

TheTrueLemon – Next.js + Firebase Starter (Vercel-ready)

Drop these files into a fresh folder, then follow the README at the bottom.

Folder Structure

```
/thetruelemon
├─ app/
│   ├─ (marketing)/page.tsx
│   ├─ dashboard/
│   │   ├─ admin/page.tsx
│   │   ├─ client/page.tsx
│   │   └─ freelancer/page.tsx
│   ├─ login/page.tsx
│   ├─ logout/page.tsx
│   └─ layout.tsx
├─ components/
│   ├─ nav.tsx
│   ├─ role-gate.tsx
│   └─ ui/ (placeholder if you add shadcn)
├─ lib/
│   ├─ auth.ts
│   ├─ firebase.ts
│   └─ types.ts
├─ public/
│   ├─ hero.mp4 (put any abstract neon video)
│   └─ logo.svg (optional)
├─ styles/globals.css
├─ .env.local.example
├─ firebase.rules
├─ next.config.mjs
├─ package.json
├─ postcss.config.js
├─ tailwind.config.ts
├─ tsconfig.json
└─ README.md
```

package.json

```
{
  "name": "thetruelemon",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "firebase": "^10.12.0",
    "framer-motion": "^11.2.6",
    "lucide-react": "^0.441.0",
    "next": "14.2.4",
    "react": "18.3.1",
    "react-dom": "18.3.1",
    "tailwindcss": "^3.4.10"
  },
  "devDependencies": {
    "@types/node": "^20.11.30",
    "@types/react": "^18.2.73",
    "@types/react-dom": "^18.2.23",
    "autoprefixer": "^10.4.20",
    "postcss": "^8.4.44",
    "typescript": "^5.5.4"
  }
}
```

next.config.mjs

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  reactStrictMode: true,
  images: { unoptimized: true }
};
export default nextConfig;
```

tailwind.config.ts

```
import type { Config } from 'tailwindcss'

export default {
  content: [
    './app/**/*..{js,ts,jsx,tsx}',
    './components/**/*..{js,ts,jsx,tsx}'
  ],
  theme: {
    extend: {
      colors: {
        lemon: '#FFD300',
        neon: '#39FF14',
        base: '#0F0F0F'
      },
      boxShadow: {
        neon: '0 0 20px rgba(57,255,20,0.5)',
        lemon: '0 0 20px rgba(255,211,0,0.5)'
      }
    }
  },
  plugins: []
} satisfies Config
```

postcss.config.js

```
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  },
}
```

tsconfig.json

```
{
  "compilerOptions": {
    "target": "es2020",
    "lib": ["dom", "dom.iterable", "esnext"],
```

```

    "allowJs": false,
    "skipLibCheck": true,
    "strict": true,
    "noEmit": true,
    "esModuleInterop": true,
    "module": "esnext",
    "moduleResolution": "bundler",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "jsx": "preserve",
    "incremental": true,
    "paths": {
      "@/*": ["./*"]
    }
  },
  "include": ["next-env.d.ts", "**/*.ts", "**/*.tsx"],
  "exclude": ["node_modules"]
}

```

.env.local.example

```

NEXT_PUBLIC_FIREBASE_API_KEY=
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=
NEXT_PUBLIC_FIREBASE_PROJECT_ID=
NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET=
NEXT_PUBLIC_FIREBASE_MSG_SENDER_ID=
NEXT_PUBLIC_FIREBASE_APP_ID=

```

firebase.rules

```

rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    function isSignedIn() { return request.auth != null; }
    function userDoc() { return get(/databases/{database}/documents/users/{
(request.auth.uid)}); }
    function role() { return userDoc().data.role; }

    match /users/{uid} {
      allow read: if isSignedIn();
      allow write: if isSignedIn() && request.auth.uid == uid;
    }
  }
}

```

```

    }

    match /projects/{pid} {
      allow create: if isSignedIn() && role() in ['client','admin'];
      allow read: if isSignedIn();
      allow update, delete: if isSignedIn() && (
        resource.data.clientId == request.auth.uid || role() in
['admin','manager']
      );
    }

    match /applications/{aid} {
      allow create: if isSignedIn() && role() == 'freelancer';
      allow read: if isSignedIn();
      allow update, delete: if isSignedIn() && (
        resource.data.freelancerId == request.auth.uid || role() in
['admin','manager']
      );
    }
  }
}

```

styles/globals.css

