OPPORTUNITY FINDER

Product Requirements Document

Version 1.5 - MVP (Complete)

10 January 2026

Status: DRAFT - Ready for Development

Owner: Mark

Target Delivery: 2-3 weeks from start

EXECUTIVE SUMMARY

Product Vision

Opportunity Finder is a systematic validation platform for micro-SaaS entrepreneurs delivered as a subscription SaaS product (£5/month). It eliminates guesswork by collecting real market demand signals from multiple sources, validating them against existing solutions and revenue data, and producing scored opportunities with clear build/no-build recommendations. Built from day one as a multi-tenant, payment-enabled platform with mobile-ready architecture.

Problem Statement

Indie developers and entrepreneurs waste months building products nobody wants because:

- They guess at problems rather than finding validated demand
- Manual research across multiple platforms is time-consuming and inconsistent
- No systematic way to validate if people will actually pay
- Opportunity assessment is subjective and emotionally driven

Solution

Automated system that monitors Reddit, Indie Hackers, ProductHunt, HackerNews, and other sources for pain points, validates them against existing paid solutions, scores opportunities 0-100 based on demand/revenue/competition/complexity, and provides clear recommendations.

Success Metrics

- **Primary:** 80% of opportunities scoring 60+ validate successfully with landing page test
- **Secondary:** User finds 1+ buildable opportunity per month
- **Tertiary:** Reduces validation time from 2-3 weeks to < 1 day

1. PRODUCT OVERVIEW

1.1 Target User

Primary:

- Solo indie developers with React/Python skills building first SaaS
- Former employees with 3-6 month runway seeking validated opportunities
- Willing to pay £5/month for systematic validation vs months of guessing
- Need to identify £1k+ MRR opportunities quickly before runway expires

1.2 Use Cases

1. **Weekly Opportunity Discovery:** User runs weekly scan, reviews top 10 opportunities, identifies 1-3 to investigate further
2. **Idea Validation:** User has idea, searches system to see if demand exists and validates score
3. **Market Research:** User explores specific niche (e.g., 'freelancer tools') to find gaps

1.3 Core Value Proposition

"Stop guessing. Build what people are actively looking for and willing to pay for."

2. TECHNICAL ARCHITECTURE

2.1 System Components

Backend (Python/Flask)

- **Authentication:** JWT-based auth with secure session management
- **Payment Processing:** Stripe integration for subscriptions (£5/month)
- **Data Collectors:** Modular scrapers for each source
- **Validation Engine:** Checks for existing solutions and revenue
- **Scoring Engine:** Calculates 0-100 score based on weighted criteria
- **REST API:** Exposes endpoints for frontend (with auth middleware)
- **Database:** PostgreSQL with multi-tenant architecture

Frontend (React)

- **Authentication:** Login/signup/password reset
- **Subscription Management:** Stripe checkout integration, billing portal
- **Dashboard:** Overview stats and top opportunities
- **Opportunity Cards:** Filterable, sortable grid view
- **Detail Modal:** Full opportunity breakdown with sources
- **Search & Filters:** Real-time filtering by score, keyword, time range
- **Theme Toggle:** Dark mode (default) with light mode option

UI Design Specification

CRITICAL: Use approved prototype design.

/mnt/c/Opportunity-Finder/docs/HTML/opportunity-finder.html

Dark Mode (Default):

- Background: Linear gradient from navy (#0f172a) to slate (#1e293b)
- Cards: Semi-transparent dark with subtle borders (rgba(148, 163, 184, 0.1))
- Text: White (#fff) for headings, light slate (#e2e8f0) for body
- Score badges: Color-coded (green 80+, blue 60-79, amber 40-59, red <40)

Light Mode (Optional Toggle):

- Background: White to light gray gradient
- Cards: White with subtle shadows
- Text: Dark gray for body, black for headings

2.2 Technology Stack

Component	Technology	Version
Backend	Python + Flask	3.11+ / 3.0+

Frontend	React + Vite	18+ / 5+
Database	PostgreSQL	15+
Authentication	Flask-JWT-Extended	4.6+
Payments	Stripe Python SDK	8.0+
Email	SendGrid / Mailgun	Latest
PDF Generation	ReportLab	4.0+
Reddit	PRAW	7.7+
Web Scraping	BeautifulSoup4	4.12+
API Calls	Requests	2.31+
Search	SerpAPI	Latest
ORM	SQLAlchemy	2.0+
CORS	Flask-CORS	4.0+
Password Hashing	bcrypt	4.1+

2.3 Deployment Architecture

- **Development:** Local (localhost:3000 frontend, localhost:5000 backend)
- **Production:** VPS deployment (DigitalOcean/Hetzner)
- **Frontend:** Nginx serving React build
- **Backend:** Gunicorn + Flask behind Nginx reverse proxy
- **Mobile Ready:** API-first design allows future React Native/Flutter app using same backend

3. DATA SOURCES

All data sources must be implemented. No optional sources in MVP.

3.1 Reddit

Purpose: Primary pain point discovery

Implementation: PRAW (Python Reddit API Wrapper)

Subreddits:

- r/Entrepreneur
- r/smallbusiness
- r/freelance
- r/SaaS
- r/startups
- r/indiehackers
- r/productivity

Search Keywords:

- "looking for a tool"
- "need software for"
- "wish there was"
- "hate that I have to"
- "tired of manually"
- "paying too much for"

Rate Limits: 60 requests/minute. Implement exponential backoff.

