Product Requirements Document (PRD)

MMAP: Multi-Modal Agent Assessment Protocol

Version: 1.0
Date: October 28, 2025
Owner: [Your Name]
Status: Draft → Review → Approved
Last Updated: October 28, 2025

Table of Contents

1. Executive Summary
2. Product Vision & Strategy
3. Problem Statement
4. Market Analysis
5. Target Users & Personas
6. User Stories & Use Cases
7. Product Requirements
8. Technical Architecture
9. Success Metrics
10. Go-to-Market Strategy
11. Roadmap
12. Open Questions & Risks

---

1. Executive Summary

Product Overview

MMAP (Multi-Modal Agent Assessment Protocol) is a comprehensive evaluation framework for autonomous AI agents that addresses the critical gap between development testing and production readiness.

The Opportunity

- **Market Size:** $50B+ AI/ML operations market growing at 40% CAGR
- **Problem:** 67% of AI agents fail in production despite passing development tests

- **Impact:** Companies lose $50K-$500K per incident from agent failures
- **Solution Gap:** No standardized, comprehensive evaluation framework exists

Product Strategy

Launch as **open-source framework** with **premium SaaS offerings** for continuous monitoring, team collaboration, and enterprise features.

Success Criteria (Year 1)

- 5,000+ GitHub stars
- 100+ production deployments
- 10+ paying enterprise customers
- $500K ARR from premium features

---

2. Product Vision & Strategy

Vision Statement

"Make every AI agent production-ready through comprehensive, standardized evaluation."

Mission

Transform AI agent evaluation from ad-hoc testing to systematic assessment across technical, business, and ethical dimensions.

Product Principles

1. **Comprehensive Over Convenient**
   Evaluate the entire agent system, not just model outputs
2. **Production-First**
   Design for continuous monitoring, not one-time testing
3. **Framework Agnostic**
   Work with any agent architecture (LangChain, LlamaIndex, custom)
4. **Transparent & Reproducible**
   Clear metrics, open methodology, reproducible results
5. **Community-Driven**
   Built with and for practitioners, guided by real use cases

Strategic Positioning

Primary Positioning:

"The evaluation framework that catches agent failures before production"

Secondary Positioning:
"MLflow for AI agents" - standardized evaluation and monitoring

---

3. Problem Statement

The Problem

Companies are deploying AI agents that appear to work but fail critically in production.

Pain Points

For ML Engineers:

- Unit tests only cover model behavior, not system integration
- No standardized way to evaluate multi-modal interactions
- Hard to catch edge cases and policy violations
- Can't measure fairness and bias systematically
- Difficult to explain agent failures to stakeholders

For Engineering Leaders:

- Can't assess production readiness objectively
- No visibility into agent reliability before deployment
- Risk of costly production failures
- Lack of compliance documentation for audits
- No way to compare agent performance across projects

For Business Stakeholders:

- Agents fail on critical business logic despite technical tests
- No confidence in deployment decisions
- Expensive post-deployment fixes and incidents
- Customer trust erosion from agent mistakes
- Regulatory compliance concerns

Current Solutions & Why They Fail

Approach	Limitations
Traditional Unit Tests	Only test individual functions, miss system-level failures
Manual Review	Doesn't scale, inconsistent, catches issues too late
Model-Only Evaluation	Ignores API integration, business logic, fairness
Custom Scripts	Non-standardized, hard to maintain, no benchmarks
LLM-as-Judge	Limited to output quality, misses system issues

The Gap

No comprehensive framework evaluates agents across input/output, model performance, system integration, business logic, AND fairness/compliance.

---

4. Market Analysis

Target Market

Primary Market: Companies deploying autonomous AI agents
Market Size: $50B AI/ML operations market, 40% CAGR
Addressable Segment: 10,000+ companies building production agents

Market Segments

1. Early Adopters (Primary Focus)

- AI-native companies (OpenAI, Anthropic users)
- Funded AI startups ($2M+ raised)
- Enterprise digital transformation teams
- **Size:** 1,000-2,000 companies
- **Willingness to Pay:** High ($10K-100K annually)

2. Enterprise AI Teams

- Fortune 500 with AI initiatives
- Banks, healthcare, e-commerce deploying agents
- **Size:** 5,000+ companies
- **Willingness to Pay:** Very high ($50K-500K annually)

3. Individual ML Practitioners

- Freelance consultants
- Academic researchers
- Side project builders
- **Size:** 50,000+ individuals
- **Willingness to Pay:** Low (free/open-source preferred)

Competitive Landscape

Direct Competitors

None. No comprehensive agent evaluation framework exists.

