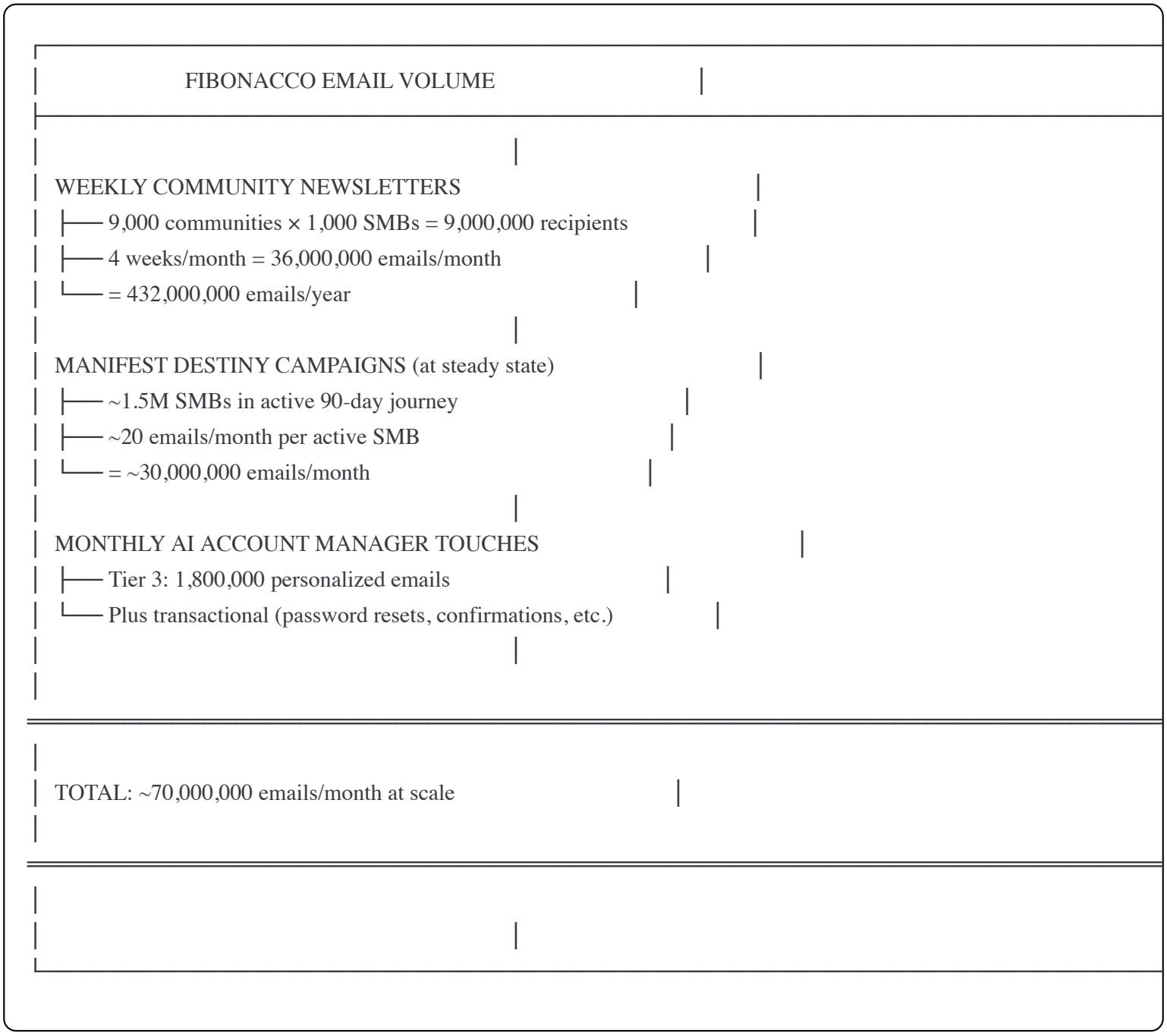
FIBONACCO EMAIL INFRASTRUCTURE ANALYSIS

AWS SES vs SendGrid vs Self-Hosted: The Complete Picture

YOUR SCALE (Reality Check)



COST COMPARISON AT SCALE

Monthly Cost for 70M Emails

MONTHLY COST: 70M EMAILS

OPTION 1: AWS SES

Pricing Tiers:
• First 62,000 emails: FREE (if sending from EC2)
• Up to 50M emails: $0.10 per 1,000 = $0.0001/email
• 50M-100M emails: Volume discount available (~$0.05-0.08/1,000)

70M emails × $0.08/1,000 = $5,600/month
+ Dedicated IPs (10 × $25) = $250/month
+ Data transfer: ~$500/month

TOTAL: ~$6,350/month

✅ Pros: Cheapest, integrates with AWS, highly reliable
❌ Cons: Bare-bones features, need to build everything else

OPTION 2: SENDGRID

At 70M emails/month, you need custom "Premier" pricing

Estimated: $0.40-0.80 per 1,000 emails at volume
70M × $0.50/1,000 = $35,000/month

+ Dedicated IPs: Included in Premier
+ Marketing contacts storage: Additional cost

TOTAL: ~$35,000-50,000/month

✅ Pros: Better UI, analytics, templates, deliverability tools
❌ Cons: 5-8x more expensive than AWS SES

OPTION 3: SELF-HOSTED SMTP (Like Patch)

Infrastructure:
• Mail servers (Postfix + Dovecot or Mailcow): $0/software
• EC2 instances for sending: ~$500-1,000/month
• Dedicated IPs (20-50 IPs for volume): ~$100/month
• DevOps/engineer time: 0.5-1 FTE = $5,000-10,000/month

Per-email cost: ~$0.00001 (essentially bandwidth only)

TOTAL: ~$6,000-12,000/month (mostly labor)

✅ Pros: No third-party dependency, lowest per-email cost
❌ Cons: MASSIVE reputation risk, requires expertise, blacklisting

OPTION 4: HYBRID (Self-Hosted Software + AWS SES Delivery)

Software (like listmonk, Sendy, or custom):
• Open-source or one-time license
• Handles: Lists, templates, tracking, analytics
• Sends through: AWS SES API

