Desafío Front-End – *Loan Tracker UI*

🎯Objetivo

Construir la interfaz web de un pequeño **Loan Tracker** aplicando las historias del *backlog* y el esquema provisto en **migration.sql**. El reto evalúa principalmente **UI/UX, calidad de código y arquitectura front-end**.

🛠️Stack obligatorio

Capa	Tecnología
Framework	**Next.js 15.3** (App Router + RSC)
Estilos	**Tailwind CSS 4.0**
UI Kit	**shadcn/ui**
Base de datos (mock)	**SQLite + Prisma ORM** (schema derivado de `migration.sql`)

> *Tip: expón tus consultas mediante **Route Handlers** (`/api/**`) o **Server Actions**; no se requiere un servidor aparte.*

📋Requisitos funcionales

Implementa la UI que cubra **todas las historias** del *backlog* entregado (alta / login / gestión de préstamos, dashboard con filtros, marcar devolución, etc.).

- Valida formularios y estados.
- Usa componentes de **shadcn/ui** con estilo consistente (light & dark mode).
- La lógica de persistencia puede ser sincrónica: SQLite embebido manejado por Prisma.

🚀Cómo partir (sugerido)

```
# 0. Requisitos: Node 20, pnpm
pnpm create next-app loan-tracker --ts --tailwind --app
cd loan-tracker

# 1. UI kit
pnpm dlx shadcn-ui@latest init

# 2. Dependencias principales
pnpm add lucide-react class-variance-authority tailwind-merge @tanstack/re
act-query better-sqlite3 prisma @prisma/client

# 3. Inicializa Prisma con SQLite
npx prisma init --datasource-provider sqlite
# → reemplaza prisma/schema.prisma con las tablas del migration.sql

# 4. Genera e impulsa la DB
npx prisma db push   # crea data.db según el schema

# 5. Dev server
pnpm dev
```

📦Entregables

1. **Repositorio GitHub público** con tu solución.
2. **README** claro (<500 palabras) con pasos de instalación y decisiones técnicas.
3. GIF corto o link a Vercel (bonus).

🏆Criterios de evaluación

Peso	Criterio
35 %	Código limpio TS, a11y, tests básicos
25 %	UI/UX (Responsive, dark mode, uso correcto de shadcn)
20 %	Arquitectura (carpetas por feature, separación client/server, Prisma bien utilizado)
10 %	Performance & DX (lazy-loading, lint, prettier)
10 %	Claridad de entrega (README, build OK)

⏳ Plazo

8 h efectivas dentro de los **7 días** siguientes a la recepción.

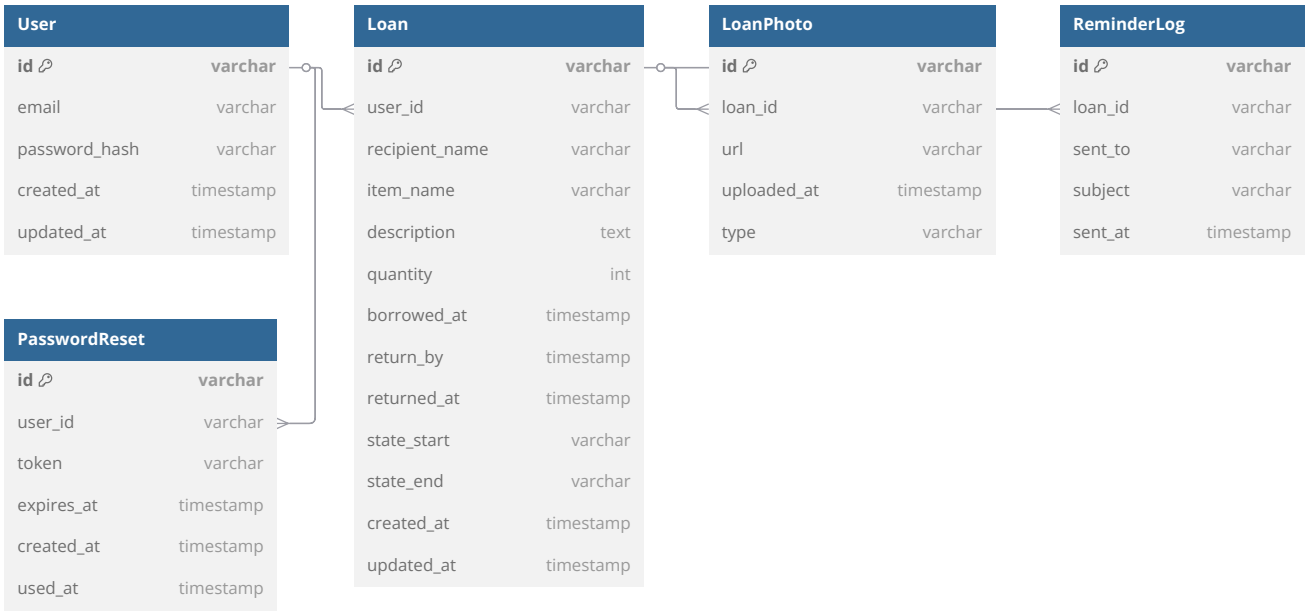
📄 Anexo: Backlog funcional

ID	Descripción	Historia de usuario	Comportamiento esperado	Validaciones clave
REQ-001	Registro de usuario	*Como nuevo usuario, quiero registrarme con mi email y contraseña para acceder a la plataforma.*	Permite crear cuenta y luego iniciar sesión.	Email válido y requerido. Contraseña ≥8 caracteres con mayúscula, minúscula y número.
REQ-002	Login	*Como usuario registrado, quiero iniciar sesión para gestionar mis préstamos.*	Valida credenciales y redirige al dashboard.	Email y contraseña requeridos; credenciales correctas.
REQ-003	Registrar préstamo	*Como usuario, quiero registrar un préstamo con detalles, condición, cantidad, fechas y fotos.*	Guarda el préstamo y lo marca como activo.	Nombre ítem y destinatario (≤100 chars). Cantidad > 0. Fecha préstamo ≤ hoy; devolución después de préstamo. Condición inicial requerida. Fotos JPG/ PNG ≤ 5 MB.
REQ-004	Dashboard	*Como usuario, quiero un resumen visual de mis préstamos para entender su estado.*	Muestra préstamos activos, vencidos y devueltos con filtros.	Solo usuarios autenticados. Filtros por estado y fecha válidos.
REQ-005	Marcar devolución	*Como usuario, quiero marcar un ítem como devuelto e incluir condición final y fotos.*	Actualiza el préstamo con fecha y condición final.	Solo préstamos activos. Condición final ≤ 200 chars. Fotos JPG/ PNG ≤ 5 MB.

📄Anexo: Esquema de Base de Datos (migration.sql)

A continuación, se presenta un diagrama visual que representa la estructura de la base de datos definida

en migration.sql y que será utilizada por Prisma ORM.



¡Éxito y buen código! 🎉