Indirect Competitors

Solution	Strengths	Weaknesses	Our Advantage
LangSmith	Great observability, LangChain integration	Limited evaluation metrics, no systematic framework	Comprehensive 5-layer evaluation
Weights & Biases	Strong ML experiment tracking	Not agent-specific, focuses on training	Production agent focus
Confident AI	LLM evaluation metrics	Narrow scope (output quality only)	System-level evaluation
Custom Solutions	Tailored to specific needs	Non-standardized, not reusable	Standardized, extensible

Competitive Moats

1. **First-Mover:** First comprehensive agent evaluation framework
2. **Community:** Open-source adoption creates network effects
3. **Methodology:** 16 years ML experience → credible framework
4. **Data:** Aggregate anonymized evaluation data for benchmarks
5. **Integrations:** Deep integration with popular agent frameworks

---

5. Target Users & Personas

Primary Persona: "Alex the AI Engineer"

Demographics:

- Age: 28-38
- Role: ML Engineer, AI Engineer, Data Scientist

- Company: AI startup or enterprise AI team
- Experience: 3-7 years in ML/AI

Goals:

- Deploy reliable AI agents to production
- Catch failures before customers do
- Prove agent readiness to stakeholders
- Maintain high quality bar

Pain Points:

- "My agent works in dev but fails in production"
- "I don't know if my agent is actually ready"
- "Stakeholders ask questions I can't answer with data"
- "Manual testing doesn't scale"

Tools They Use:

- Python, LangChain/LlamaIndex
- GitHub, Docker, Kubernetes
- Weights & Biases, MLflow
- Jupyter notebooks

Quote: *"I need to know my agent won't embarrass me in production."*

Secondary Persona: "Morgan the Engineering Manager"

Demographics:

- Age: 35-45
- Role: Engineering Manager, ML Lead, Director of AI
- Company: Scale-up or enterprise
- Experience: 10+ years, managing 5-15 engineers

Goals:

- Ensure production readiness across projects
- Standardize evaluation practices

- Reduce post-deployment incidents
- Build stakeholder confidence

Pain Points:

- "Each team evaluates differently"
- "Can't compare agent quality across projects"
- "Too many production incidents"
- "Hard to explain agent reliability to executives"

Buying Authority: High ($10K-50K budget)

Quote: *"I need standardized evaluation so I can sleep at night."*

Tertiary Persona: "Sam the ML Consultant"

Demographics:

- Age: 30-50
- Role: Independent ML consultant
- Clients: Multiple companies, 2-10 projects/year
- Experience: 8-15 years

Goals:

- Deliver high-quality agent implementations
- Differentiate from competitors
- Show measurable value to clients
- Build reputation

Pain Points:

- "Need to prove my agents work better than others"
- "Clients ask for evidence of quality"
- "Can't charge premium without proof of value"

Willingness to Pay: Medium (prefers open-source + paid consulting)

Quote: *"MMAP helps me charge 2x because I can prove quality."*

---

6. User Stories & Use Cases

Epic 1: Agent Evaluation Setup

User Story 1.1: Quick Start

As an AI engineer,
I want to evaluate my agent in under 10 minutes,
So that I can quickly check if it's production-ready.

Acceptance Criteria:
- Install MMAP with single pip command
- Run evaluation with < 10 lines of code
- Get readable report in terminal
- See pass/fail status for each layer

User Story 1.2: Custom Metrics

As an ML engineer,
I want to add custom evaluation metrics,
So that I can evaluate domain-specific requirements.