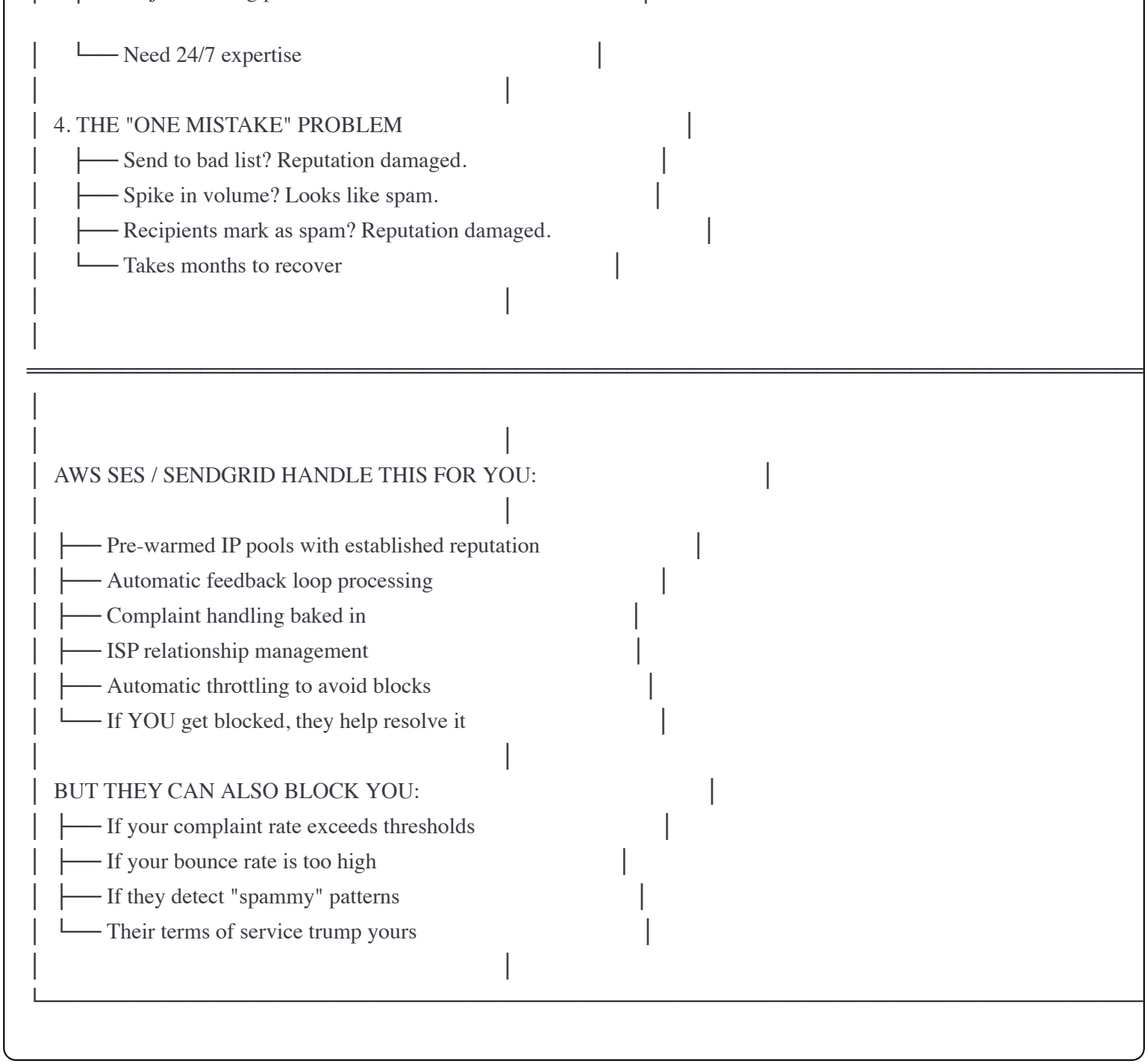
Cost:
• AWS SES for delivery: ~$6,350/month (same as Option 1)
• EC2 for software: ~$200/month
• Development/maintenance: Built into your platform

TOTAL: ~$6,550/month

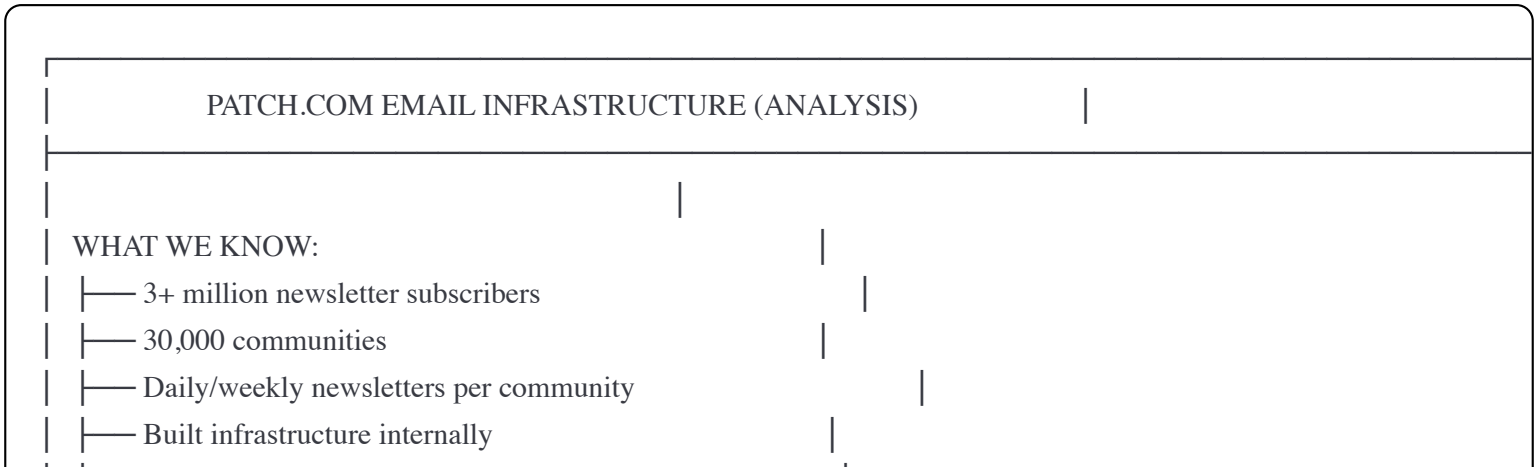
✅ Pros: AWS handles reputation, you control software
✅ Pros: Can switch delivery provider if AWS restricts you
✅ Pros: No vendor lock-in on subscriber data
❌ Cons: Need to build/maintain the management layer

THE REPUTATION PROBLEM (Why Self-Hosted SMTP is Risky)