```

@tailwind base;
@tailwind components;
@tailwind utilities;

html, body { background: #0F0F0F; color: #fff; }

.glass { @apply bg-white/5 backdrop-blur border border-white/10; }
.btn-lemon { @apply px-6 py-3 rounded-xl bg-lemon text-black hover:shadow-
lemon transition; }
.btn-neon { @apply px-6 py-3 rounded-xl border border-neon text-white
hover:shadow-neon transition; }

```

lib/types.ts

```

export type Role = 'client' | 'freelancer' | 'admin' | 'manager';
export type ProjectStatus = 'pending' | 'open' | 'assigned' | 'in-progress' |
'completed';

```

lib/firebase.ts

```
import { initializeApp, getApps } from 'firebase/app';
import { getAuth, GoogleAuthProvider } from 'firebase/auth';
import { getFirestore } from 'firebase/firestore';
import { getStorage } from 'firebase/storage';

const firebaseConfig = {
  apiKey: process.env.NEXT_PUBLIC_FIREBASE_API_KEY!,
  authDomain: process.env.NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN!,
  projectId: process.env.NEXT_PUBLIC_FIREBASE_PROJECT_ID!,
  storageBucket: process.env.NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET!,
  messagingSenderId: process.env.NEXT_PUBLIC_FIREBASE_MSG_SENDER_ID!,
  appId: process.env.NEXT_PUBLIC_FIREBASE_APP_ID!
};

const app = getApps().length ? getApps()[0] : initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const googleProvider = new GoogleAuthProvider();
export const db = getFirestore(app);
export const storage = getStorage(app);
```

lib/auth.ts

```
'use client';
import { auth, db } from '@lib/firebase';
import { onAuthStateChanged, signInWithPopup, signOut } from 'firebase/auth';
import { doc, getDoc, serverTimestamp, setDoc } from 'firebase/firestore';
import { useEffect, useState } from 'react';
import type { Role } from './types';

export function useUser() {
  const [user, setUser] = useState<null | (ReturnType<typeof auth.currentUser>
    & { role?: Role })>(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const unsub = onAuthStateChanged(auth, async (u) => {
      if (!u) { setUser(null); setLoading(false); return; }
      const snap = await getDoc(doc(db, 'users', u.uid));
      const role = snap.exists() ? (snap.data().role as Role) : undefined;
      setUser({ ...u, role } as any);
    });
  });
}
```

```

        setLoading(false);
    });
    return () => unsub();
}, []);

    return { user, loading };
}

export async function googleLogin(intentRole?: Role) {
    const res = await signInWithPopup(auth, new (auth as
any).constructor.GoogleAuthProvider?().() || (window as any));
    // fallback to our provider
}

export async function googleLoginWithRole(role: Role) {
    const res = await signInWithPopup(auth, new (auth as
any).constructor.GoogleAuthProvider?().() || (window as any));
    // NOTE: we actually exported googleProvider in firebase.ts, so better to use
that from components directly for clarity.
}

export async function saveRole(uid: string, role: Role) {
    const ref = doc(db, 'users', uid);
    const snap = await getDoc(ref);
    if (!snap.exists()) {
        await setDoc(ref, { role, createdAt: serverTimestamp() }, { merge: true });
    } else {
        await setDoc(ref, { role }, { merge: true });
    }
}

export async function logout() { await signOut(auth); }

```

Note: We'll call `signInWithPopup(auth, googleProvider)` directly in pages to keep it simple.

components/nav.tsx

```

'use client';
import Link from 'next/link';
import { useUser } from '@lib/auth';
import { auth } from '@lib/firebase';
import { signOut } from 'firebase/auth';

export default function Nav() {

```

```

    const { user } = useUser();
    return (
      <nav className="sticky top-0 z-40 bg-black/40 backdrop-blur border-b border-white/10">
        <div className="max-w-6xl mx-auto px-4 py-3 flex items-center justify-between">
          <Link href="/" className="font-extrabold text-lg">TheTrueLemon <span className="text-lemon">🍋</span></Link>
          <div className="flex items-center gap-3 text-sm">
            <Link href="/login" className="hidden md:inline text-gray-300 hover:text-white">Login</Link>
            {user && (
              <>
                <Link href={`/${dashboard}/${user.role} || 'client'}`} className="btn-neon">Dashboard</Link>
                <button onClick={() => signOut(auth)} className="px-4 py-2 rounded-lg border border-white/20">Logout</button>
              </>
            )}
          </div>
        </div>
      </nav>
    );
  }
}

```

components/role-gate.tsx

