

You said:

Yes, I recently just purchased `yooahoo.guru`. which is like calling a guru to help out. I think im going to use it as the name of a sasquatch or yeti handyman. He rides a Harley with his yeti wife Angel on the back of it. She has a little section of the site called Angel's list who handles handyman or label rentals. Yooahoo handles the skill swapping service. All categories of skills or services could be then setup like `cooking.yooahoo.guru`, `painting.yooahoo.guru`, and each category could be represented by a cousin or other yeti family member.

ChatGPT said:

That's a killer concept, Brandon — mixing a brandable domain (`yooahoo.guru`) with a mascot-driven universe (the Yeti family) and the **nested domain** idea. Let me break down how you could actually set this up so it works as both a **business** and a **passive income machine**.

Concept Blueprint: YooHoo Guru & Family

- **Main Brand:** `yooahoo.guru` → “Call a guru to help out.”
 - **Mascot:** Sasquatch handyman *YooHoo*, Harley-riding with his wife *Angel*.
 - **Angel's List:** A section for rentals / odd jobs (like “handyman Yellow Pages”).
 - **Nested Cousins:** Each cousin (another Yeti) runs a niche subdomain.
 - `cooking.yooahoo.guru` → Chef Yeti
 - `painting.yooahoo.guru` → Painter Yeti
 - `plumbing.yooahoo.guru` → Uncle Pipewrench
 - etc.
-

How to Build It (Step-by-Step)

1. Set Up the Domain Nest

- Register `yooahoo.guru` (already done).
- In your DNS or hosting panel, create wildcard subdomains (`*.yooahoo.guru`).
- This way, you can spin up new “cousins” instantly.

2. Content & Characters

- Each subdomain has:
 - A **Yeti family character** (drawn/AI-generated).
 - Content tailored to that skill.
 - Links back to `yoohoo.guru` (main brand).
- Style: light humor + helpful resources (keeps bounce rates down).

3. Monetization Streams

- **Skill Swap SaaS:** Main site could be like a “time bank” (people trade skills).
- **Affiliate Products:** Subdomains can link to tools, courses, Amazon gear.
- **Ads:** Once traffic grows, run Google AdSense / Ezoic.
- **Premium Listings:** Angel’s List can charge for handyman listings or rentals.
- **Membership:** Paid “Guru Pass” → discounts, early access, no ads.

4. Automation

- Use AI to generate starter content (articles, FAQs, guides).
- Automate posting (cron jobs, CMS scheduling).
- Auto-embedding of affiliate widgets or lead forms.

5. SEO Advantage of Nesting

- Internal links between subdomains build authority.
- Each cousin site can rank for its niche keywords.
- The parent site (`yoohoo.guru`) becomes the umbrella “brand authority.”

Passive Income Math

- One subdomain with **10–20 decent AI-human hybrid articles** can pull in:
 - **\$10–\$100/month** from ads/affiliates (conservative).
 - With **10 active cousins**, that’s **\$100–\$1,000/month**.
 - With **30+ cousins**, you could realistically scale toward a few thousand/month.
-



Your Yeti family can be the **branding glue**:

- Visitors remember the characters (fun, relatable).
 - It humanizes what could otherwise be a generic SEO farm.
 - Merch angle: T-shirts, mugs, stickers (Sasquatch handyman + Angel biker theme).
-

Next Steps

1. **Decide initial 3 niches** (skills/services you want to launch first).
 2. **Design mascots** (AI images or commission an artist for consistency).
 3. **Launch main hub** (yooahoo.guru with Angel's List + Skill Swap engine).
 4. **Spin up 3 subdomains** and populate with content.
 5. **Monetize** (ads, affiliate, premium listing options).
 6. Rinse & repeat for other Yeti cousins.
-

👉 Do you want me to mock up a **site architecture diagram** (main hub + cousin subdomains + monetization flows), so you can see how the nested domain system works visually for YooHoo Guru?