3.2 Indie Hackers

Purpose: Revenue validation + pain points

Implementation: BeautifulSoup4 web scraping

Data to Collect:

- Product names and descriptions
- MRR/ARR figures
- Founder interviews (pain points)
- Discussion threads

Update Frequency: Weekly scan of new products

3.3 ProductHunt

Purpose: New product launches + user feedback

Implementation: GraphQL API (free tier)

Data to Collect:

- Daily top launches
- Comment threads (pain points)
- Upvote counts (demand signal)

3.4 HackerNews

Purpose: Technical pain points from 'Ask HN' threads

Implementation: Algolia HN Search API (free)

Query Patterns:

- "Ask HN: What tool"
- "Ask HN: How do you"
- "Show HN" (for validation)

3.5 Google Search

Purpose: Competitor discovery + revenue validation

Implementation: SerpAPI (100 free searches/month)

Search Pattern: "[problem] software", "[problem] tool", "[problem] SaaS"

Use: Validate existing solutions, count competitors

3.6 Microns.io / Acquire.com

Purpose: Real MRR data from businesses for sale

Implementation: Web scraping (public listings)

Data: Product name, MRR, problem solved

4. SCORING ALGORITHM

Each opportunity receives a score 0-100 based on four weighted factors.

Factor	Weight	Scoring Criteria
Demand Frequency	25%	50+ mentions = 25pts, 30-49 = 15pts, 20-29 = 10pts
Revenue Proof	35%	£10k+ MRR = 35pts, £5-10k = 25pts, £2-5k = 15pts, £1-2k = 10pts
Competition	20%	1-2 competitors = 20pts, 3-5 = 15pts, 6-10 = 10pts, 10+ = 5pts
Build Complexity	20%	Simple CRUD = 20pts, Moderate = 15pts, Complex = 10pts

4.1 Validation Rules

CRITICAL: An opportunity cannot score above 40 unless it passes ALL of these:

- **Existing Paid Solution:** At least 1 competitor charging money
- **Revenue Proof:** Evidence of £1,000+ MRR in niche
- **Frequency:** Minimum 20 mentions across sources
- **B2B Problem:** Businesses will pay, not just consumers

4.2 Recommendation Logic

- **80-100:** "Build immediately" - All signals green
- **60-79:** "Validate with landing page first" - Strong but needs confirmation
- **40-59:** "High risk - need unique angle" - Proceed with caution
- **0-39:** "Reject - insufficient validation" - Do not build

5. FEATURE SPECIFICATIONS

5.1 Backend API Endpoints

POST /api/auth/register

Purpose: Create new user account

Body: email, password

Action: Sends verification email to user

GET /api/auth/verify-email/:token

Purpose: Verify email address via link in email

Action: Sets user.email_verified=true, redirects to login

POST /api/auth/login

Purpose: Authenticate and get JWT token

Returns: access_token, refresh_token (only if email verified)

POST /api/auth/forgot-password

Purpose: Request password reset

Body: email

Action: Sends password reset email with token (expires in 1 hour)

POST /api/auth/reset-password

Purpose: Reset password with token

Body: token, new_password

POST /api/payments/create-checkout

Purpose: Create Stripe checkout session for £5/month subscription

Returns: checkout_url

POST /api/payments/webhook

Purpose: Handle Stripe webhook events (payment success, subscription cancelled)

Updates: user.subscription_status in database

GET /api/opportunities

Purpose: Retrieve user's opportunities with filtering

Auth: Requires valid JWT token. Returns only current user's opportunities.

Query Parameters:

- min_score (int, default: 0)
- sort (string: 'score', 'revenue', 'mentions', default: 'score')
- search (string: keyword search)
- time_range (string: '3days', '1week', '1month', 'all', default: 'all')
- status (string: 'new', 'researching', 'building', 'rejected', default: all)
- limit (int, default: 100)

GET /api/opportunities/:id

Purpose: Get single opportunity with full details including source links

PATCH /api/opportunities/:id

Purpose: Update opportunity status and user notes

Body:

- status (optional): 'new', 'researching', 'building', 'rejected'
- user_notes (optional): Research notes text

POST /api/scan

Purpose: Trigger new opportunity scan

Response: Job ID for tracking scan progress

GET /api/scan/:job_id

Purpose: Check scan progress (pending/running/complete)

GET /api/stats

Purpose: Summary statistics (total, validated, high score, avg score)

GET /api/user/profile

Purpose: Get current user's profile (email, subscription tier, created date)

PATCH /api/user/profile

Purpose: Update user profile

Body: email (optional), current_password, new_password (optional)

GET /api/user/export-data

Purpose: GDPR data export - download all user data

Access: Pro and Premium tiers only

Returns: JSON file with all opportunities, notes, searches, exports

POST /api/email/configure-alerts

Purpose: Configure email alert settings

Body: alert_threshold (int), weekly_digest (bool)

POST /api/landing-page/generate

Purpose: Generate validation landing page for opportunity

Body: opportunity_id, custom_headline (optional), custom_cta (optional)