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                    EMAIL REPUTATION: THE REAL CHALLENGE                      │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  WHY EMAIL REPUTATION MATTERS:                                               │
│                                                                              │
│  When you send email, receiving servers (Gmail, Outlook, Yahoo) check:       │
│  ├── Is this IP address known for spam?                                      │
│  ├── Is this domain authenticated (SPF, DKIM, DMARC)?                        │
│  ├── What's the complaint rate from recipients?                              │
│  ├── What's the bounce rate?                                                 │
│  └── Is this sender on any blacklists?                                       │
│                                                                              │
╞══════════════════════════════════════════════════════════════════════════════╡
│                                                                              │
│  SELF-HOSTED SMTP RISKS:                                                     │
│                                                                              │
│  1. IP REPUTATION STARTS AT ZERO                                             │
│     ├── New IPs have no history                                              │
│     ├── Must "warm up" slowly (weeks to months)                              │
│     ├── Gmail/Microsoft may block or throttle initially                      │
│     └── One bad day can destroy months of warming                            │
│                                                                              │
│  2. BLACKLISTING IS EASY                                                     │
│     ├── Too many bounces? Blacklisted.                                       │
│     ├── Too many spam complaints? Blacklisted.                               │
│     ├── Sending too fast? Throttled or blocked.                              │