You said:

Yes, please

ChatGPT said:

```
Always show details
# Create a simple site architecture diagram for YooHoo Guru as a PNG using
matplotlib.
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle, FancyArrowPatch
import matplotlib.lines as mlines

fig, ax = plt.subplots(figsize=(12, 8))

def add_box(x, y, w, h, text, fontsize=10):
    rect = Rectangle((x, y), w, h, fill=False)
    ax.add_patch(rect)
    ax.text(x + w/2, y + h/2, text, ha='center', va='center',
            fontsize=fontsize)
    return rect
```

```

def add_arrow(x1, y1, x2, y2):
    arrow = FancyArrowPatch((x1, y1), (x2, y2), arrowstyle='->',
mutation_scale=12, linewidth=1)
    ax.add_patch(arrow)

# Main Hub
hub = add_box(3.8, 6.8, 4.4, 1.0, "Main Hub: yoohoo.guru\n• Landing +
Brand\n• Skill Swap (SaaS)\n• Angel's List (Rentals/Jobs)\n• Auth +
Payments")

# Core Services under Hub
auth = add_box(0.4, 5.1, 3.5, 0.9, "Core Services\n• Auth/Profiles\n•
Payments (Stripe)\n• CMS (blog/pages)\n• Search/Index")
market = add_box(4.2, 5.1, 3.5, 0.9, "Marketplace Modules\n• Listings\n•
Requests/Offers\n• Messaging\n• Reviews")
analytics = add_box(8.0, 5.1, 3.5, 0.9, "Ops & Analytics\n• Admin Console\n•
Metrics/SEO\n• Moderation\n• Email/CRM")

# Arrows from Hub to Core
add_arrow(6.0, 6.8, 2.2, 6.0) # to Core Services
add_arrow(6.0, 6.8, 6.0, 6.0) # to Marketplace
add_arrow(6.0, 6.8, 9.8, 6.0) # to Analytics

# Subdomains (Yeti Cousins)
cooking = add_box(0.2, 3.1, 3.4, 0.9, "cooking.yoohoo.guru\n• Chef Yeti\n•
Recipes/Classes\n• Affiliate + Ads\n• Booking")
painting = add_box(4.0, 3.1, 3.4, 0.9, "painting.yoohoo.guru\n• Painter
Yeti\n• Tutorials/Guides\n• Tool Affiliates\n• Booking")
plumbing = add_box(7.8, 3.1, 3.4, 0.9, "plumbing.yoohoo.guru\n• Uncle
Pipewrench\n• Fix-It Guides\n• Part Affiliates\n• Booking")

# Arrows from Services to Subdomains
add_arrow(2.2, 5.1, 1.9, 4.0) # core -> cooking
add_arrow(6.0, 5.1, 5.7, 4.0) # market -> painting
add_arrow(9.8, 5.1, 9.5, 4.0) # analytics -> plumbing

# Cross-links between subdomains
add_arrow(3.6, 3.55, 4.0, 3.55)
add_arrow(7.4, 3.55, 7.8, 3.55)

# Monetization Layer
ads = add_box(0.8, 1.3, 3.0, 0.9, "Monetization\n• Ads (AdSense/Ezoic)\n•
Affiliate (Amazon/Tools)\n• Lead Gen\n• Membership")
seo = add_box(4.6, 1.3, 3.0, 0.9, "SEO & Content Engine\n• Keyword
Research\n• AI-Assisted Posts\n• Internal Linking\n• Sitemaps")
infra = add_box(8.4, 1.3, 3.0, 0.9, "Infra\n• Hosting/CDN\n• SSL/Wildcard\n•
Backups\n• Observability")

# Arrows from subdomains to monetization
add_arrow(1.9, 3.1, 2.3, 2.2)
add_arrow(5.7, 3.1, 6.1, 2.2)
add_arrow(9.5, 3.1, 9.9, 2.2)

# Legend / Notes
ax.text(0.2, 0.3, "Notes:\n1) Wildcard DNS (*.yoohoo.guru) for instant
subdomain spin-up.\n2) Shared auth/payments so users have one account across
cousins.\n3) Each subdomain links to others + hub to compound SEO.\n4) Start