Returns: landing_page_url, slug

GET /api/export/csv

Purpose: Export filtered opportunities to CSV

Query Parameters: Same as GET /api/opportunities

GET /api/export/pdf

Purpose: Generate PDF report of top opportunities

Query Parameters: top_n (int, default: 10)

5.4 Admin API Endpoints

Authentication: Requires JWT with role='admin'

Access Control: All endpoints return 403 Forbidden if user is not admin

GET /api/admin/tiers

Purpose: List all subscription tiers

POST /api/admin/tiers

Purpose: Create new subscription tier

Body: name, price_monthly, max_sources, scan_frequency, export_limit_monthly, landing_pages_allowed, email_alerts_allowed

PATCH /api/admin/tiers/:id

Purpose: Update existing tier (price, features, enabled status)

GET /api/admin/sources

Purpose: List all data sources with stats

POST /api/admin/sources

Purpose: Add new data source

Body: name, type, config (JSON with API keys, keywords, etc.), rate_limit_per_minute

PATCH /api/admin/sources/:id

Purpose: Update source config or enable/disable

POST /api/admin/sources/:id/test

Purpose: Test source connection and return sample data

GET /api/admin/users

Purpose: List all users with subscription details

Query Parameters: search (email), status (active/cancelled), page, limit

PATCH /api/admin/users/:id

Purpose: Update user (cancel subscription, change role to admin)

GET /api/admin/analytics

Purpose: System analytics (users, MRR, opportunities, engagement)

PATCH /api/admin/settings

Purpose: Update system settings

Body: currency (GBP/USD), scoring_weights (JSON), default_scan_frequency, validation_rules (JSON)

Note: Changing currency triggers Stripe Price creation and tier price updates

GET /api/admin/trial-settings

Purpose: Get current free trial configuration

PATCH /api/admin/trial-settings

Purpose: Configure free trial settings

Body: enabled (bool), duration_days (int), trial_tier_id (UUID), max_sources (int), max_exports (int), landing_pages_allowed (bool), email_alerts_allowed (bool), min_opportunity_rank (int), require_card (bool), convert_to_tier_id (UUID)

5.2 Frontend Features

Authentication & Profile

- Registration form with email verification
- Login form
- "Forgot password" link → Email reset link

- Password reset page (token-based)
- User profile page:
  1. Change email
  2. Change password
  3. View subscription details
  4. Update payment method (Stripe portal)
  5. Export data button (Pro/Premium only)

Dashboard View

- 4 stat cards: Total, Validated, High Score, Average
- Search bar with real-time filtering
- Sort dropdown (score/revenue/mentions)
- Filter toggle with min score slider

Opportunity Cards

- Score badge (color-coded by range)
- Title and problem statement
- Revenue, mentions, competition, complexity
- Recommendation text

Detail Modal

- Full opportunity breakdown
- Source links (clickable)
- Market size estimate
- Existing competitor examples with direct links
- Action buttons (Mark as Researching/Building/Rejected)

Time-Based Filtering

Default View: "Top Opportunities" - All time opportunities sorted by score (highest first)

Purpose of Time Filters: When you need more context or saw something that's no longer on main page

- Filter options: Last 3 days, Last week, Last month, All time
- "Last 3 days" shows recent discoveries if something dropped off top page
- Time filters complement score sorting, don't replace it

Historical Archive

- All opportunities permanently stored (never deleted)
- Track mentions over time (trending up/down indicators)
- See when opportunity was first discovered
- Research notes and status tracking per opportunity

Email Alerts

- Automatic email when opportunity scores 80+ found
- Weekly digest of top 5 opportunities
- Configurable alert threshold in settings

Landing Page Builder

Quick validation tool: Generate a "Coming Soon - Join Waitlist" page for any opportunity to test demand before building.

- One-click landing page generation from opportunity data
- Customizable headline, description, and CTA
- Email capture form with Mailchimp/ConvertKit integration
- Hosted on subdomain (validate.opportunityfinder.app/[slug])

Export Functionality

- Export filtered opportunities to CSV
- Generate PDF report with top opportunities
- Share link for specific opportunity

5.3 Admin Panel

CRITICAL: Admin panel is core MVP. Without it, you cannot manage pricing, tiers, or data sources.

Access: Separate admin authentication. Admin users have role='admin' in database.

Pricing Management

- View all subscription plans
- Create new pricing tier (name, price, features)
- Edit existing tier price and features
- Enable/disable tiers
- Set feature gates per tier (sources allowed, scan frequency, export limits)

User Management

- View all users (email, subscription tier, status, created date)
- Search/filter users
- View user's subscription details
- Cancel user subscription
- Grant/revoke admin access

Data Source Management

- View all configured data sources
- Add new source (name, type, API credentials, search config)

- Edit source configuration (keywords, rate limits, enabled subreddits)
- Enable/disable source
- Test source connection
- View source statistics (requests made, opportunities found)

System Settings

- **Currency selection (£ GBP or $ USD)**
- Scoring algorithm weights (demand, revenue, competition, complexity)
- Default scan frequency (daily/weekly)
- Email alert thresholds
- Validation rules (minimum mentions, minimum revenue)
- API rate limit configuration

Free Trial Configuration

CRITICAL: Full flexibility to create any trial structure. Admin controls all trial parameters.