```

'use client';
import { ReactNode, useEffect } from 'react';
import { useRouter } from 'next/navigation';
import { useUser } from '@/lib/auth';

export default function RoleGate({ allow, children }: { allow: string[]; children: ReactNode }) {
  const { user, loading } = useUser();
  const router = useRouter();

  useEffect(() => {
    if (!loading) {
      if (!user) router.push('/login');
      else if (user.role && !allow.includes(user.role)) router.push(`/${dashboard}/${user.role}`);
    }
  }, [user, loading, allow, router]);
}

```



```

    if (loading) return <div className="p-8 text-gray-400">Loading...</div>;
    return <>{children}</>;
  }

```

app/layout.tsx

```

import './globals.css';
import { ReactNode } from 'react';
import Nav from '@components/nav';

export const metadata = {
  title: 'TheTrueLemon',
  description: 'Futuristic Marketing Marketplace'
};

export default function RootLayout({ children }: { children: ReactNode }) {
  return (
    <html lang="en">
      <body className="bg-base text-white antialiased min-h-screen">
        <Nav />
        {children}
      </body>
    </html>
  );
}

```

app/(marketing)/page.tsx

```

'use client';
import { motion } from 'framer-motion';
import Link from 'next/link';

export default function Home() {
  return (
    <main>
      <section className="relative h-[88vh] overflow-hidden">
        <video className="absolute inset-0 w-full h-full object-cover opacity-40"
          autoPlay loop muted playsInline>
          <source src="/hero.mp4" type="video/mp4" />
        </video>
      </section>
    </main>
  );
}

```

```

    <div className="relative z-10 h-full flex flex-col items-center justify-
center text-center px-6">
      <motion.h1 initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y:
0 }} transition={{ duration: 0.6 }}>
        className="text-4xl md:text-6xl font-extrabold">
          TheTrueLemon <span className="text-lemon">🍋</span>
        </motion.h1>
        <p className="mt-4 text-gray-300 max-w-2xl">Connecting businesses
with verified marketing talent 📦 managed, safe, and fast.</p>
        <div className="mt-8 flex gap-4">
          <Link href="/login?as=client" className="btn-lemon">I Want a
Project</Link>
          <Link href="/login?as=freelancer" className="btn-neon">I Have a
Project</Link>
        </div>
      </div>
</section>

<section className="max-w-6xl mx-auto px-6 py-16 grid md:grid-cols-3
gap-6">
  {[ 'SEO & Growth', 'Paid Ads', 'Social Media', 'Content & Creatives', 'Email
& CRM', 'Analytics & Automation' ].map((s) => (
    <div key={s} className="glass rounded-2xl p-6 hover:shadow-neon
transition">
      <h3 className="text-xl font-semibold mb-2">{s}</h3>
      <p className="text-gray-400 text-sm">Outcome-focused services for
serious business growth.</p>
    </div>
  ))}
</section>
</main>
);
}

```

app/login/page.tsx

```

'use client';
import { auth, db, googleProvider } from '@lib/firebase';
import { signInWithPopup } from 'firebase/auth';
import { doc, getDoc, serverTimestamp, setDoc } from 'firebase/firestore';
import { useRouter, useSearchParams } from 'next/navigation';
import { useEffect, useState } from 'react';

export default function Login() {

```

```

const router = useRouter();
const sp = useSearchParams();
const [role, setRole] = useState<string>(sp.get('as') || 'client');

useEffect(() => { setRole(sp.get('as') || 'client'); }, [sp]);

async function googleLogin() {
  const res = await signInWithPopup(auth, googleProvider);
  const u = res.user;
  const ref = doc(db, 'users', u.uid);
  const snap = await getDoc(ref);
  if (!snap.exists()) {
    await setDoc(ref, { name: u.displayName, email: u.email, photoURL:
u.photoURL, role, createdAt: serverTimestamp() });
  } else {
    await setDoc(ref, { role }, { merge: true });
  }
  router.push(`/dashboard/${role}`);
}