```

```
with 3 cousins; add more monthly.\n5) Angel's List = premium listings &
rentals.", fontsize=9, va='top')
```

```
# Aesthetics
ax.set_xlim(0, 12)
ax.set_ylim(0, 8)
ax.axis('off')
```

```
plt.tight_layout()
out_path = "/mnt/data/YooHoo_Guru_Architecture.png"
plt.savefig(out_path, dpi=200, bbox_inches='tight')
out_path
```

Result

```
'/mnt/data/YooHoo_Guru_Architecture.png'
```


Here's a clean, high-level **site architecture diagram** for YooHoo Guru (hub + Yeti cousin subdomains + monetization/ops).
Download the diagram (PNG)

Rollout plan (fast + realistic)

1. Domain & DNS

- Enable wildcard DNS: `*.yooahoo.guru` → points to your app/load balancer.
- SSL: issue a wildcard cert for `*.yooahoo.guru` + apex `yooahoo.guru`.

2. Core platform (single codebase, multi-tenant by subdomain)

- Router reads `req.subdomains[0]` to select the active “cousin.”
- Shared services: Auth (email + OAuth), Profiles, Payments (Stripe), Listings, Messages, Reviews.
- CMS for posts/pages; per-subdomain themes (logo/mascot, palette, copy).

3. Hub features (`yooahoo.guru`)

- Landing + brand story (YooHoo & Angel).
- **Skill-Swap** module (requests/offers, calendar, escrow/credits).
- **Angel's List** (rental & handyman listings; paid featured slots).
- Search across all cousins; category directory.

4. Cousin subdomains (initial 3)

- `cooking.yooahoo.guru` (Chef Yeti) – classes, recipes, equipment affiliates.
- `painting.yooahoo.guru` (Painter Yeti) – how-tos, tool affiliates, booking.
- `plumbing.yooahoo.guru` (Uncle Pipewrench) – fixes, part affiliates, booking.
- Each: 10–20 cornerstone articles + local booking form, inter-link to hub/other cousins.

5. Monetization

- Ads (AdSense/Ezoic) once traffic qualifies.
- Affiliates (Amazon/Tool-specific programs, course platforms).
- Premium: featured listings on Angel's List; subscription “Guru Pass” (no ads, discounts, priority).
- Lead capture: email + SMS; drip for upsells (classes, kits, services).

6. Automation & SEO

- Content engine: keyword research → AI draft → human polish; scheduled posting.
- Internal linking between cousins + hub; XML sitemaps per subdomain; shared analytics.
- Local SEO: per-city pages for handyman categories in Angel's List.

7. Ops & Analytics

- Admin console: approvals, moderation, payouts, metrics.
- Observability: uptime, error logs, search performance, conversion funnels.
- Backups and versioned media storage.

Minimal tech blueprint (keep it simple)

- **Frontend/Server:** Next.js (app router). Middleware resolves subdomain → loads cousin config (name, mascot, palette, menu, default category).
- **DB:** Postgres (users, profiles, listings, bookings, messages, reviews, posts, categories, subdomain_config).
- **Auth:** NextAuth (email + Google).
- **Payments:** Stripe (one-time featured listings + subscriptions).
- **Storage/CDN:** Cloudflare R2 or S3 + Cloudflare CDN.
- **Email/SMS:** Resend + Twilio (or alternatives).
- **Hosting:** Vercel/Cloudflare/Railway; map apex + wildcard.

Example URL map

- **Hub:** <https://yoohoo.guru/>, [/angels-list](https://yoohoo.guru/angels-list), [/skill-swap](https://yoohoo.guru/skill-swap), [/search](https://yoohoo.guru/search)
- **Cousins:**
 - <https://cooking.yoohoo.guru/classes/knife-skills>
 - <https://painting.yoohoo.guru/guides/exterior-prep>
 - <https://plumbing.yoohoo.guru/fixes/leaky-trap>
- **City landing (Angel's List):** <https://yoohoo.guru/angels-list/johnson-city-tn/handyman>

Data model sketch (tables)