│     ├── Shared IP range has a spammer? You're affected.                      │
│     └── Getting off blacklists = manual appeals + waiting                    │
│                                                                              │
│  3. CONSTANT MONITORING REQUIRED                                             │
│     ├── Check multiple blacklists daily                                      │
│     ├── Monitor delivery rates by ISP                                        │
│     ├── Watch feedback loops (complaint reports)                             │
│     ├── Adjust sending patterns in real-time                                 │
```

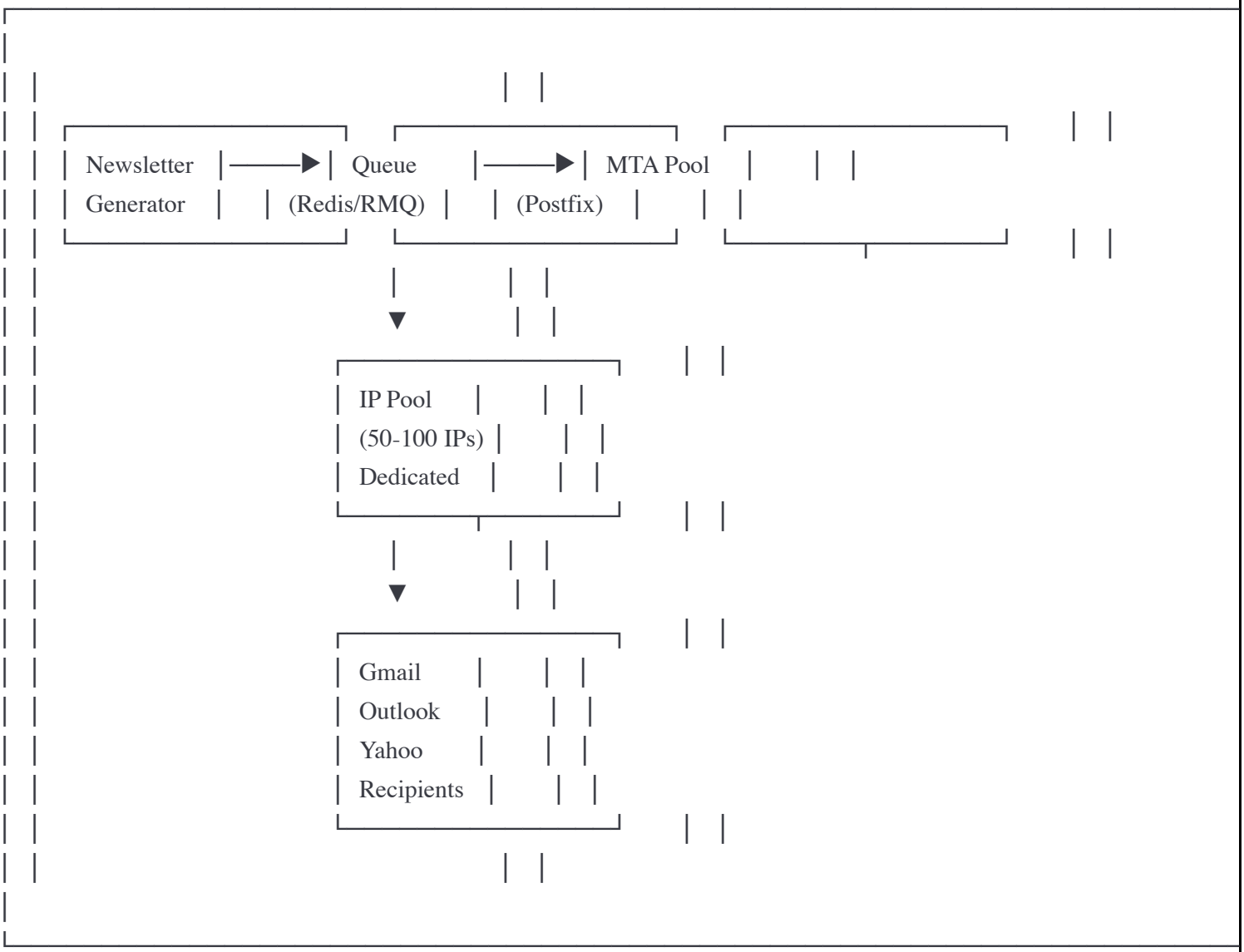


HOW PATCH.COM LIKELY DOES IT



- Been operating for 10+ years
- Likely sending 10-50M emails/month

LIKELY ARCHITECTURE:



WHY PATCH CAN DO THIS:

- 10+ year track record = established IP reputation
- Legitimate local news = low complaint rates
- Opt-in subscribers = clean lists
- Engineering team to maintain infrastructure
- Gradual growth allowed reputation building over time
- Content is genuinely valuable (not spam)

WHY YOU CAN'T JUST COPY THIS (YET):

- No established reputation
- Starting with cold lists (SMBs who didn't opt-in)
- Rapid scale-up = looks like spam to ISPs
- Need to prove legitimacy first

└── Risk of immediate blacklisting

RESTRICTION RISK COMPARISON

WHO MIGHT BLOCK/RESTRICT YOU?

AWS SES RESTRICTIONS:

TRIGGERS:
- Bounce rate > 5% (warning), > 10% (suspension)
- Complaint rate > 0.1% (warning), > 0.5% (suspension)
- Sending to purchased/scraped lists
- Sudden volume spikes without warmup

WHAT HAPPENS:
- Account put in "probation"
- Sending paused until you fix issues
- Must submit remediation plan
- Can take days/weeks to resolve

RISK LEVEL FOR FIBONACCO: MEDIUM
- Your lists are SMBs you're reaching out to (not opt-in)
- Initial bounce rates may be high (stale data)
- Need careful warmup strategy

SENDGRID RESTRICTIONS:

TRIGGERS:
- Same as AWS SES (bounce/complaint thresholds)
- Content filtering (looks like spam?)