Acceptance Criteria:
- Extend BaseMetric class easily
- Add custom metric to evaluation pipeline
- See custom metric in reports
- Share custom metrics with team

User Story 1.3: Framework Integration

As a LangChain user,
I want MMAP to work seamlessly with my existing code,
So that I don't have to rewrite my agent.

Acceptance Criteria:
- Import LangChain integration
- Wrap existing agent with MMAP evaluator
- Run evaluation without code changes
- Get LangChain-specific insights

Epic 2: Comprehensive Evaluation

User Story 2.1: Multi-Layer Assessment

As an engineering manager,
I want to evaluate agents across all 5 layers,
So that I catch issues that unit tests miss.

Acceptance Criteria:
- Input/output validation complete
- Model performance measured
- API integration tested
- Business logic verified
- Fairness assessed

User Story 2.2: Failure Detection

As an AI engineer,
I want MMAP to identify specific failure modes,
So that I know exactly what to fix.

Acceptance Criteria:
- Clear description of each failure
- Severity rating (critical/warning/info)
- Suggested remediation steps
- Links to relevant documentation

User Story 2.3: Benchmark Comparison

As an ML engineer,
I want to compare my agent against benchmarks,
So that I know if performance is good or bad.

Acceptance Criteria:
- See percentile ranking vs other agents
- Compare to industry standards
- Identify improvement opportunities
- Track improvement over time

Epic 3: Reporting & Documentation

User Story 3.1: Executive Reports

As an engineering manager,
I want to generate executive-friendly reports,
So that I can communicate agent quality to stakeholders.

Acceptance Criteria:
- One-page summary with key metrics
- Risk assessment (high/medium/low)
- Business impact estimation
- Export to PDF/PowerPoint

User Story 3.2: Audit Trails

As a compliance officer,
I want complete audit trails of all evaluations,
So that I can demonstrate due diligence to regulators.

Acceptance Criteria:
- All evaluations logged with timestamps
- Immutable evaluation records
- Exportable compliance reports
- GDPR/SOC2 compliant storage

User Story 3.3: Visual Dashboards

As an AI engineer,
I want interactive visual dashboards,
So that I can quickly understand evaluation results.

Acceptance Criteria:
- HTML dashboard auto-generated
- Interactive charts and graphs
- Drill-down into specific failures
- Shareable via URL

Epic 4: Continuous Monitoring

User Story 4.1: Production Monitoring

As an engineering manager,
I want to monitor agent performance in production,
So that I catch degradation before customers complain.

Acceptance Criteria:
- Real-time metric tracking
- Automatic alerts on threshold breaches
- Weekly performance reports
- Trend analysis over time

User Story 4.2: A/B Testing

As an AI engineer,
I want to compare two agent versions,
So that I can deploy the better one.

Acceptance Criteria:
- Run same evaluation on both versions
- Side-by-side comparison report
- Statistical significance tests
- Recommendation on which to deploy

User Story 4.3: Regression Detection

As an ML engineer,
I want to be alerted if agent quality regresses,
So that I can fix it immediately.

Acceptance Criteria:
- Baseline evaluation stored
- Automatic comparison on new evaluations
- Slack/email alerts on regression
- GitHub issue auto-created

Epic 5: Team Collaboration

User Story 5.1: Team Workspaces

As an engineering manager,
I want my team to share evaluations,
So that we can collaborate on improvements.

Acceptance Criteria:
- Team workspace with all evaluations
- Role-based access control
- Comments and discussions
- Version history

User Story 5.2: CI/CD Integration

As a DevOps engineer,
I want MMAP in our CI/CD pipeline,
So that we block bad agents from deploying.