- `users(id, email, role)`
- `profiles(user_id, display_name, bio, location)`
- `subdomains(id, slug, title, mascot_name, palette, status)`
- `posts(id, subdomain_id, title, slug, body, tags, published_at)`
- `listings(id, owner_id, subdomain_id|null, type[service|rental], title, desc, price, city, featured, status)`
- `messages(id, from_id, to_id, listing_id, body, created_at)`
- `bookings(id, listing_id, buyer_id, starts_at, ends_at, price, status)`
- `reviews(id, subject_user_id, author_user_id, rating, text)`
- `subscriptions(id, user_id, stripe_customer_id, plan, status)`
- `affiliates(id, subdomain_id, merchant, link, widget_config)`

Launch in 10 steps

1. Wildcard DNS + wildcard SSL.
2. Next.js repo with subdomain middleware + Postgres schema above.
3. Auth + Profiles + CMS (posts) working on hub.
4. Angel's List MVP: create/list/browse, featured flag (paid), moderation.
5. Skill-Swap MVP: post request/offer, match by tags, basic inbox.
6. Cousin theming: load mascot/assets from `subdomains` table; ship first 3 cousins.
7. Seed each cousin with 10 polished articles + 3 affiliate widgets.
8. Apply for AdSense; wire Stripe for featured listings + Guru Pass.
9. Analytics (search terms, CTR, conversions) + sitemap per subdomain.
10. Add 1–2 new cousins per month; expand city pages for Angel's List.

If you want, I can turn this into a **starter Next.js repo structure** (with subdomain middleware, Postgres schema, basic pages for hub + 3 cousins) so you can plug in assets and go.

You said:

So yes, let's do this. My domain is through porkbun - should i get a static email through them or just use domain aliases from intellme.com. REMEMBER yoohoo.guru is being built as a community skill swap and small job kiosk via angel's list. Maybe subdomains coach.yoohoo.guru and angel.yoohoo.guru?

ChatGPT said:

Perfect — let's line this up with what you're trying to build. You've already got **yoohoo.guru** on Porkbun, so the next questions are about email + subdomain strategy.

Email Choices

Option 1: Use Porkbun's Email Hosting

- **Pros:**
 - Quick, integrated with the domain.
 - Cheap (\$2–3/month per mailbox).
 - Works well if you just need static branded mailboxes like `support@yoohoo.guru`, `angel@yoohoo.guru`.
- **Cons:**
 - Limited features compared to full business suites (Google/Microsoft).
 - Less polished for team collaboration.

Option 2: Use Aliases on intellmeai.com (your business tenant)

- **Pros:**
 - Everything stays under one Microsoft 365 / Google Workspace tenant.
 - One inbox can send/receive as multiple branded aliases (e.g. angel@yooahoo.guru forwards into your main brandon@intellmeai.com).
 - Centralized billing, admin, and compliance.
- **Cons:**
 - Feels less “separate” — might confuse if YooHoo Guru is meant to be a standalone public-facing community brand.
 - Some users notice when a reply header comes from another root domain.

Recommendation:

- If YooHoo Guru is a **serious standalone brand**, I’d go Porkbun email hosting (or hook it to Google Workspace with its own account).
- If it’s more of a **satellite project under InTellMe**, then domain aliases from intellmeai.com are cheaper and easier.

Subdomain Strategy for YooHoo Guru

You’re spot on with using subdomains as “cousins” in the Yeti family. Based on your idea:

- **Hub:** yooahoo.guru
 - Landing, Skill Swap engine, Angel’s List
- **Angel’s Section:** angel.yooahoo.guru
 - Rentals, handyman listings, job kiosk (could mirror Angel’s List but feel like “her corner”).
- **Coach Yeti:** coach.yooahoo.guru
 - Guidance, mentorship, life coaching, maybe courses/workshops.
- **Expansion Cousins:**
 - cooking.yooahoo.guru → Chef Yeti
 - painting.yooahoo.guru → Painter Yeti
 - plumbing.yooahoo.guru → Uncle Pipewrench
 - etc.

Key Technical Setup

1. **Wildcard DNS:**
In Porkbun, set *.yooahoo.guru → your hosting provider.