- Purchased list detection

- Purchased list detection
- Excessive unsubscribes

WHAT HAPPENS:
- Account suspension (often with little warning)
- Must appeal through support
- Recovery can take weeks
- May require switching plans or getting audited

RISK LEVEL FOR FIBONACCO: MEDIUM-HIGH
- More strict than AWS SES
- Twilio (parent company) has tight policies
- Less transparency in decisions

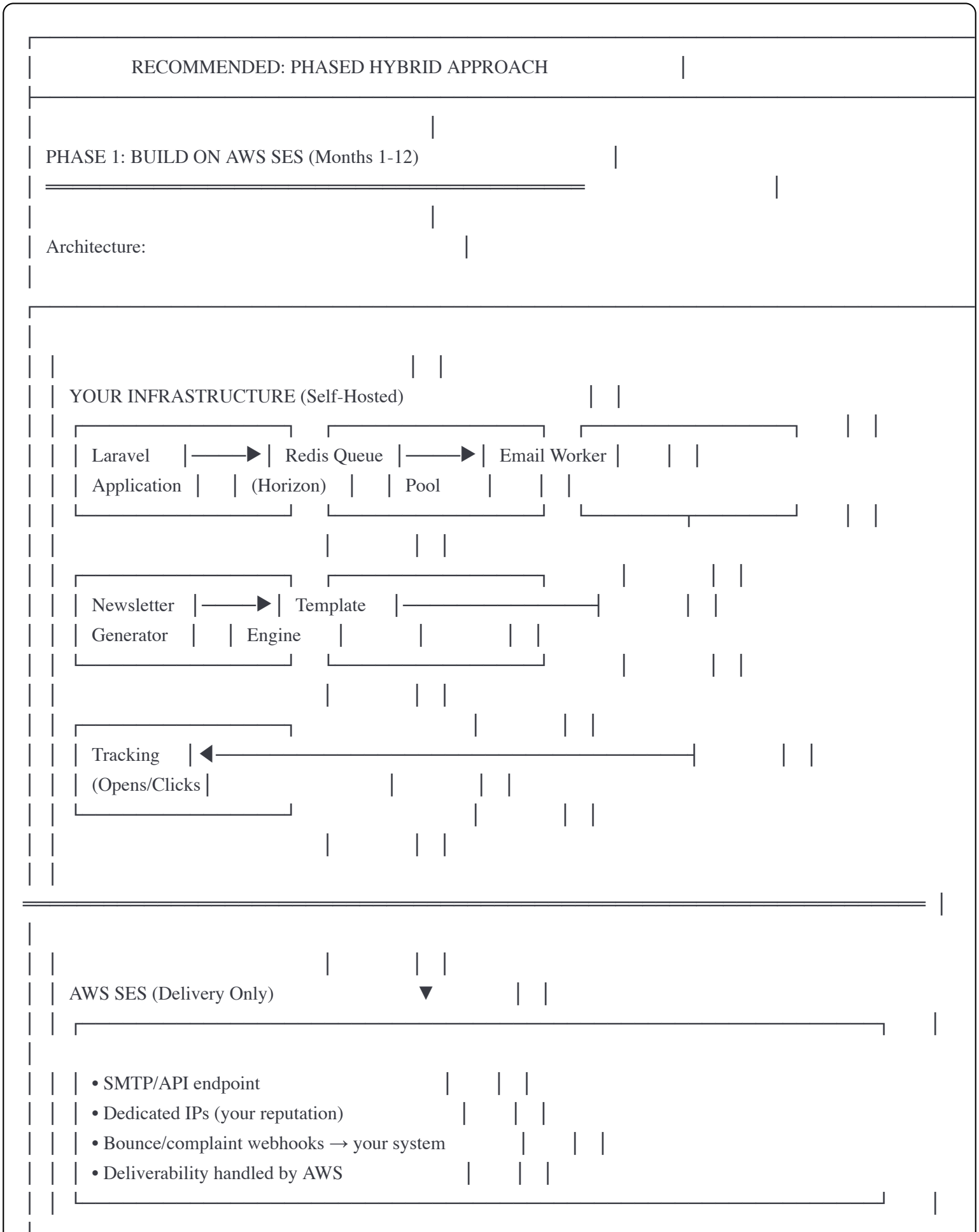
SELF-HOSTED RESTRICTIONS:

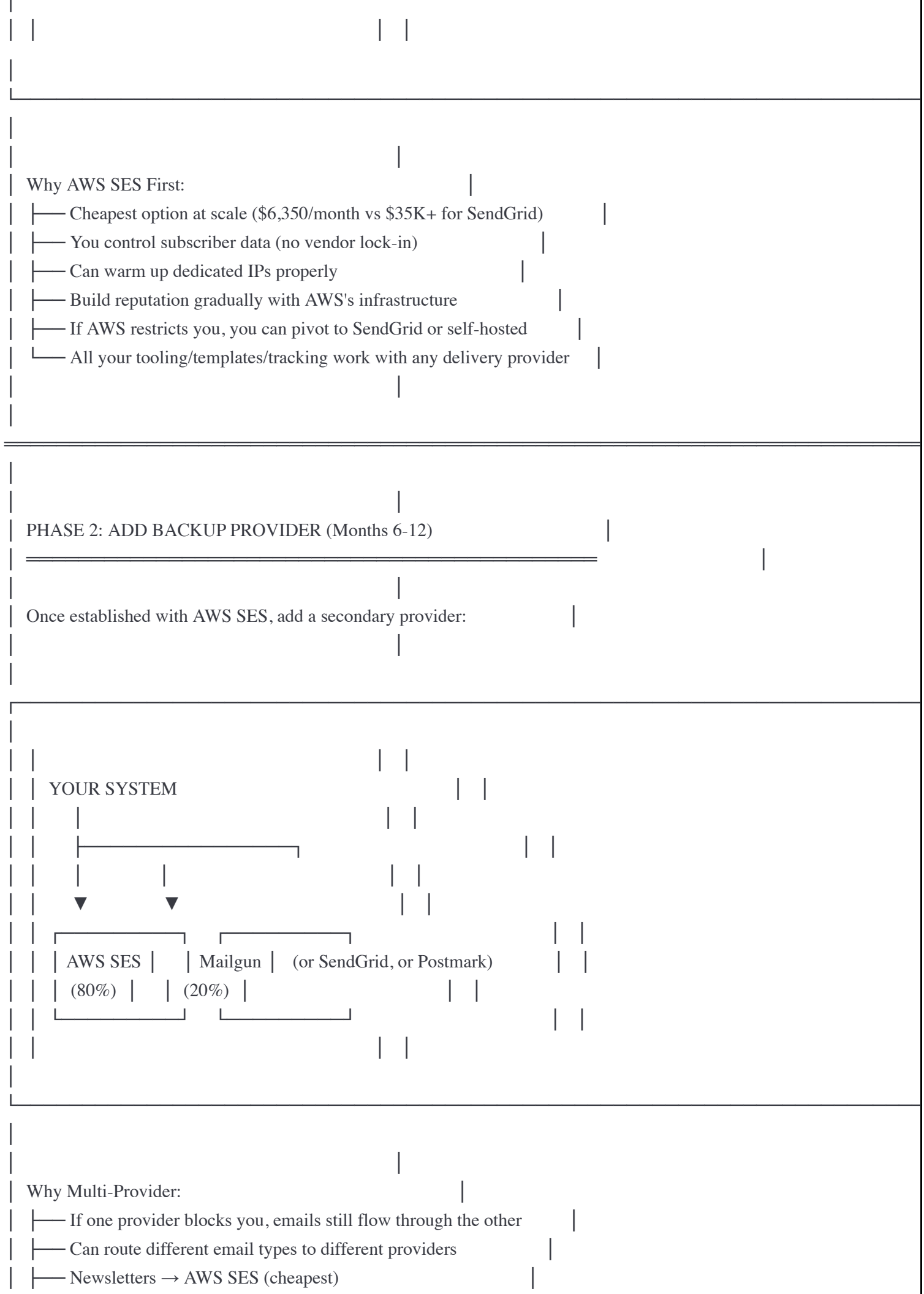
TRIGGERS:
- ISP blocks (Gmail, Microsoft, Yahoo decide individually)
- Blacklist inclusion (Spamhaus, Barracuda, etc.)
- Spam reports from recipients
- Failed authentication (SPF, DKIM, DMARC)

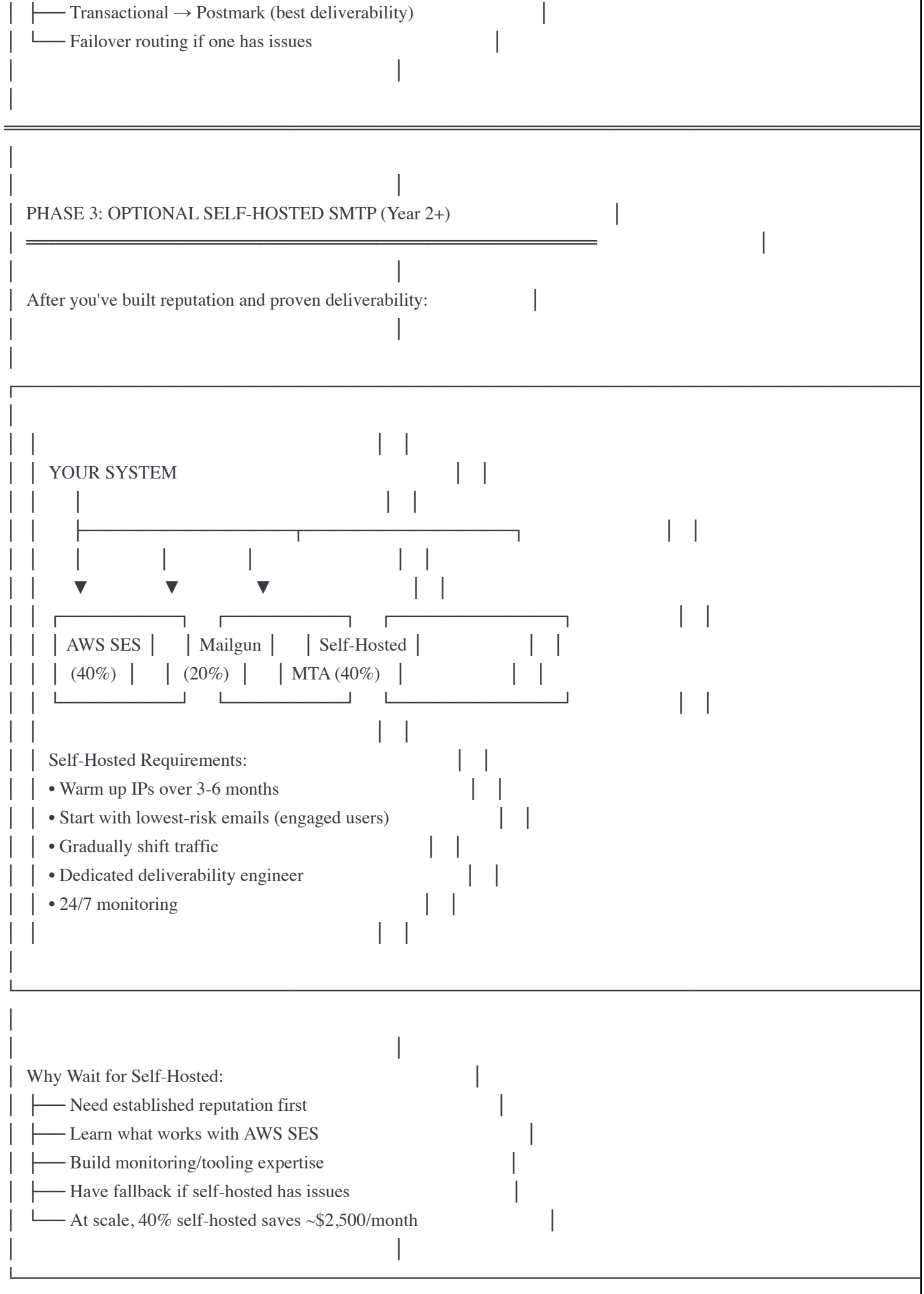
WHAT HAPPENS:
- Emails silently dropped or sent to spam
- No notification - you discover by low open rates
- Must appeal to each ISP individually
- Blacklist removal is manual and slow

RISK LEVEL FOR FIBONACCO: VERY HIGH
- No established reputation
- Cold outreach = high spam potential
- Scale-up would trigger immediate blocks
- Recovery could take months

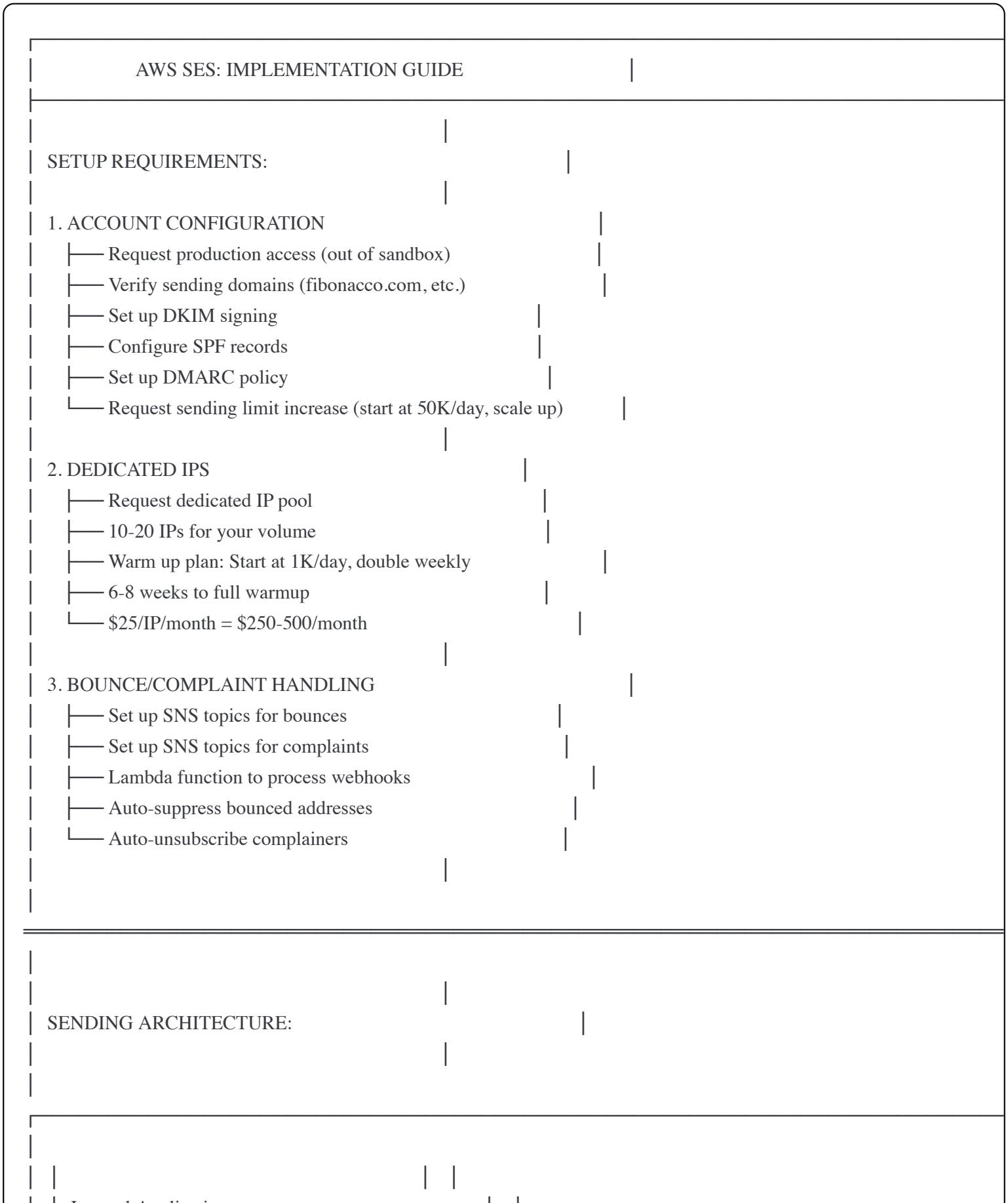
RECOMMENDED APPROACH: PHASED HYBRID

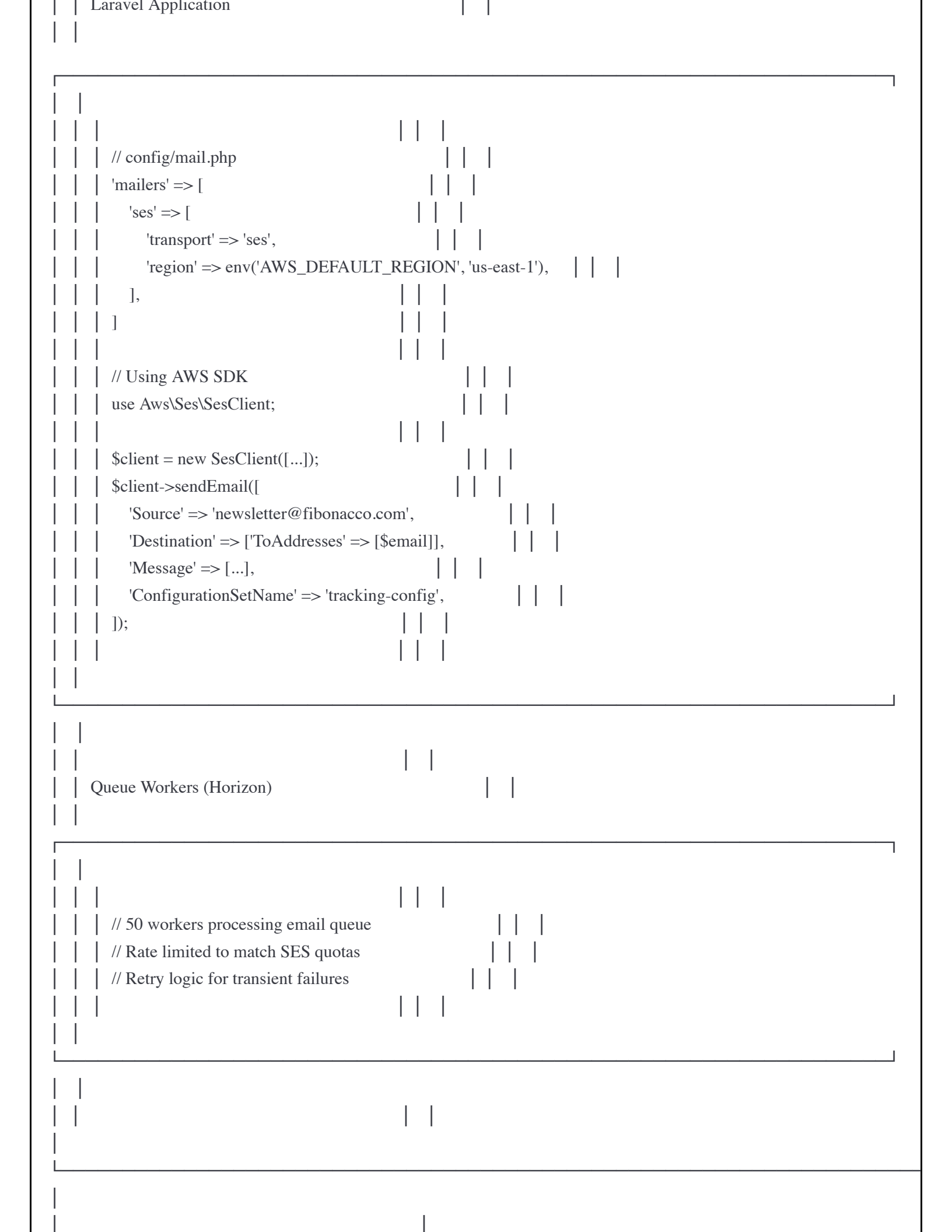



Why AWS SES First:
├── Cheapest option at scale ($6,350/month vs $35K+ for SendGrid)
├── You control subscriber data (no vendor lock-in)
├── Can warm up dedicated IPs properly
├── Build reputation gradually with AWS's infrastructure
├── If AWS restricts you, you can pivot to SendGrid or self-hosted
└── All your tooling/templates/tracking work with any delivery provider
PHASE 2: ADD BACKUP PROVIDER (Months 6-12)
Once established with AWS SES, add a secondary provider:
YOUR SYSTEM
AWS SES
(80%)
Mailgun
(20%)
(or SendGrid, or Postmark)
Why Multi-Provider:
├── If one provider blocks you, emails still flow through the other
├── Can route different email types to different providers
├── Newsletters → AWS SES (cheapest)


├── Transactional → Postmark (best deliverability)
└── Failover routing if one has issues