- **Enabled/Disabled:** Toggle free trials on/off globally
- **Duration:** Input field - any number of days (1, 5, 7, 14, 30, 60, etc.)
- **Trial Tier:** Which subscription tier during trial (Basic/Pro/Premium)
- **Feature Overrides:** Custom restrictions during trial:
  1. Max sources allowed (override tier default)
  2. Max exports allowed (override tier default)
  3. Landing pages enabled (yes/no)
  4. Email alerts enabled (yes/no)
  5. Opportunity rank access (which ranks visible)
- **Require Credit Card:** Collect payment method upfront (yes/no)
- **Auto-Convert To:** Which paid tier after trial ends (Basic/Pro/Premium)

Example Configurations:

- "5-day Premium trial, 3 exports max, card required, converts to Pro"
- "30-day Pro trial, no feature restrictions, no card, converts to Basic"
- "7-day Basic trial, rank 11+ access only, card required, converts to Basic"

Analytics Dashboard

- Total users, active subscriptions, MRR
- Opportunities discovered (total, by source)
- User engagement (searches, exports, landing pages created)
- System health (API errors, scan failures)

6. SUBSCRIPTION TIERS & PACKAGES

CRITICAL: Pricing is dynamic. Admin can create/modify tiers at any time without code changes.

The system supports multiple subscription tiers with different feature gates. Initial tiers are suggestions - admin can adjust pricing and features based on market response.

6.1 Suggested Initial Tiers

Basic Tier - £5/month

- 3 data sources (Reddit, Indie Hackers, ProductHunt)
- Weekly scans
- 5 exports per month
- Email alerts disabled
- Landing pages disabled
- **Access to opportunities ranked 11+ only**

Pro Tier - £15/month

- All 6 core data sources
- Daily scans
- Unlimited exports
- Email alerts enabled
- 5 landing pages per month
- **Access to opportunities ranked 6+ (includes mid-tier)**

Premium Tier - £30/month

- All data sources + admin can add more
- Daily scans + on-demand scans
- Unlimited exports
- Priority email alerts (instant for 90+ scores)
- Unlimited landing pages
- **Access to ALL opportunities including top 5 ranked (highest quality)**

6.2 Feature Gates

All features are gated by subscription tier. Frontend checks user's tier and disables features accordingly. Backend enforces limits.

Opportunity Ranking Access:

- Opportunities are ranked 1-N based on score (1=highest, N=lowest)
- Ranking recalculated after each scan
- Basic tier: Can only view opportunities ranked 11+ (lower quality)

- Pro tier: Can view opportunities ranked 6+ (includes mid-tier)
- Premium tier: Can view ALL opportunities including top 5 (highest quality)

Example Enforcement:

- User on Basic tries to export for 6th time this month → API returns 403 with "Upgrade to Pro for unlimited exports"
- User on Pro clicks 'Create Landing Page' → Works (allowed 5/month)
- User on Basic sees only 3 sources in UI, other sources greyed out with lock icon
- User on Basic sees opportunities ranked #11, #12, #13... but top 10 show as "Premium Only" with upgrade prompt

6.3 Upgrade/Downgrade Flow

- User clicks 'Upgrade' on any tier
- Redirects to Stripe checkout with new price
- Stripe handles proration automatically
- Webhook updates user.subscription_tier_id
- New features immediately available

6.4 Multi-Currency Support

CRITICAL: Admin can switch between £ GBP and $ USD. All pricing displays and Stripe products update accordingly.

Currency is a global system setting. When admin changes currency, all tier prices are converted at market rate and Stripe Price IDs are updated.

Implementation

- Admin selects currency in System Settings
- Backend converts all tier prices (£5 → $6, £15 → $19, £30 → $37)
- Creates new Stripe Price objects in USD currency
- Updates subscription_tiers.stripe_price_id for all tiers
- Frontend displays currency symbol (£ or $) from system_settings
- Existing subscriptions continue on current currency until renewal

Conversion Rates

- £5 → $6 (1.20x)
- £15 → $19 (1.27x)
- £30 → $37 (1.23x)

Note: Rates rounded to psychologically appealing prices ($6 not $6.25, $19 not $18.75)

6.5 Failed Payment Handling (Dunning)

Problem: Credit cards decline for expired cards, insufficient funds, bank fraud alerts, etc.

Solution: Automated retry logic with email notifications to recover failed payments.

Retry Schedule

- Day 0: Payment fails → User.subscription_status = 'past_due'
- Day 1: Automatic retry + email "Payment failed - please update card"
- Day 3: Second retry + email "Final notice - update card within 4 days"
- Day 7: Third retry + email "Subscription will be cancelled tomorrow"
- Day 8: Cancel subscription → User.subscription_status = 'cancelled'

User Experience During Past Due

- Login shows banner: "Payment failed. Update card to continue service."
- Full access continues for 7 days (grace period)
- User can update card in profile → triggers immediate retry
- After 8 days: Access revoked, downgraded to no access until payment succeeds

Stripe Webhooks

- invoice.payment_failed → Set status to past_due, send first email
- invoice.payment_succeeded → Set status to active, restore full access
- customer.subscription.deleted → Set status to cancelled, revoke access

7. DATABASE SCHEMA

CRITICAL: Multi-tenant architecture from day 1. All user-specific data includes user_id foreign key.