- That way, every new cousin subdomain is live instantly.
 - You can control which ones resolve through your app logic.
 - 2. **Wildcard SSL Certificate:**
So all subdomains are HTTPS-secure out of the gate.
 - Cloudflare or Let's Encrypt will handle this free.
 - 3. **App Routing:**
Your platform checks the subdomain part (`req.subdomains[0]`) and loads the right cousin config (mascot, theme, content, monetization).
 - Shared database (users, profiles, payments).
 - Separate branding for each cousin.
-

Branding & Roleplay Layer

- **Main Yeti (YooHoo):** Handyman/skill swap guru.
- **Angel:** Manages rentals & community listings.
- **Coach Yeti:** Motivator, advice, tutorials.
- **Other cousins:** Each skill niche.

This way, the brand **feels fun and character-driven**, while still functioning like a serious community service.

👉 Do you want me to draft the **exact Porkbun DNS + email setup steps** (like which records to create, MX configs, alias forwarding rules), so you can have both branded mailboxes and wildcard subdomains ready in one shot?

You said:

Coach yeti would be the skill swapping platform for someone to be able to have the structured way to teach their skills and vice versa from those wanting to learn.

ChatGPT said:

Love it. **Coach Yeti** = the structured skill-swap engine; **Angel's List** = small jobs & rentals. Here's a tight, build-ready plan that you can hand straight to a dev (or Copilot) and start wiring up.

What Coach Yeti does (in one line)

A subdomain (`coach.yoohoo.guru`) where people **teach or learn** skills via swaps, credits, or cash—matched by goals, skill levels, availability, and location—using one shared account across the whole YooHoo universe.

Product slices (MVP → V1)

A) Core user flows

1. Profile & Skills

- Roles: Learner, Coach (most users are both).
- Skill entries: {name, level[1-5], teach|learn, hourly_value, tags, city, remote_ok}.
- Portfolio: links, files, short intro video (optional).

2. Offer / Ask

- **Offer** = “I can teach X for Y credits/hr or \$Z/hr.”
- **Ask** = “I want to learn X; I’ll pay with credits or swap with Y.”
- Each has duration, format (1:1, group), location/online, and availability window.

3. Matching

- Engine scores pairs by: skill match, level gap, schedule overlap, distance (if in-person), rating, and price/credit fit.
- Return a ranked list; user can invite/DM.

4. Booking & Escrow

- Scheduler (calendar sync optional).
- Escrow **Credits** (native) or **Stripe** (cash).
- Session happens → completion confirmation → escrow releases.

5. Reputation

- After each session: 1–5 stars + tags (“clear”, “patient”, “prepared”), optional testimonial.
- Reputation weights future match rank.

6. Disputes & Safety

- 24–48h resolution window on escrow.
 - Report & block; simple TOS + community standards.
-

B) Currency options (pick both; users choose per session)

- **Credits (default):** 1 credit \approx 1 hour baseline. Coach can set multipliers (e.g., “Photography: 1.5 credits/hr”).
 - Earn by teaching or by completing community jobs on **Angel’s List**.
 - **Cash:** Stripe (one-off or subscription “Guru Pass”).
-

Data model (starter)

users(id, email, name, avatar, city, is_verified, created_at)
profiles(user_id, bio, tagline, hourly_default, social_links_json)
skills(id, name, canonical_slug, category)
user_skills(user_id, skill_id, role[teach|learn|both], level, hourly_value, tags, remote_ok, city)
offers(id, user_id, skill_id, format[1v1|group], price_type[credit|cash|either], rate_numeric, min_block_mins, location, availability_json, status)
asks(id, user_id, skill_id, target_level, budget_type[credit|cash], max_rate, availability_json, city, status)
matches(id, offer_id, ask_id, score, created_at, accepted_by_learner, accepted_by_coach)
bookings(id, match_id, starts_at, ends_at, duration_mins, mode[online|inperson], status[pending|escrowed|complete|disputed|refunded])
wallets(user_id, credit_balance, pending_escrow)
transactions(id, user_id, type[earn|spend|refund|bonus|cash_payout], amount, ref_table, ref_id, created_at)
reviews(id, booking_id, rater_id, ratee_id, stars, tags, text, created_at)
disputes(id, booking_id, opened_by, reason, details, status)