PHASE 3: OPTIONAL SELF-HOSTED SMTP (Year 2+)
After you've built reputation and proven deliverability:
YOUR SYSTEM
AWS SES
(40%)
Mailgun
(20%)
Self-Hosted
MTA (40%)
Self-Hosted Requirements:
• Warm up IPs over 3-6 months
• Start with lowest-risk emails (engaged users)
• Gradually shift traffic
• Dedicated deliverability engineer
• 24/7 monitoring
Why Wait for Self-Hosted:
├── Need established reputation first
├── Learn what works with AWS SES
├── Build monitoring/tooling expertise
├── Have fallback if self-hosted has issues
└── At scale, 40% self-hosted saves ~$2,500/month

AWS SES IMPLEMENTATION DETAILS




Laravel Application
// config/mail.php
'mailers' => [
'ses' => [
'transport' => 'ses',
'region' => env('AWS_DEFAULT_REGION', 'us-east-1'),
],
]
// Using AWS SDK
use Aws\Ses\SesClient;
$client = new SesClient([...]);
$client->sendEmail([
'Source' => 'newsletter@fibonacco.com',
'Destination' => ['ToAddresses' => [$email]],
'Message' => [...],
'ConfigurationSetName' => 'tracking-config',
]);
Queue Workers (Horizon)
// 50 workers processing email queue
// Rate limited to match SES quotas
// Retry logic for transient failures

COST BREAKDOWN (70M emails/month):

- Email sending: 70M × $0.0001 = $7,000/month
  (With volume discount: ~$5,600/month)
- Dedicated IPs (20): 20 × $25 = $500/month
- SNS notifications: ~$100/month
- CloudWatch monitoring: ~$50/month
- Data transfer: ~$200/month

TOTAL: ~$6,450/month

vs SendGrid: ~$35,000-50,000/month
SAVINGS: $28,550-43,550/month = $342K-522K/year

SELF-HOSTED SOFTWARE OPTIONS (For Management Layer)

SELF-HOSTED EMAIL MANAGEMENT SOFTWARE

OPTION A: LISTMONK (Recommended for your scale)

- Open source, free
- Written in Go (fast, low resource)
- Handles millions of subscribers
- PostgreSQL backend (matches your stack)
- Multi-SMTP support (can route to AWS SES, Mailgun, etc.)
- Template engine with Liquid-like syntax
- Built-in analytics
- REST API for integration

Proven: "7+ million emails with CPU fraction of a core, 57MB RAM"

Limitation: You'd need to build custom integration with your Laravel app, or run it as a separate service