7.1 users Table

Column	Type	Description
id	`UUID PRIMARY KEY`	User unique identifier
email	`VARCHAR(255) UNIQUE`	Login email
email_verified	`BOOLEAN`	Email confirmed via verification link
password_hash	`VARCHAR(255)`	Bcrypt hashed password
role	`VARCHAR(50)`	user/admin
stripe_customer_id	`VARCHAR(255)`	Stripe customer ID
subscription_tier_id	`UUID REFERENCES tiers(id)`	Current subscription tier
subscription_status	`VARCHAR(50)`	trial/active/cancelled/past_due
subscription_end_date	`TIMESTAMP`	When subscription expires
trial_ends_at	`TIMESTAMP`	When free trial ends (null if not on trial)
created_at	`TIMESTAMP`	Account creation date
last_login	`TIMESTAMP`	Last login time

7.2 subscription_tiers Table

Purpose: Define pricing tiers and feature gates. Admin can create/modify tiers dynamically.

Column	Type	Description
id	`UUID PRIMARY KEY`	Tier unique identifier
name	`VARCHAR(100)`	Tier name (e.g., Basic, Pro, Premium)
price_monthly	`INTEGER`	Price in pence (e.g., 500 = £5.00)

stripe_price_id	VARCHAR(255)	Stripe Price ID
max_sources	INTEGER	Number of data sources allowed
min_opportunity_rank	INTEGER	Minimum rank to view (1=top, 11+=low)
scan_frequency	VARCHAR(50)	daily/weekly/monthly
export_limit_monthly	INTEGER	Max exports per month (-1 = unlimited)
landing_pages_allowed	BOOLEAN	Can create landing pages
email_alerts_allowed	BOOLEAN	Receives email alerts
enabled	BOOLEAN	Tier available for signup
created_at	TIMESTAMP	When tier created
updated_at	TIMESTAMP	Last modified

7.3 data_sources Table

Purpose: Dynamic data source configuration. Admin can add new sources without code changes.

Column	Type	Description
id	UUID PRIMARY KEY	Source unique identifier
name	VARCHAR(100)	Display name (e.g., Reddit, Twitter)
type	VARCHAR(50)	reddit/api/scraper/rss
config	JSONB	Source-specific config (API keys, keywords, etc.)
enabled	BOOLEAN	Currently active
rate_limit_per_minute	INTEGER	Max requests per minute
last_scan	TIMESTAMP	Last successful scan
total_opportunities_found	INTEGER	Lifetime count
created_at	TIMESTAMP	When added
updated_at	TIMESTAMP	Last modified

7.4 opportunities Table

Column	Type	Description
id	SERIAL PRIMARY KEY	Auto-incrementing ID
user_id	UUID REFERENCES users(id)	Owner of this opportunity record
title	VARCHAR(255)	Opportunity title
problem	TEXT	Problem statement
score	INTEGER	Calculated score (0-100)
rank	INTEGER	Ranking position (1=best, updated on each scan)
mentions	INTEGER	Total mentions across sources
mentions_trend	VARCHAR(20)	up/down/stable
revenue	VARCHAR(100)	Display string (e.g. "£5k MRR")
revenue_amount	INTEGER	Numeric amount for sorting
competitors	INTEGER	Number of competitors
competition_level	VARCHAR(50)	Low/Medium/High/Very High
build_complexity	VARCHAR(50)	Low/Medium/High
sources	JSONB	Array of source names
source_urls	JSONB	Array of source URLs
example	TEXT	Existing competitor examples
competitor_urls	JSONB	Array of competitor URLs
validated	BOOLEAN	Passed validation criteria
recommendation	TEXT	Action recommendation
market_size	TEXT	Market size description
status	VARCHAR(50)	new/researching/building/rejected
user_notes	TEXT	User research notes
created_at	TIMESTAMP	When first discovered
updated_at	TIMESTAMP	Last updated
last_seen	TIMESTAMP	Last time mentioned in sources

7.5 pain_points Table

Column	Type	Description
id	SERIAL PRIMARY KEY	Auto-incrementing ID
user_id	UUID REFERENCES users(id)	User who discovered this
source	VARCHAR(100)	Source platform (e.g. r/Entrepreneur)
text	TEXT	Pain point text
url	TEXT	Source URL
mentions	INTEGER	Frequency count
created_at	TIMESTAMP	When collected

7.6 scan_jobs Table

Column	Type	Description
id	UUID PRIMARY KEY	Job ID
user_id	UUID REFERENCES users(id)	User who initiated scan
status	VARCHAR(50)	pending/running/complete/failed
started_at	TIMESTAMP	Start time
completed_at	TIMESTAMP	Completion time
opportunities_found	INTEGER	Number found
error_message	TEXT	Error if failed

7.7 system_settings Table

Purpose: Global system configuration. Single-row table (id=1).

Column	Type	Description
id	INTEGER PRIMARY KEY	Always 1 (singleton)
currency	VARCHAR(3)	GBP or USD
currency_symbol	VARCHAR(5)	£ or $
scoring_weights	JSONB	Algorithm weights {demand:25,

		revenue:35...}
default_scan_frequency	VARCHAR(50)	daily/weekly
validation_rules	JSONB	Min mentions, min revenue thresholds
updated_at	TIMESTAMP	Last modified
updated_by	UUID REFERENCES users(id)	Admin who made change

7.8 trial_settings Table

Purpose: Free trial configuration. Single-row table (id=1).