Acceptance Criteria:
- GitHub Actions integration
- Pass/fail status in PR
- Block merge on critical failures
- Evaluation artifacts stored

---

7. Product Requirements

MVP (Minimum Viable Product) - v0.1-0.5

Timeline: Months 1-3
Goal: Prove core value proposition with minimal feature set

Core Features (Must Have)

F1: 5-Layer Evaluation Framework

- Layer 1: Input/Output Validation
- Layer 2: Model Performance
- Layer 3: System Integration
- Layer 4: Business Logic
- Layer 5: Fairness & Compliance
- **Priority:** P0 (Critical)
- **Effort:** 4 weeks

F2: 10 Pre-Built Metrics

- 2 metrics per layer minimum
- Intent accuracy, entity extraction
- Decision accuracy, hallucination detection
- API latency, transaction success
- Policy compliance, edge case handling
- Demographic parity, audit trail
- **Priority:** P0 (Critical)
- **Effort:** 2 weeks

F3: Python SDK/Library

- Pip-installable package
- Simple API: `AgentEvaluator` class
- Metric registration system
- Test case loader (JSON/CSV)
- **Priority:** P0 (Critical)
- **Effort:** 3 weeks

F4: Command-Line Interface

- `mmap init` - scaffold new evaluation
- `mmap run` - execute evaluation
- `mmap report` - view results
- **Priority:** P1 (High)
- **Effort:** 1 week

F5: Terminal Report Output

- Formatted text summary
- Pass/fail status with colors
- Score for each metric
- Critical issues highlighted
- **Priority:** P0 (Critical)

- **Effort:** 1 week

F6: JSON Report Export

- Complete evaluation results
- Structured, machine-readable
- Timestamp and metadata
- **Priority:** P0 (Critical)
- **Effort:** 3 days

F7: Refund Agent Example

- Complete working example
- 50+ test cases
- All 5 layers evaluated
- Documented findings
- **Priority:** P0 (Critical)
- **Effort:** 1 week

F8: Documentation

- README with quick start
- API reference
- Example usage
- Contributing guide
- **Priority:** P0 (Critical)
- **Effort:** 1 week

MVP Success Criteria

- ✅ Can evaluate any Python agent in <10 minutes
- ✅ Catches failures traditional testing misses
- ✅ 500+ GitHub stars within 30 days
- ✅ 5+ community contributions
- ✅ 3+ companies using in production

---

v1.0 - Production Ready

Timeline: Months 4-6
Goal: Enterprise-ready with integrations and advanced features

Must-Have Features

F9: HTML Dashboard

- Interactive web-based report
- Charts and visualizations
- Drill-down capabilities
- Shareable via static HTML
- **Priority:** P0 (Critical)
- **Effort:** 3 weeks

F10: Framework Integrations

- LangChain evaluator wrapper
- LlamaIndex support
- AutoGen compatibility
- CrewAI integration
- **Priority:** P0 (Critical)
- **Effort:** 4 weeks

F11: Extended Metrics Library

- 30+ metrics total (6 per layer)
- Domain-specific metrics (finance, healthcare, e-commerce)
- Confidence intervals
- Statistical significance tests
- **Priority:** P1 (High)
- **Effort:** 3 weeks

F12: Batch Evaluation

- Evaluate multiple agents at once
- Comparison reports

- Performance benchmarking
- **Priority:** P1 (High)
- **Effort:** 2 weeks

F13: Test Dataset Builder

- GUI for creating test cases
- Import from CSV/JSON
- Synthetic data generation
- Ground truth labeling interface
- **Priority:** P2 (Medium)
- **Effort:** 3 weeks

Should-Have Features

F14: CI/CD Integration

- GitHub Actions workflow
- GitLab CI support
- Pass/fail PR status
- **Priority:** P1 (High)
- **Effort:** 2 weeks

F15: Regression Testing

- Store baseline evaluations
- Automatic comparison
- Alert on degradation
- **Priority:** P1 (High)
- **Effort:** 2 weeks

F16: Plugin System

- Custom metric plugins
- Custom reporters
- Extensibility framework
- **Priority:** P2 (Medium)

- **Effort:** 2 weeks

---

v2.0 - SaaS Platform (Premium)

Timeline: Months 7-12

Goal: Monetization through premium features

Premium Features (SaaS Only)

F17: Continuous Monitoring

- Real-time production monitoring
- Automatic evaluation scheduling
- Trend analysis and alerts
- **Business Model:** $99-499/month per team
- **Priority:** P0 (Critical for monetization)
- **Effort:** 6 weeks

F18: Team Collaboration

- Shared workspaces
- Multi-user access
- Role-based permissions
- Comments and annotations
- **Business Model:** $199-999/month per team
- **Priority:** P0 (Critical for enterprise)
- **Effort:** 4 weeks