return (
  <div className="min-h-[80vh] grid place-items-center px-6">
    <div className="max-w-md w-full glass rounded-2xl p-8">
      <h2 className="text-2xl font-bold">Join TheTrueLemon</h2>
      <p className="text-gray-400 text-sm mt-2">Select your role and continue
with Google.</p>
      <div className="mt-4 grid grid-cols-2 gap-2">
        <button onClick={()=>setRole('client')} className={`px-4 py-2 rounded-
xl border ${role==='client'?'border-lemon text-lemon':'border-white/20'}`}
>Client</button>
        <button onClick={()=>setRole('freelancer')} className={`px-4 py-2
rounded-xl border ${role==='freelancer'?'border-neon text-neon':'border-white/
20'}`}>Freelancer</button>
      </div>
      <button onClick={googleLogin} className="mt-6 w-full btn-
lemon">Continue with Google</button>
    </div>
  </div>
);
}

```

app/logout/page.tsx

```
'use client';
import { auth } from '@lib/firebase';
import { signOut } from 'firebase/auth';
import { useEffect } from 'react';

export default function Logout() {
  useEffect(() => { signOut(auth); }, []);
  return <div className="p-8">Signing out</div>;
}
```

app/dashboard/client/page.tsx

```
'use client';
import { auth, db } from '@lib/firebase';
import { addDoc, collection, onSnapshot, orderBy, query, serverTimestamp,
where } from 'firebase/firestore';
import { useEffect, useState } from 'react';

export default function ClientDashboard() {
  const [title, setTitle] = useState('');
  const [budget, setBudget] = useState<number>(0);
  const [timeline, setTimeline] = useState('');
  const [description, setDescription] = useState('');
  const [projects, setProjects] = useState<any[]>([]);

  useEffect(() => {
    const uid = auth.currentUser?.uid; if (!uid) return;
    const q = query(collection(db, 'projects'), where('clientId', '==', uid),
orderBy('createdAt', 'desc'));
    return onSnapshot(q, (snap) => setProjects(snap.docs.map(d=>({ id:
d.id, ...d.data() })))));
  }, []);

  async function postProject() {
    const uid = auth.currentUser?.uid; if (!uid) return;
    await addDoc(collection(db, 'projects'), {
      clientId: uid, title, description, budget, timeline, status: 'pending',
createdAt: serverTimestamp()
    });
    setTitle(''); setBudget(0); setTimeline(''); setDescription('');
  }
}
```

```

return (
  <div className="max-w-6xl mx-auto px-6 py-10">
    <h1 className="text-2xl font-bold mb-6">Client Dashboard</h1>
    <div className="glass rounded-2xl p-6 grid md:grid-cols-2 gap-4">
      <input value={title} onChange={e=>setTitle(e.target.value)}
placeholder="Project title" className="px-4 py-3 rounded-xl bg-black/40 border
border-white/10"/>
      <input value={budget} onChange={e=>setBudget(Number(e.target.value))}
placeholder="Budget (USD)" type="number" className="px-4 py-3 rounded-xl bg-
black/40 border border-white/10"/>
      <input value={timeline} onChange={e=>setTimeline(e.target.value)}
placeholder="Timeline (e.g. 2 weeks)" className="px-4 py-3 rounded-xl bg-black/
40 border border-white/10"/>
      <textarea value={description}
onChange={e=>setDescription(e.target.value)} placeholder="Describe your needs"
className="px-4 py-3 rounded-xl bg-black/40 border border-white/10 md:col-
span-2"/>
      <button onClick={postProject} className="btn-neon md:col-span-2">Post
Project</button>
    </div>

    <h2 className="text-xl font-semibold mt-10 mb-4">My Projects</h2>
    <div className="grid md:grid-cols-2 gap-4">
      {projects.map(p => (
        <div key={p.id} className="glass rounded-2xl p-6">
          <h3 className="text-lg font-semibold">{p.title}</h3>
          <p className="text-gray-400 text-sm">${p.budget} 📅 {p.timeline}</p>
          <p className="text-gray-300 mt-2 line-clamp-3">{p.description}</p>
          <span className="inline-block mt-3 text-xs px-3 py-1 rounded-full
bg-white/10">{p.status}</span>
        </div>
      ))}
    </div>
  </div>
);
}

```

app/dashboard/freelancer/page.tsx