Matching formula (simple & effective)

For a candidate pair (offer O, ask A):

```
score =  
  0.40 * skill_match           # exact skill + useful sub-skill synonymy  
+ 0.20 * schedule_overlap     # % overlap in next 2 weeks  
+ 0.10 * distance_factor      # 1.0 online; else f(mi)  
+ 0.15 * rep_factor           # normalized (avg_stars * log(1+reviews))
```

```
+ 0.10 * price_fit           # credits/cash within tolerance
+ 0.05 * recency_boost       # newer coaches get a temporary nudge
```

- **skill_match**: 1.0 exact; 0.8 close synonym; 0.5 category proximity.
- **distance_factor**: e.g., $1 / (1 + \text{miles}/10)$ for local; 1.0 if remote_ok.
- **price_fit**: 1.0 if within learner budget or credit balance; declines linearly after.

Return top N results; allow user filters (remote only, price cap, rating ≥ 4.5 , etc.).

Booking + escrow lifecycle

1. Learner taps **Book** → choose slot → **escrow** (credits or Stripe).
 2. Session happens (video link or address auto-shared).
 3. Either party marks complete → the other confirms; if both confirm or 24h passes with no dispute → **release**.
 4. Auto-request review; apply no-show logic (partial refund rules configurable).
-

Pricing & monetization (you keep it simple)

- **Free tier**: unlimited browsing, limited active asks/offers (e.g., 3).
 - **Guru Pass (\$/mo)**: unlimited asks/offers, lower fees, badge, priority ranking.
 - **Platform fee**: 8–12% on cash sessions, **0% on credit swaps** (community-friendly).
 - **Angel's List cross-earn**: complete a paid job → earn bonus credits usable in Coach Yeti.
-

Admin & ops

- Approvals queue for flagged listings.
 - Category controls (min/ max rate, allowed tags).
 - Dispute dashboard with one-click refund/partial split.
 - Metrics: conversion to booking, session completion rate, refund rate, supply/demand by skill & city.
-

Tech implementation (concrete)

- **Subdomain:** `coach.yoohoo.guru` served by the same app; subdomain middleware for branding.
 - **Stack:** Next.js (App Router) + Postgres + Prisma + NextAuth + Stripe.
 - **Video:** Start simple—“bring your own Zoom/Meet link”; V1 later use Daily.co/Twilio.
 - **Search/Match:** SQL + trigram/pgvector (optional) for synonymy; cron to refresh scores nightly.
 - **Credits:** Simple integer in `wallets.credit_balance`; guarded transactions in a single DB TX.
 - **Email/SMS:** Resend + Twilio for reminders and “session starting now” pings.
 - **Moderation:** Word filter + image size/type checks; abuse report queue.
-

UX copy (fast start)

- **CTA (Learner):** “Tell Coach Yeti what you want to learn.”
 - **CTA (Coach):** “List a skill and earn credits or cash.”
 - **Empty state:** “No perfect match yet—Coach Yeti is scouting. Try widening your time window.”
-

Launch checklist (14 steps)

1. Create `coach.yoohoo.guru` DNS and ensure wildcard is active.
 2. Add subdomain theme config (`/config/subdomains/coach.json`).
 3. Build Profile → Skills → Offer/Ask forms.
 4. Seed canonical skill list (top 200).
 5. Implement matching service & endpoint `/api/match`.
 6. Wallet + credit transactions (earn/spend/escrow).
 7. Stripe onboarding & test payouts.
 8. Booking calendar + email/SMS reminders.
 9. Session completion → escrow release → review prompt.
 10. Dispute flow & admin actions.
 11. Public browse pages (SEO) for skills & cities.
 12. Angel’s List bridge: award credits on completed Angel jobs.
 13. Basic analytics: signups, publishes, matches, bookings, completion.
 14. Content: 10 guide posts + “How swapping works” + safety page.
-

If you want, I can generate a **starter Next.js repo scaffold** (routes, Prisma schema above, subdomain theming, Stripe test mode, wallet/escrow boilerplate) that you can drop on Railway/Vercel and connect to your DNS.

ChatGPT can make mistakes. Check important info.