OPTION B: SENDY ($69 one-time license)

- PHP-based (matches Laravel stack)
- Built specifically for AWS SES
- Handles bounce/complaint processing automatically
- Simple UI for campaigns
- One-time purchase, no recurring fees

Limitation: Designed for humans using UI, not programmatic sending at your scale

OPTION C: BUILD CUSTOM (Recommended)

Since you're building Fibonacco in Laravel already:

- Build email system into your existing platform
- Use Laravel queues + Horizon for job processing
- AWS SES SDK for sending
- Your own tracking pixel/link redirector
- Database for engagement tracking
- Webhook receiver for bounces/complaints

Advantages:

- Full integration with your CRM/data
- Customized for your exact needs
- No external dependencies

• Can switch email providers easily

RISK MITIGATION: AVOIDING BLOCKS

HOW TO AVOID GETTING BLOCKED

1. LIST HYGIENE (Critical for cold outreach)

BEFORE sending to any list:
- Email verification service (ZeroBounce, NeverBounce)
- Remove invalid/catch-all addresses
- Remove role addresses (info@, admin@, etc.)
- Remove disposable email domains
- Cost: ~$3-5 per 1,000 emails verified (one-time)

For 9M SMBs: ~$27,000-45,000 one-time verification cost
WORTH IT: Prevents bounces that destroy reputation

2. WARMUP STRATEGY

Week 1: 1,000 emails/day
Week 2: 2,500 emails/day
Week 3: 5,000 emails/day
Week 4: 10,000 emails/day

Week 4: 10,000 emails/day
Week 5: 25,000 emails/day
Week 6: 50,000 emails/day

Week 7: 100,000 emails/day
Week 8+: Scale as reputation allows

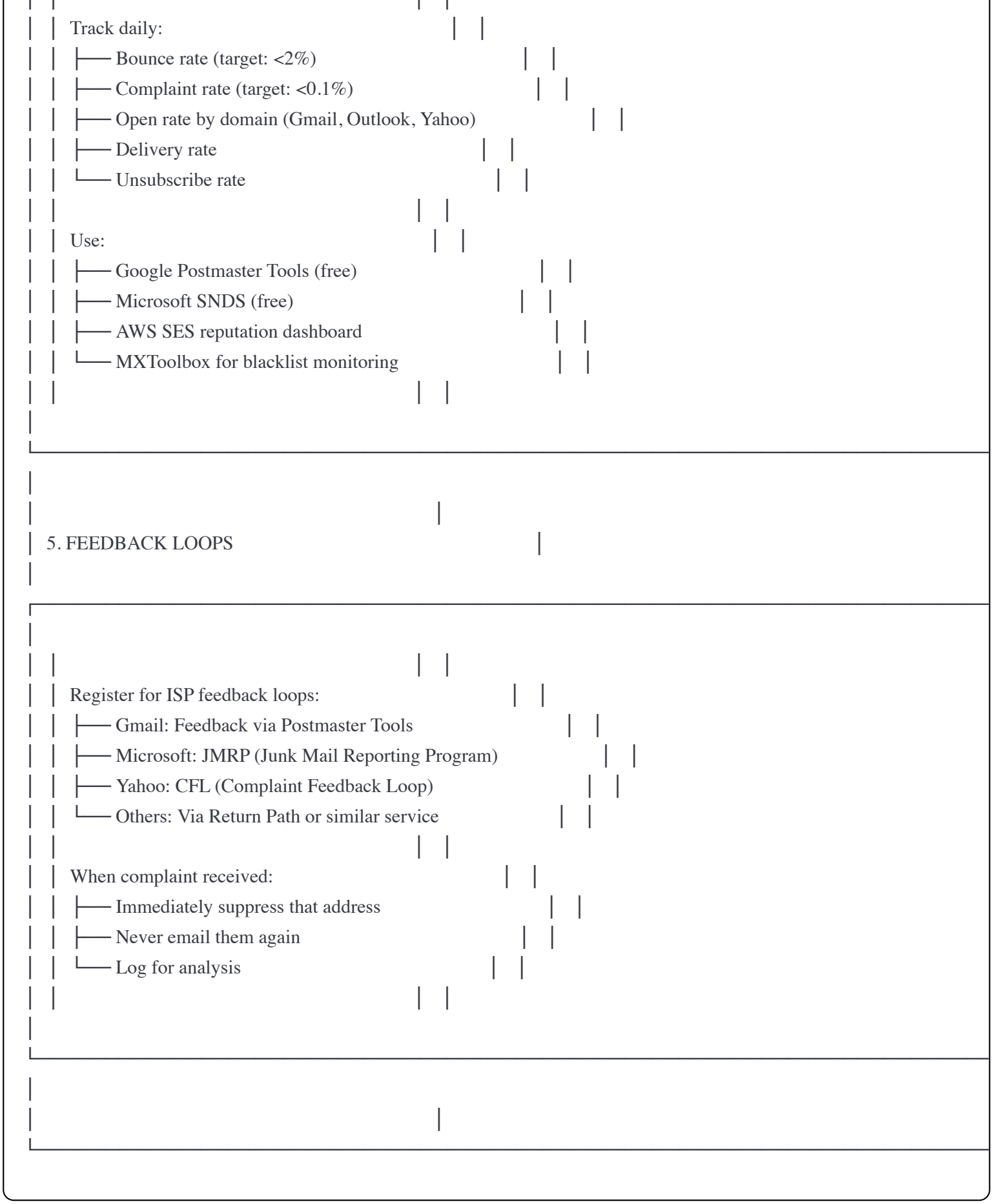
IMPORTANT: Start with your BEST contacts
- Most likely to engage
- Valid, verified emails
- Content they actually want

3. CONTENT BEST PRACTICES

DO:
- Clear unsubscribe link (prominent)
- Physical address in footer
- Recognizable "From" name
- Relevant, valuable content
- Personalization (their name, their community)
- Plain text version alongside HTML

DON'T:
- ALL CAPS SUBJECT LINES
- Excessive exclamation marks!!!