F19: Cloud-Hosted Evaluations

- Run evaluations in managed cloud
- No infrastructure needed
- Automatic scaling
- **Business Model:** Usage-based pricing
- **Priority:** P1 (High)
- **Effort:** 6 weeks

F20: Advanced Analytics

- Cross-project comparison
- Industry benchmarks
- Predictive failure analysis
- ROI calculator
- **Business Model:** Enterprise tier ($2K-10K/month)
- **Priority:** P1 (High)
- **Effort:** 4 weeks

F21: Compliance Reporting

- SOC2 compliance reports
- GDPR audit trails
- Industry-specific reports (HIPAA, PCI-DSS)
- **Business Model:** Enterprise tier
- **Priority:** P1 (High for enterprise)
- **Effort:** 3 weeks

F22: LLM Observability Integration

- LangSmith integration
- Weights & Biases integration
- DataDog/New Relic plugins
- **Business Model:** Free (drives adoption)
- **Priority:** P1 (High)
- **Effort:** 4 weeks

F23: White-Glove Onboarding

- Custom metric development
- Agent architecture review
- Implementation support
- **Business Model:** Professional services ($5K-50K)
- **Priority:** P1 (High for enterprise)
- **Effort:** Ongoing

---

8. Technical Architecture

System Overview

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                          MMAP Architecture                                   │
├──────────────────────────────────────────────────────────────────────────────┤
│                                    │                                         │
│  ┌──────────────────────────────────────────────────────────────────────┐   │
│  │      User Interface Layer                                            │   │
│  ├──────────────────────┬──────────────────────┬──────────────────────┤   │
│  │  CLI Tool            │  Python SDK          │  Web Dashboard       │   │
│  │  (Typer)             │  (Core API)          │  (React/Plotly)      │   │
│  └──────────┬───────────┴──────────┬───────────┴──────────┬───────────┘   │
│             │                      │                      │                 │
│             └──────────────────────┴──────────────────────┘                 │
│                                    │                                         │
│  ┌──────────────────────────────────▼──────────────────────────────────┐   │
│  │      Core Evaluation Engine                                          │   │
│  ├──────────────────────────────────────────────────────────────────────┤   │
│  │  • AgentEvaluator (orchestrator)                                     │   │
│  │  • MetricRegistry (metric management)                                │   │
│  │  • TestLoader (test case loading)                                    │   │
│  │  • ResultAggregator (scoring)                                        │   │
│  └──────────┬──────────────────────────────────────────────────────────┘   │
│             │                                                               │
│  ┌──────────▼──────────┐   ┌──────────▼──────────┐                         │
│  │  Metrics Library    │   │  Integration Layer  │                         │
│  ├─────────────────────┤   ├─────────────────────┤                         │
│  │  • Layer 1 Metrics  │   │  • LangChain        │                         │
│  │  • Layer 2 Metrics  │   │  • LlamaIndex       │                         │
│  │  • Layer 3 Metrics  │   │  • AutoGen          │                         │
│  │  • Layer 4 Metrics  │   │  • Custom Agents    │                         │
│  │  • Layer 5 Metrics  │   │                     │                         │
│  │  • Custom Plugins   │   │                     │                         │
│  └─────────────────────┘   └─────────────────────┘                         │
│                                    │                                         │
│  ┌──────────────────────────────────────────────────────────────────────┐   │
│  │      Data & Storage Layer                                            │   │
│  ├──────────────────────────────────────────────────────────────────────┤   │
│  │  • Test Datasets (JSON/CSV)                                          │   │
│  │  • Evaluation Results (JSON/SQLite)                                  │   │
│  │  • Baselines (for regression)                                        │   │
│  │  • Benchmark Data                                                    │   │
│  └──────────────────────────────────────────────────────────────────────┘   │
│                                    │                                         │
```

```
└──────────────────────────────────────────────────────────────────────────────┘