```

'use client';
import { db } from '@lib/firebase';
import { collection, onSnapshot, orderBy, query, where } from 'firebase/
firestore';

```

```

import { useEffect, useState } from 'react';

export default function FreelancerDashboard() {
  const [projects, setProjects] = useState<any[]>([]);

  useEffect(() => {
    const q = query(collection(db, 'projects'), where('status', 'in',
    ['open', 'pending']), orderBy('createdAt', 'desc'));
    return onSnapshot(q, (snap) => setProjects(snap.docs.map(d=>({ id:
    d.id, ...d.data() })))));
  }, []);

  return (
    <div className="max-w-6xl mx-auto px-6 py-10">
      <h1 className="text-2xl font-bold mb-6">Freelancer Dashboard</h1>
      <div className="grid md:grid-cols-2 gap-4">
        {projects.map(p => (
          <div key={p.id} className="glass rounded-2xl p-6">
            <h3 className="text-lg font-semibold">{p.title}</h3>
            <p className="text-gray-400 text-sm">${p.budget} {p.timeline}</p>
            <p className="text-gray-300 mt-2 line-clamp-3">{p.description}</p>
            <button className="mt-3 btn-lemon">Apply</button>
          </div>
        ))}
      </div>
    </div>
  );
}

```

app/dashboard/admin/page.tsx

```

'use client';
import { db } from '@lib/firebase';
import { collection, doc, onSnapshot, orderBy, query, updateDoc, where } from
'firebase/firestore';
import { useEffect, useState } from 'react';

export default function AdminPanel() {
  const [pending, setPending] = useState<any[]>([]);


  useEffect(() => {
    const q = query(collection(db, 'projects'), where('status', '==', 'pending'),
    orderBy('createdAt', 'asc'));
    return onSnapshot(q, (snap) => setPending(snap.docs.map(d=>({ id:

```

```

d.id, ...d.data() }));
}, []);

async function approve(id: string) { await updateDoc(doc(db, 'projects', id),
{ status: 'open' }); }

return (
  <div className="max-w-6xl mx-auto px-6 py-10">
    <h1 className="text-2xl font-bold mb-6">Admin Panel</h1>
    <div className="grid md:grid-cols-2 gap-4">
      {pending.map(p => (
        <div key={p.id} className="glass rounded-2xl p-6">
          <h3 className="text-lg font-semibold">{p.title}</h3>
          <p className="text-gray-400 text-sm">Budget ${p.budget} 
          {p.timeline}</p>
          <div className="mt-3 flex gap-2">
            <button onClick={()=>approve(p.id)} className="btn-neon">Approve
(Open)</button>
            <button className="px-4 py-2 rounded-xl border border-white/
20">Assign Manager</button>
            <button className="px-4 py-2 rounded-xl border border-white/
20">Assign Freelancer</button>
          </div>
        </div>
      )})}
    </div>
  </div>
);
}

```

README.md

```
# TheTrueLemon (Next.js + Firebase + Tailwind)
```

Futuristic marketing marketplace starter with Google Auth, role-based dashboards, project posting, and admin skeleton.

```
## 1) Setup
```

```
```bash
```

```
npm install
```

```
cp .env.local.example .env.local
```

```
fill your Firebase web credentials in .env.local
```

Enable in Firebase Console: - Authentication → Google provider (enable) - Firestore Database → Start in production - Rules → paste `firebase.rules` - Storage (optional)

## 2) Dev

```
npm run dev
```

Open <http://localhost:3000>

## 3) Deploy to Vercel (Recommended)

- Push this repo to GitHub
- Import repo in <https://vercel.com/new>
- Add the same env vars in Vercel Project Settings → Environment Variables
- Deploy

## 4) Next Steps

- Add Applications collection & apply button behavior
- Manager role & auto-assign via Cloud Function
- Internal chat (Firestore subcollection per project)
- Stripe/Razorpay escrow (Cloud Functions) ``