- Spam trigger words ("FREE", "ACT NOW", etc.)
- Too many images, not enough text
- URL shorteners (link to spam)
- Hidden unsubscribe links

4. MONITORING



FINAL RECOMMENDATION

FINAL RECOMMENDATION

FOR FIBONACCO'S SCALE AND GOALS:

✅ USE AWS SES FOR DELIVERY
- Cheapest at scale ($6,350/month vs $35K+)
- Handles reputation management
- Can use dedicated IPs (your reputation)
- Integrates with existing AWS infrastructure

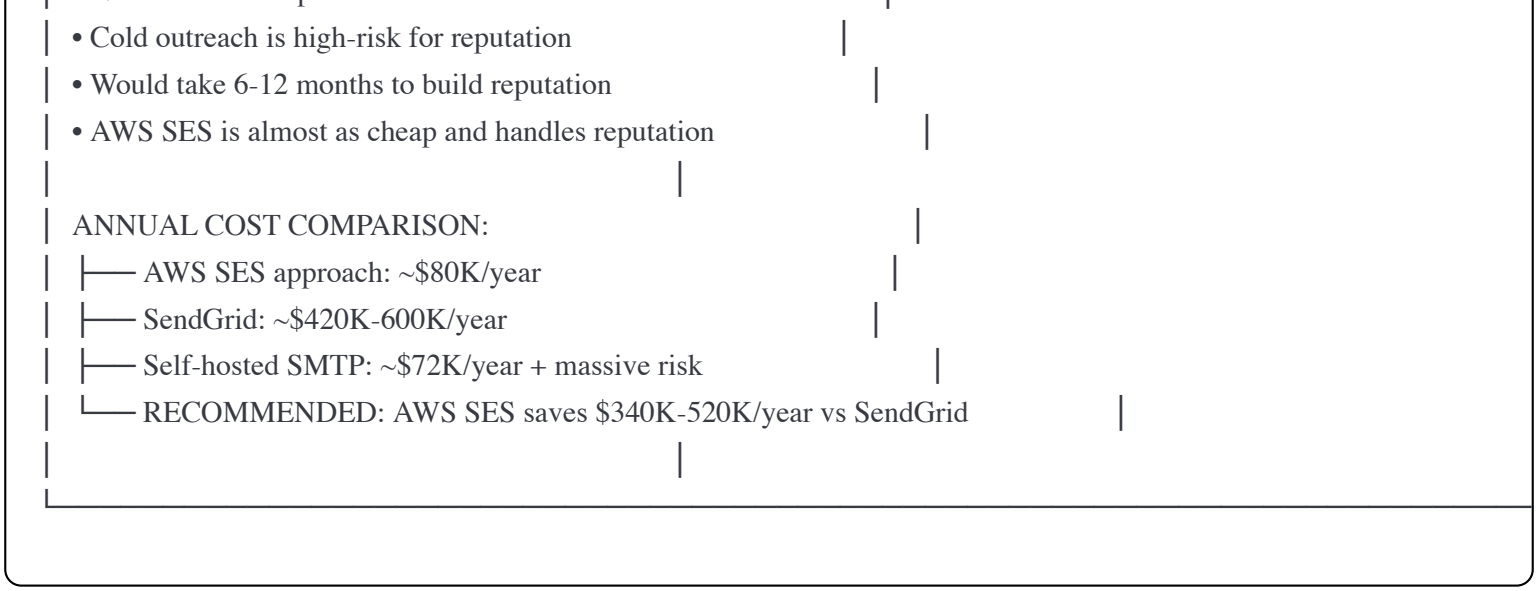
✅ BUILD YOUR OWN MANAGEMENT LAYER
- Integrate into Laravel platform
- Full control over data and logic
- Can switch providers if needed
- No vendor lock-in on subscriber data

✅ ADD SECONDARY PROVIDER (Month 6+)
- Mailgun or Postmark as backup
- Route 20% through secondary
- Failover capability

⏳ CONSIDER SELF-HOSTED SMTP (Year 2+)
- Only after established reputation
- Dedicated deliverability engineer
- Gradual migration
- Keep AWS SES as fallback

WHY NOT SENDGRID:
- 5-8x more expensive at your scale
- No additional deliverability benefit over AWS SES
- Same risk of being blocked
- More restrictive policies

WHY NOT SELF-HOSTED SMTP NOW:
- No established reputation = immediate blocks



NEXT STEPS

1. **Set up AWS SES account** - Request production access, verify domains
2. **Email verification** - Run your SMB list through ZeroBounce/NeverBounce ($30-45K)
3. **Request dedicated IPs** - 10-20 IPs, plan for warmup
4. **Build Laravel email module** - Queues, templates, tracking
5. **Implement bounce/complaint handling** - SNS webhooks, auto-suppression
6. **Start warmup** - 1,000 emails/day, best contacts first
7. **Monitor obsessively** - Postmaster Tools, SNDS, bounce rates
8. **Scale gradually** - Double volume weekly as metrics allow

---

Questions to decide:

1. Do you want to run email verification on your SMB list before launch? (Recommended)
2. Do you have existing relationships with any SMBs that could be "warm" first contacts?
3. What AWS region should we use? (us-east-1 typically best for email)
4. Do you want me to spec out the Laravel email module architecture?