        ┌──────────────────────────────────────────┐
        │  Optional SaaS Components        │
        │  (Premium Features Only)         │
        ├──────────────────────────────────────────┤
        │ • Web Application (FastAPI)      │
        │ • PostgreSQL Database            │
        │ • Redis Cache                    │
        │ • Background Workers (Celery)    │
        │ • Monitoring (Prometheus)        │
        │ • Authentication (Auth0)         │
        └──────────────────────────────────────────┘
```

Technology Stack

Core Framework (Open Source)

```yaml
Language: Python 3.8+
Dependencies:
  - dataclasses: Data structures
  - typing: Type hints
  - json: Data serialization
  - pathlib: File handling

Optional Dependencies:
  - pandas: Data manipulation
  - numpy: Numerical operations
  - scikit-learn: Statistical tests
  - plotly: Visualizations

Integrations:
  - langchain: LangChain support
  - llama-index: LlamaIndex support
  - openai: LLM evaluation
```

CLI Tool

```yaml
```

```
Framework: Typer
Features:
  - Command parsing
  - Progress bars
  - Colored output
  - Auto-completion
```

Web Dashboard (Open Source)

```yaml
Type: Static HTML + JavaScript
Libraries:
  - Plotly.js: Interactive charts
  - D3.js: Custom visualizations
  - Tailwind CSS: Styling

Output: Single HTML file (portable)
```

SaaS Platform (Premium)

```yaml
```

```
Backend:
  - FastAPI: API server
  - PostgreSQL: Primary database
  - Redis: Caching and queues
  - Celery: Background jobs
  - MinIO/S3: Object storage

Frontend:
  - React: UI framework
  - TypeScript: Type safety
  - Tailwind CSS: Styling
  - Recharts: Charts
  - React Query: Data fetching

Infrastructure:
  - Docker: Containerization
  - Kubernetes: Orchestration
  - GitHub Actions: CI/CD
  - Terraform: Infrastructure as code

Monitoring:
  - Prometheus: Metrics
  - Grafana: Dashboards
  - Sentry: Error tracking
  - Datadog: APM
```

Data Models

Core Models

```python
```

```
# Test Case
{
  "id": "test_001",
  "input": {
    "text": "...",
    "metadata": {}
  },
  "ground_truth": {
    "intent": "...",
    "decision": "...",
    "entities": {}
  },
  "tags": ["edge_case", "high_value"]
}

# Evaluation Result
{
  "evaluation_id": "eval_abc123",
  "timestamp": "2025-10-15T10:30:00Z",
  "agent_id": "refund_agent_v1.2",
  "overall_score": 0.92,
  "layers": [
    {
      "layer_number": 1,
      "layer_name": "Input/Output Validation",
      "score": 0.94,
      "status": "pass",
      "metrics": [...]
    }
  ],
  "critical_issues": [],
  "test_cases_count": 100,
  "duration_seconds": 45.2
}

# Metric Result
{
  "metric_name": "Intent Accuracy",
  "layer": 1,
  "score": 0.94,
  "threshold": 0.90,
  "passed": true,
  "details": {
    "correct": 94,
    "total": 100,
    "precision": 0.94
```

```
  }
}
```

API Design

Python SDK API

```python
# Basic usage
from mmap import AgentEvaluator
from mmap.metrics import IntentAccuracy, DecisionAccuracy

evaluator = AgentEvaluator(
    agent=my_agent_function,
    test_dataset="tests.json"
)

evaluator.add_metric(IntentAccuracy(threshold=0.90))
evaluator.add_metric(DecisionAccuracy(threshold=0.95))

report = evaluator.evaluate()
report.print_summary()
report.to_json("results.json")

# Advanced usage
from mmap import AgentEvaluator, BaseMetric

class CustomMetric(BaseMetric):
    def evaluate(self, output, truth):
        # Custom logic
        return EvaluationResult(...)

evaluator.add_metric(CustomMetric(), layer=4)

# Batch comparison
results = evaluator.compare(
    agents=[agent_v1, agent_v2],
    test_dataset="tests.json"
)
```

CLI API