Column	Type	Description
id	INTEGER PRIMARY KEY	Always 1 (singleton)
enabled	BOOLEAN	Free trials enabled/disabled
duration_days	INTEGER	Trial length in days
trial_tier_id	UUID REFERENCES tiers(id)	Which tier during trial
max_sources	INTEGER	Source limit override (null = use tier)
max_exports	INTEGER	Export limit override (null = use tier)
landing_pages_allowed	BOOLEAN	Landing page access override
email_alerts_allowed	BOOLEAN	Email alert override
min_opportunity_rank	INTEGER	Rank access override (null = use tier)
require_card	BOOLEAN	Require payment method upfront
convert_to_tier_id	UUID REFERENCES tiers(id)	Tier after trial ends
updated_at	TIMESTAMP	Last modified
updated_by	UUID REFERENCES users(id)	Admin who made change

8. COMPETITIVE ANALYSIS

The micro-SaaS validation space has several players, but most are static databases or manual services. Our competitive advantage is real-time automated scanning with systematic scoring.

8.1 Direct Competitors

BigIdeasDB

Pricing: £39-79/month

Strengths:

- Aggregates Reddit, G2, ProductHunt, app stores
- AI-powered research assistant (BuildGuide)
- Millions of data points

Weaknesses: No systematic scoring, no build complexity assessment, expensive for indie developers

MicroSaaSHQ

Pricing: £130 one-time payment

Strengths:

- 1,200+ pre-validated ideas
- Market data and competition analysis
- AI build prompts for Cursor/Lovable

Weaknesses: Static database (not live scraping), no automated updates, no personalized discovery

Validator (ProductHunt Focus)

Pricing: Unknown (new product)

Strengths:

- Automated competitor analysis
- Trend analysis

Weaknesses: Only ProductHunt data, limited sources, newer player with less data

8.2 Adjacent Competitors

ValidateMySaaS

Pricing: £29-99 per report

Model: Manual competitor analysis reports (24hr turnaround)

Weaknesses: Manual process, slow, pay per validation, not continuous monitoring

Exploding Topics

Pricing: £39-79/month

Focus: Trend discovery, rising search terms

Weaknesses: Not SaaS-specific, no validation framework, requires manual interpretation

8.3 Our Competitive Advantages

4. **Real-Time Automated Scanning:** Not a static database - continuously monitors sources
5. **Systematic Scoring Algorithm:** Clear 0-100 scores with actionable recommendations, not just idea lists
6. **Multi-Source Validation:** 6+ data sources vs competitors' 1-3
7. **Developer-First:** Includes tech stack, build complexity, not just market data
8. **Historical Tracking:** See trending data over time, not point-in-time snapshots
9. **Revenue Validation:** Won't show opportunities unless existing solutions make £1k+ MRR
10. **Price Point:** £5/month vs £39-130 - accessible to bootstrappers

8.4 Market Positioning

Position: "The systematic validation engine for indie developers - not guessing, not static lists, but live market signals with clear build/no-build scores."

Target: Developers building their first micro-SaaS who want validation BEFORE building, not idea inspiration.

9. IMPLEMENTATION PLAN

9.1 Phase 1: Core Backend + Auth + Admin (Days 1-5)

11. Set up Flask project structure
12. Implement database models with multi-tenant architecture (SQLAlchemy)
13. Build authentication system (JWT, bcrypt, register/login endpoints)
14. Set up Stripe integration (checkout, webhooks)
15. Build Reddit collector with PRAW
16. Implement basic scoring algorithm
17. Create API endpoints with auth middleware (GET /opportunities, GET /stats)

Success Criteria: Can register/login, pay via Stripe, collect Reddit data, score it, save to multi-tenant DB, serve via authenticated API

9.2 Phase 2: Data Sources + Tiers (Days 6-10)

18. Indie Hackers scraper
19. ProductHunt API integration
20. HackerNews Algolia API
21. Google Search via SerpAPI
22. Microns.io / Acquire.com scraper

Success Criteria: All sources collecting data, aggregation working, scores accurate

9.3 Phase 3: Frontend + Admin Panel (Days 11-15)

23. Set up React project (Vite)
24. Build authentication pages (login, register, password reset)
25. Implement Stripe checkout flow and billing portal
26. Build dashboard layout with dark/light mode toggle
27. Implement opportunity cards with filtering and time ranges
28. Create detail modal with status tracking
29. Connect to backend API with JWT auth

Success Criteria: Full auth flow works, users can pay and access app, UI connected to live backend, both themes functional, mobile responsive

9.4 Phase 4: Polish + Deploy (Days 16-21)

30. Add loading states and error handling
31. Implement background job system for scans
32. Build email alert system (SendGrid/Mailgun integration)
33. Create landing page builder with email capture
34. Implement CSV/PDF export functionality
35. Set up deployment (Nginx, Gunicorn, PostgreSQL)
36. Configure automated weekly scans (cron)

37. Testing and bug fixes

Success Criteria: Deployed to VPS, automated scans running, email alerts working, landing pages generating, exports functional, no critical bugs

