma || new PrismaClient({ adapter });

if (process.env.NODE_ENV !== "production") {
global.prisma = db;
}
```

Create model classes

1. Add the following code to `models/customer.ts` file

```
export interface Customer {
    id: string;
    name: string;
    email: string;
    phone: string;
    address: string;
    createdAt: Date;
}
```

2. Add the following code to `components/customerForm.ts`

```
"use client";

import { useState } from "react";

interface CustomerFormProps {
    onSubmit: (data: {
        name: string;
        email: string;
        phone: string;
        address?: string;
    }) => void;
}

export default function CustomerForm({ onSubmit }: CustomerFormProps) {
    const [formData, setFormData] = useState({
        name: "",
        email: "",
        phone: "",
        address: "",
    });

    function handleChange(e: React.ChangeEvent<HTMLInputElement>) {
        setFormData({ ...formData, [e.target.name]: e.target.value });
    }

    function handleSubmit(e: React.FormEvent) {
        e.preventDefault();
        onSubmit(formData);
    }

    return (
        <form onSubmit={handleSubmit} className="space-y-4 p-4 border
rounded">
        <input
            name="name"
            placeholder="Name"
            value={formData.name}
            onChange={handleChange}
            className="w-full border p-2"
        />
        <input
            name="email"
            placeholder="Email"
            value={formData.email}
            onChange={handleChange}
            className="w-full border p-2"
        />
        <input
            name="phone"
            placeholder="Phone"
            value={formData.phone}
            onChange={handleChange}
            className="w-full border p-2"
        />
        <input
            name="address"
            placeholder="Address"
            value={formData.address}
            onChange={handleChange}
            className="w-full border p-2"
        />
        <button type="submit" className="bg-blue-600 text-white px-4 py-2
rounded">
            Save Customer
        </button>
        </form>
    );
}
```

3. Add the following code to `services/customerService.ts`

```
import { db } from "../lib/db";

export async function getCustomers() {
    return await db.customer.findMany({
        include: { loans: true },
    });
}

export async function getCustomerById(id: string) {
    return await db.customer.findUnique({
        where: { id },
        include: { loans: true },
    });
}

export async function createCustomer(data: {
    name: string;
    email: string;
    phone: string;
    address?: string;
    }) {
    return await db.customer.create({ data });
}

export async function updateCustomer(id: string, data: Partial<{
    name: string;
    email: string;
    phone: string;
    address: string;
    }>) {
    return await db.customer.update({
        where: { id },
        data,
    });
}

export async function deleteCustomer(id: string) {
    return await db.customer.delete({ where: { id } });
}
```

4. Add the following code to `app/api/customers/route.ts` file

```
import { NextResponse } from "next/server";
import {
    getCustomers,
    createCustomer,
} from "../../../services/customerService";

// GET: Fetch all customers
export async function GET() {
    const customers = await getCustomers();
    return NextResponse.json(customers);
}

// POST: Create a new customer
export async function POST(req: Request) {
    const body = await req.json();
    const customer = await createCustomer(body);
    return NextResponse.json(customer);
}
```

5. Add the following code to `app/customers/add/page.tsx`

```
"use client";

import CustomerForm from "../../components/customerForm";

export default function CustomersPage() {
    async function handleCustomerSubmit(data: any) {
        await fetch("/api/customers", {
        method: "POST",
        body: JSON.stringify(data),
        });
    }

    return (
        <div>
        <h1 className="text-xl font-bold mb-4">Add Customer</h1>
        <CustomerForm onSubmit={handleCustomerSubmit} />
        </div>
    );
}
```

6. Add the following code to `app/customers/page.tsx`

```
"use client";

import { useEffect, useState } from "react";
import { Customer } from "@/models/customer";

export default function ListCustomersPage() {

    const [customers, setCustomers] = useState<Customer[]>([]);

    useEffect(() => {
        async function fetchCustomers() {
            const response = await fetch("/api/customers", {
                method: "GET",
            });
            const data = await response.json();
            setCustomers(data);
        }

        fetchCustomers();
    }, []);



    return (
        <div>
            <h1 className="text-xl font-bold mb-4"> Customers</h1>
            <table className="min-w-full bg-white">
                <thead>
                    <tr>
                        <th className="py-2">ID</th>
                        <th className="py-2">Name</th>
                        <th className="py-2">Email</th>
                    </tr>
                </thead>
                <tbody>
                    {customers.map((customer) => (
                        <tr key={customer.id}>
                            <td className="py-2">{customer.id}</td>
                            <td className="py-2">{customer.name}</td>
                            <td className="py-2">{customer.email}</td>
                        </tr>
                    ))}
                </tbody>
            </table>
        </div>
    );
}
```