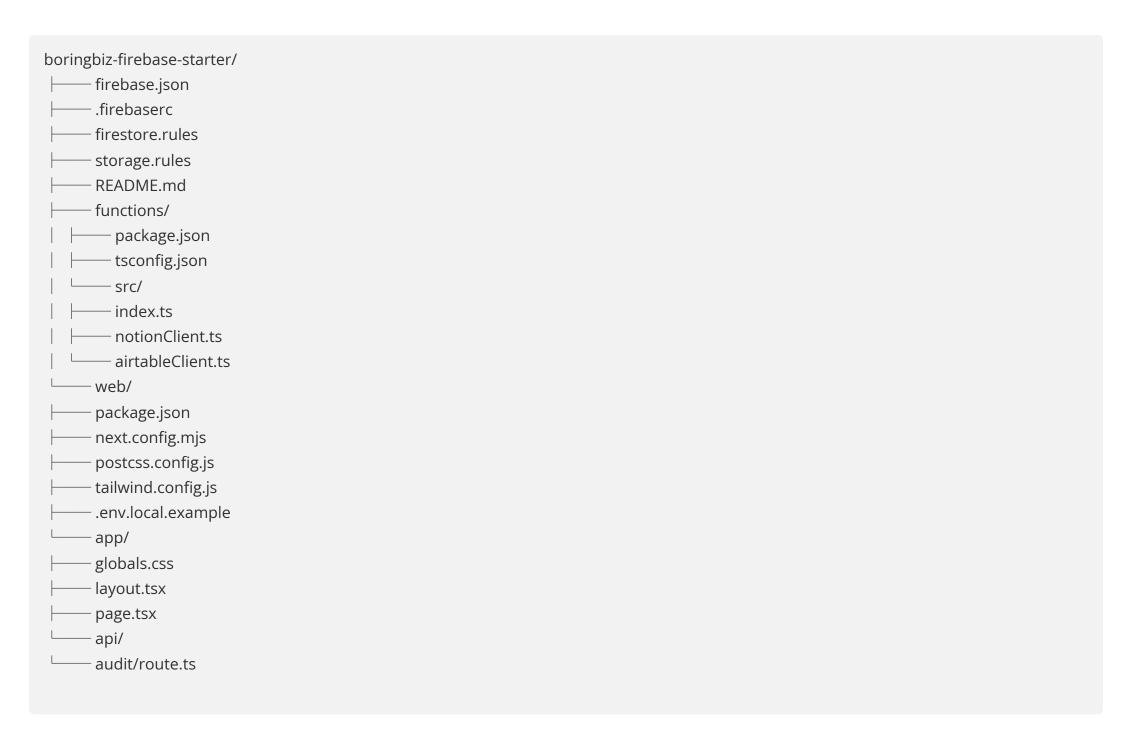
BoringBiz.ai Firebase Scaffolding

Awesome — here's the **full BoringBiz.ai Firebase scaffolding** as **rich text with code blocks** so you can copy/paste into Firebase Studio (no ZIP needed). This is the **no-auth MVP** wired for Hosting + Cloud Functions + Firestore + Next.js 14 (App Router), with optional Notion & Airtable hooks.

Project Tree (for reference)



Root files

firebase.json

```
{
 "functions": {
  "source": "functions"
 },
 "hosting": {
  "public": "web/out",
  "ignore": ["firebase.json", "**/.*", "**/node_modules/**"],
  "headers": [{
   "source": "**/*.@(css|js)",
   "headers": [{
    "key": "Cache-Control",
    "value": "public,max-age=31536000,immutable"
   }]
  }]
 },
 "firestore": {
  "rules": "firestore.rules"
 },
 "storage": {
  "rules": "storage.rules"
```

ſ

.firebaserc

```
"projects": {
    "default": "YOUR_PROJECT_ID"
    }
}

firestore_rules
```

rules_version = '2'; service cloud.firestore {

```
match /databases\{database\}\documents {
  match /audits\{docld\} {
    allow read: if true; // demo; tighten later
    allow write: if true; // demo; tighten later
}

match /leads\{docld\} {
  allow read, write: if request.auth != null;
}
}

storage_rules
```

rules_version = '2'; service firebase.storage { match /b/{bucket}/o {

match /{allPaths=**} {

allow read: if true; // demo

"notionDbId":"optional",
"notionToken":"optional",
"airtableBaseId":"optional",
"airtableToken":"optional"

functions/package.json

"module": "ES2022",

"esModuleInterop": true,

"outDir": "lib",
"rootDir": "src",

"strict": true

"moduleResolution": "Bundler",

"name": "boringbiz-functions",

allow write: if request.auth != null;

```
README.md
  # BoringBiz.ai — Firebase Starter (No-Auth MVP)
  Stack: Firebase Hosting · Cloud Functions (Node 20, TS) · Firestore · Next.js 14 (App Router) · Tailwind
  ## Quickstart
  ```bash
 npm i -g firebase-tools # install deps
 cd functions && npm i && cd ../web && npm i && cd .. # connect project
 firebase use --add # env for web
 cp web/.env.local.example web/.env.local # fill Firebase Web SDK fields
 # build static web
 cd web && npm run build:static && cd .. # deploy
 firebase deploy --only functions, hosting # finally
 edit web/app/api/audit/route.ts -> replace YOUR_PROJECT_ID
 ### API
 POST https://us-central1-.cloudfunctions.net/auditUrl
 body:
 "url": "https://example.com",
 "projectId": "default",
```

### "type": "module",

**Cloud Functions (Node 20, TypeScript)** 