10. ACCEPTANCE CRITERIA

10.1 Backend Requirements

- ✓ User authentication working (register, login, JWT tokens)
- ✓ Stripe integration functional (checkout, webhooks, subscription status updates)
- ✓ Multi-tenant data isolation (users only see their own opportunities)
- ✓ Collects data from all 6 sources (Reddit, IH, PH, HN, Google, Microns)
- ✓ Scores opportunities correctly (manual verification with 10 test cases)
- ✓ Stores data in PostgreSQL with proper foreign keys
- ✓ API returns JSON with correct schema
- ✓ Handles rate limits gracefully (no crashes)
- ✓ Weekly automated scans run via cron

10.2 Frontend Requirements

- ✓ Registration flow with email verification works
- ✓ Login and logout functional
- ✓ Password reset flow works (forgot password → email → reset)
- ✓ User profile page works (change email, change password)
- ✓ Stripe checkout integration functional
- ✓ Free trial displays correctly (if enabled)
- ✓ Failed payment banner shows for past_due status
- ✓ Subscription status displayed correctly
- ✓ Displays all opportunities from API
- ✓ Default view shows top opportunities (sorted by score)
- ✓ Tier-based rank filtering works (Basic sees 11+, Pro sees 6+, Premium sees all)
- ✓ Search works in real-time
- ✓ Filters work (min score slider, time range, status)
- ✓ Sorting works (score/revenue/mentions)
- ✓ Detail modal shows all data with clickable source links
- ✓ Dark/light mode toggle works
- ✓ Status tracking (mark as researching/building/rejected)
- ✓ Email alerts configured and sending correctly
- ✓ Export to CSV/PDF functional
- ✓ Data export (GDPR) works for Pro/Premium users
- ✓ Landing page builder generates valid pages
- ✓ Responsive design (works on mobile)
- ✓ Loading states for API calls

10.3 Deployment Requirements

- ✓ Runs on VPS (accessible via domain or IP)

- ✓ Also runs locally (development mode)
- ✓ Environment variables for API keys
- ✓ Database migrations work
- ✓ Setup documentation complete

10.4 Quality Criteria

- ✓ 80%+ of score 60+ opportunities validate with landing page
- ✓ System finds 10+ new opportunities per week
- ✓ No crashes during 7-day continuous operation
- ✓ API response time < 500ms for GET /opportunities

11. OUT OF SCOPE (MVP v1.0)

These features are explicitly NOT included in MVP v1.0:

- Team/multi-user accounts (MVP is single-user per account)
- Saving favorites/bookmarks
- Advanced NLP/clustering for theme detection
- Competitor deep analysis (beyond basic URL listing)
- Custom data source configuration
- Mobile apps (iOS/Android) - architecture supports future development
- Slack/Discord integrations
- Browser extension for pain point capture

11.1 Post-MVP Features (Next Iteration)

DOCUMENTED FOR FUTURE DEVELOPMENT: These features are required but deferred to post-MVP iteration.

Both features are essential for user success but can be added after MVP launch based on initial user feedback.

Help / FAQ Section

Purpose: Self-service support to reduce admin workload

- Searchable FAQ database
- Categories: Getting Started, Pricing, Data Sources, Scoring, Exports, Billing
- Video tutorials (3-5 minute screencasts)
- Contact form for unanswered questions
- Admin can add/edit FAQ entries without code changes

Onboarding Flow

Purpose: Guide new users to first value (seeing validated opportunities) in under 5 minutes

- Step 1: Account created → Show welcome modal
- Step 2: "Let's run your first scan" → Click button, show progress
- Step 3: Results appear → Highlight top opportunity with tooltip
- Step 4: Click opportunity → Modal shows "This is how opportunities work"
- Step 5: "Try filtering by score" → Highlight filter controls
- Step 6: Done → "You're ready! Explore on your own or upgrade for more features"

Implementation:

- Use library like Shepherd.js or Intro.js for step-by-step tooltips
- User.onboarding_completed flag tracks progress

- "Skip tutorial" option available at any step

Rationale: MVP v1.0 is a complete validation platform with auth, payments, email alerts, landing page builder, export functionality, and full admin control. Help/FAQ and Onboarding improve UX but aren't blockers for launch.

12. ENVIRONMENT 11. ENVIRONMENT & SETUP SETUP

12.1 Required API Keys

- **Reddit:** client_id, client_secret (https://www.reddit.com/prefs/apps)
- **ProductHunt:** API token (https://api.producthunt.com/v2/docs)
- **SerpAPI:** API key (https://serpapi.com)

12.2 Development Environment

- Python 3.11+
- Node.js 18+
- PostgreSQL 15+
- Git

12.3 Project Structure

```
opportunity-finder/ ├── backend/ │   ├── app.py                # Flask application │
├── models.py             # Database models │   ├── collectors/           # Data
collectors │   │   ├── reddit.py │   │   ├── indie_hackers.py │   │   ├──
producthunt.py │   │   ├── hackernews.py │   │   └── google_search.py │   ├──
scoring.py            # Scoring algorithm │   ├── requirements.txt │   └── config.py
# Configuration ├── frontend/ │   ├── src/ │   │   ├── components/ │   │   ├── api/ │
│   └── App.jsx │   ├── package.json │   └── vite.config.js └── README.md
```