```bash
```

```
# Initialize new evaluation
mmap init my-agent-eval

# Run evaluation
mmap run --config config.yaml --output report.json

# View results
mmap report results.json

# Compare versions
mmap compare agent_v1.json agent_v2.json

# Continuous monitoring
mmap monitor --agent my_agent --interval 1h
```

REST API (SaaS Only)

```yaml
POST /api/v1/evaluations
  - Create new evaluation
  - Request: evaluation config + test data
  - Response: evaluation_id

GET /api/v1/evaluations/{id}
  - Get evaluation results
  - Response: full evaluation report

GET /api/v1/evaluations
  - List all evaluations
  - Query params: agent_id, date_range, status

POST /api/v1/agents
  - Register agent for monitoring
  - Request: agent metadata

GET /api/v1/metrics
  - List available metrics
  - Response: metric catalog

POST /api/v1/benchmarks
  - Submit to benchmark
  - Request: evaluation results (anonymized)
```

Security & Privacy

Open Source

- No data collection by default
- All processing local
- Optional anonymous usage telemetry (opt-in)

SaaS Platform

- SOC2 Type II compliance
- Data encryption at rest (AES-256)
- Data encryption in transit (TLS 1.3)
- GDPR compliant (EU data residency)
- HIPAA compliance option (healthcare)
- Regular security audits
- Penetration testing

Scalability

Open Source

- Designed for single-machine execution
- Parallel test case evaluation (multiprocessing)
- Efficient memory usage (streaming large datasets)
- Target: Evaluate 10,000 test cases in <5 minutes

SaaS Platform

- Horizontal scaling (Kubernetes)
- Async job processing (Celery)
- Caching layer (Redis)
- CDN for static assets
- Target: 1,000 concurrent evaluations

---

9. Success Metrics

Product Metrics

Adoption Metrics (Open Source)

Metric	30 Days	90 Days	6 Months	12 Months
GitHub Stars	500	2,000	5,000	10,000
Weekly Downloads	100	500	2,000	5,000
Contributors	5	20	50	100
Forks	50	200	500	1,000
Community Posts	10	50	200	500

Usage Metrics

Metric	Target (Month 3)	Target (Month 12)
Active Projects	50	500
Evaluations Run	1,000	50,000
Total Test Cases	100,000	5M
Production Deployments	10	100

Engagement Metrics

Metric	Target
Documentation Page Views	5,000/month
Example Notebook Opens	1,000/month
Discord Active Users	500
Newsletter Subscribers	2,000

Business Metrics (SaaS)

Metric	Year 1	Year 2
Paying Customers	10	50
MRR	$10K	$100K
ARR	$120K	$1.2M
Churn Rate	<5%	<3%
NPS Score	50+	60+

Quality Metrics

Framework Quality

- Code coverage: >80%
- Type hint coverage: >95%
- Documentation coverage: 100%
- Average evaluation time: <30s for 100 test cases
- False positive rate: <5%
- False negative rate: <2%

User Experience

- Time to first evaluation: <10 minutes
- Success rate (new users): >90%
- Average setup time: <5 minutes
- Documentation clarity score: 8/10+

Impact Metrics

Value Delivered

- Production incidents prevented: Track via user surveys
- Cost savings: Estimate $50K-500K per major incident
- Time saved: vs manual review (estimate 10-50x faster)

Community Health

- Response time to issues: <24 hours
- PR merge time: <7 days
- Active maintainers: 3-5
- Monthly releases: 1-2

---

10. Go-to-Market Strategy

Launch Strategy

Phase 1: Stealth Build (Weeks 1-4)

- Build MVP in private
- Recruit 5 beta testers
- Iterate based on feedback

- Prepare launch materials

Phase 2: Public Launch (Week 5)

- Coordinated launch:
  - Product Hunt
  - Hacker News
  - Reddit (r/MachineLearning, r/LangChain)
  - LinkedIn post
  - Twitter announcement
- Goal: 500 stars in week 1

Phase 3: Content Blitz (Weeks 6-12)

- Publish 2 blog posts per week
- Create 2 YouTube tutorials
- Guest post on major ML blogs
- Podcast appearances
- Goal: 2,000 stars by month 3

Phase 4: Community Building (Months 4-6)