```
"engines": {
 "node": "20"
 "main": "lib/index.js",
 "scripts": {
 "build": "tsc -p tsconfig.json",
 "serve": "npm run build && firebase emulators:start --only functions",
 "deploy": "npm run build && firebase deploy --only functions"
 "dependencies": {
 "axios": "^1.7.2",
 "cheerio": "^1.0.0-rc.12",
 "firebase-admin": "^12.5.0",
 "firebase-functions": "^4.9.0",
 "metascraper": "^5.44.7",
 "metascraper-title": "^5.44.7",
 "metascraper-description": "^5.44.7",
 "metascraper-url": "^5.44.7",
 "notionhq": "^2.2.14",
 "airtable": "^0.12.2",
 "cors": "^2.8.5"
 "devDependencies": {
 "typescript": "^5.5.4"
functions/tsconfig.json
 "compilerOptions": {
 "target": "ES2022",
```

```
functions/src/index.ts
 import * as functions from 'firebase-functions';
 import * as admin from 'firebase-admin';
 import axios from 'axios';
 import * as cheerio from 'cheerio';
 import metascraper from 'metascraper';
 import metascraperTitle from 'metascraper-title';
 import metascraperDescription from 'metascraper-description';
 import metascraperUrl from 'metascraper-url';
 import { logAuditToNotion } from './notionClient.js';
 import { upsertCompetitorToAirtable } from './airtableClient.js';
 const scraper = metascraper([metascraperTitle(), metascraperDescription(), metascraperUrl()]);
 admin.initializeApp();
 const db = admin.firestore();
 function validUrl(u: string): boolean {
 try {
 new URL(u);
 return true;
```

```
} catch {
 return false;
 }
export const auditUrl = functions
.region('us-central1')
.runWith({ memory: '512MB', timeoutSeconds: 120 })
.https.onRequest(async (req, res) => {
 res.set('Access-Control-Allow-Origin','*');
 res.set('Access-Control-Allow-Headers','Content-Type');
 if (req.method === 'OPTIONS') return res.status(204).send(");
 try {
 const { url, projectId = 'default', notionDbId, notionToken, airtableBaseId, airtableToken } = req.body | | {};
 if (!url | | !validUrl(url)) return res.status(400).json({ error: "Provide a valid 'url'." });
 const resp = await axios.get(url, { timeout: 20000, headers: { 'User-Agent':'Mozilla/5.0' } });
 const html = resp.data as string;
 const $ = cheerio.load(html);
 const meta = await scraper({ html, url });
 const h1 = $('h1').first().text().trim();
 const title = meta.title | | $('title').text().trim();
 const description = meta.description || $('meta[name="description"]').attr('content') || ";
 const bodyText = $('body').text().replace(/\s+/g,' ').trim();
 const wordCount = bodyText.split(' ').filter(Boolean).length;
 const imagesCount = $('img').length;
 const scriptsCount = $('script').length;
 const hasWebp = $("img[src$='.webp']").length > 0;
 const\ links = Array.from(new\ Set(\$('a[href]').map((_,a)=>String(\$(a).attr('href'))).get())).slice(0,100);
 const doc = {
 url,
 projectId,
 fetchedAt: admin.firestore.FieldValue.serverTimestamp(),
 title,
 description,
 h1,
 wordCount,
 speedHints: {
 imagesCount,
 scriptsCount,
 hasWebp
 },
 links
 };
 const ref = await db.collection('audits').add(doc);
 // Optional sinks
 await logAuditToNotion({ notionDbld, notionToken, title, url, projectId, wordCount });
 await upsertCompetitorToAirtable({ baseld: airtableBaseld, token: airtableToken, url, title });
 return res.json({ id: ref.id, ...doc });
 } catch (e:any) {
 console.error(e);
 return res.status(500).json({ error: e.message | | 'Failed to audit URL' });
 }
});
 functions/src/airtableClient.ts
```

```
functions/src/notionClient.ts
 export async function logAuditToNotion({
 import Airtable from 'airtable';
 notionDbld,
 notionToken,
 export async function upsertCompetitorToAirtable({
 title,
 baseld,
 url,
 token,
 url,
 projectld,
 wordCount
 title
 }:{
 }:{
 notionDbld?: string,
 baseld?: string,
 notionToken?: string,
 token?: string,
 title?: string,
 url: string,
 title?: string
 url: string,
 projectId?: string,
 }){
 wordCount?: number
 if (!baseId | | !token) return;
 }){
 if (!notionDbld | | !notionToken) return;
 const base = new Airtable({ apiKey: token }).base(baseld);
 const { Client } = await import('@notionhq/client');
 // naive insert; enhance to find/update if needed
 const notion = new Client({ auth: notionToken });
 await base('Competitors').create([{
 fields: {
 await notion.pages.create({
 URL: url,
 parent: { database_id: notionDbld },
 Title: title || url
 properties: {
 }
 Name: { title: [{ text: { content: title | | url } }] },
 }]);
 }
 Project: { rich_text: [{ text: { content: projectId | | 'default' } }] },
 WordCount: { number: wordCount | | 0 }
 }
```

**})**;

```
{data.title || data.url}

{data.description}

Words: {data.wordCount}

Imgs: {data.speedHints?.imagesCount}

Scripts: {data.speedHints?.scriptsCount}

Links (sample)
```

{JSON.stringify(data.links,null,2)}

### **Deploy Steps (copy/paste)**

#### **Install Dependencies**

npm i -g firebase-tools # install cd functions && npm i && cd ../web && npm i && cd ..

#### **Connect Project**

firebase use --add

### **Configure Web Environment**

cp web/.env.local.example web/.env.local
# fill Firebase Web SDK fields (Project Settings → Web App)

#### **Build Static Export**

cd web && npm run build:static && cd ..

#### **Deploy & Update**

firebase deploy --only functions,hosting
# update proxy target
# edit web/app/api/audit/route.ts and replace YOUR\_PROJECT\_ID with your real id