13. RISKS 12. RISKS & MITIGATION MITIGATION

Risk	Likelihood	Impact	Mitigation
API rate limits block collection	High	Medium	Exponential backoff + caching
Scoring algorithm inaccurate	Medium	High	Manual validation of 50 test cases
Web scraping breaks	Medium	Medium	Monitor errors, fallback to APIs
Low data quality	Medium	High	Multiple source validation
Performance issues at scale	Low	Medium	Database indexing + pagination

14. SUCCESS METRICS

14.1 Product Metrics

- **Validation Accuracy:** 80%+ of score 60+ opportunities validate with landing page test
- **Discovery Rate:** 10+ validated opportunities discovered per week
- **System Uptime:** 99%+ uptime over 30 days
- **Data Freshness:** All data sources updated within 7 days

14.2 User Metrics (If Offered as SaaS)

- **Time to First Value:** < 5 minutes from signup to viewing opportunities
- **Weekly Active Users:** Track returning users
- **Conversion Rate:** Users who validate an opportunity

APPENDIX A: REQUIRED ACCOUNTS & API KEYS

The following accounts and API keys are required to build and deploy the MVP. Most have free tiers sufficient for initial launch.

A.1 Essential (Must Have for MVP)

Stripe

Purpose: Payment processing and subscription management

URL: https://stripe.com

Cost: 2.9% + 20p per transaction

Keys Needed: Publishable Key, Secret Key, Webhook Secret

SendGrid or Mailgun

Purpose: Email alerts and notifications

URL: https://sendgrid.com or https://mailgun.com

Cost: Free tier: 100 emails/day (SendGrid), 5,000 emails/month (Mailgun)

Keys Needed: API Key

Reddit API

Purpose: Primary pain point discovery from subreddits

URL: https://www.reddit.com/prefs/apps

Cost: Free (rate limited: 60 requests/minute)

Keys Needed: Client ID, Client Secret

ProductHunt API

Purpose: Product launches and user feedback

URL: https://api.producthunt.com/v2/docs

Cost: Free (GraphQL API)

Keys Needed: API Token

SerpAPI

Purpose: Google Search for competitor discovery

URL: https://serpapi.com

Cost: Free tier: 100 searches/month, then $50/month for 5,000 searches

Keys Needed: API Key

A.2 Infrastructure (Deployment)

VPS Provider

Purpose: Host backend and database

Options: DigitalOcean, Hetzner, Linode

Cost: £10-20/month for 2GB RAM droplet

Domain Name

Purpose: opportunityfinder.app or similar

Provider: Namecheap, Cloudflare, Google Domains

Cost: £10-15/year

A.3 Optional (Can Start Without)

Mailchimp or ConvertKit

Purpose: Landing page email capture integration

Cost: Free tier: 500 contacts

Alternative: Store emails in database initially, migrate later

A.4 Account Status Checklist

- ❒ Stripe account created and verified
- ❒ SendGrid/Mailgun API key obtained
- ❒ Reddit app created (client ID/secret)
- ❒ ProductHunt API token obtained
- ❒ SerpAPI key obtained (100 free searches)
- ❒ VPS provisioned (2GB+ RAM, Ubuntu 24)
- ❒ Domain purchased and DNS configured

APPENDIX B: EXAMPLE API RESPONSES

GET /api/opportunities


```
{   "success": true,   "data": [     {       "id": 1,       "title": "Testimonial
Collection Tool",       "problem": "Businesses struggle to collect customer
testimonials efficiently",       "score": 88,       "mentions": 67,       "revenue":
"£83,000 MRR",       "revenue_amount": 83000,       "competitors": 4,
"competition_level": "Medium",       "build_complexity": "Low",       "sources":
["r/Entrepreneur", "r/smallbusiness", "Indie Hackers"],       "example": "Senja.io",
"validated": true,       "recommendation": "Build immediately",       "market_size":
"200k+ businesses need testimonials",       "created_at": "2025-01-08T10:30:00Z"     }
],   "count": 1 }
```

DOCUMENT CONTROL

Version History

Version	Date	Author	Changes
1.0	10/01/2026	Mark	Initial PRD - Core validation system
1.1	10/01/2026	Mark	Added multi-tenant architecture, authentication, Stripe payments, mobile-ready API design, competitive analysis
1.2	10/01/2026	Mark	Moved Email Alerts, Landing Page Builder, and Export Functionality into core MVP v1.0 (removed from out of scope)
1.3	10/01/2026	Mark	Added Admin Panel, subscription tiers/packages, dynamic data source management, required accounts list, updated timeline to 2-3 weeks
1.4	10/01/2026	Mark	Added tier-based opportunity ranking access (Basic: rank 11+, Pro: rank 6+, Premium: all), multi-currency support (£ GBP / $ USD switchable)
1.5	10/01/2026	Mark	Added password reset, email verification, user profile page, failed payment handling (dunning), GDPR data export (Pro/Premium), flexible free trial configuration, documented Help/FAQ and Onboarding for post-MVP

Approvals

Document Owner: Mark

Status: APPROVED FOR DEVELOPMENT

Date: 10/01/2026

— END OF DOCUMENT —