- Discord/Slack community launch
- Virtual meetups
- Conference talks
- Partnerships with agent frameworks
- Goal: 5,000 stars by month 6

Phase 5: Monetization (Months 7-12)

- Launch SaaS platform
- Enterprise outreach
- Consulting services
- Workshop/training programs
- Goal: $10K MRR by month 12

Marketing Channels

Owned Channels

1. **Blog** (SEO-optimized)
   - Weekly technical deep-dives
   - Case studies
   - Best practices guides
2. **YouTube Channel**
   - Tutorials
   - Live coding
   - Interviews with practitioners
3. **Newsletter**
   - Bi-weekly updates
   - Community highlights
   - New metrics and features
4. **Documentation Site**
   - Comprehensive guides
   - API reference
   - Examples and templates

Community Channels

1. **Discord/Slack**
   - Support channel
   - Feature discussions
   - Showcase projects
2. **GitHub Discussions**
   - Technical Q&A
   - Feature requests
   - Best practices
3. **Stack Overflow**
   - Answer agent evaluation questions
   - Build reputation

Paid Channels (Later)

1. **Conference Sponsorships**
   - AI Engineer Summit
   - PyData conferences
   - MLOps events
2. **Google Ads**
   - Target keywords: "agent evaluation", "LLM testing"
   - Budget: $2K-5K/month (Year 2)
3. **LinkedIn Ads**
   - Target ML engineers, AI companies
   - Budget: $3K-5K/month (Year 2)

Partnership Strategy

Framework Partners

- **LangChain**: Official evaluation integration
- **LlamaIndex**: Featured in documentation
- **AutoGen**: Example implementations
- **CrewAI**: Joint webinars

Observability Partners

- **LangSmith**: Native integration
- **Weights & Biases**: MLOps pipeline
- **DataDog**: Production monitoring

Cloud Partners

- **AWS**: Marketplace listing
- **Google Cloud**: Partner program
- **Azure**: Joint go-to-market

Pricing Strategy

Open Source Tier

- **Price**: Free forever

- **Features**:
  - Core framework
  - All basic metrics
  - CLI tool
  - Local dashboard
  - Community support

Team Tier (SaaS)

- **Price**: $199/month (5 users)
- **Features**:
  - Team workspace
  - Cloud evaluations
  - Continuous monitoring
  - Priority support
  - Advanced analytics

Enterprise Tier (SaaS)

- **Price**: $2,000-10,000/month
- **Features**:
  - Unlimited users
  - SSO/SAML
  - Custom metrics
  - Compliance reports
  - Dedicated support
  - SLA guarantees
  - Professional services

Professional Services

- **Implementation**: $10K-50K
- **Custom Development**: $200-300/hour
- **Training Workshops**: $5K-10K per session
- **Annual Support**: $50K-200K

11. Roadmap

Q4 2025: MVP & Launch

Milestone: Prove Product-Market Fit

October (Month 1):

- Week 1-2: Core framework development
- Week 3: Refund agent example
- Week 4: Documentation and polish

November (Month 2):

- Week 1: Beta testing with 5 users
- Week 2: Launch (Product Hunt, HN, Reddit)
- Week 3-4: Community engagement, bug fixes

December (Month 3):

- Weeks 1-4: Content marketing
- Add 10 more metrics
- Framework integrations (LangChain)
- **Target**: 2,000 stars, 50 active users

Q1 2026: Growth & Maturity

Milestone: 5,000 Stars, Production-Ready

January (Month 4):

- HTML dashboard
- LlamaIndex integration
- CI/CD plugins
- Conference talk applications

February (Month 5):

- Extended metrics library
- Batch evaluation

- Regression testing
- First enterprise pilot

March (Month 6):

- Community growth initiatives
- Partnership announcements
- Documentation expansion
- **Target**: 5,000 stars, 100 production users

Q2 2026: SaaS Platform

Milestone: Launch Premium Features

April (Month 7):

- SaaS infrastructure setup
- Web application MVP
- User authentication
- Team workspaces

May (Month 8):

- Continuous monitoring
- Cloud-hosted evaluations
- Billing integration
- Beta customers (5 companies)

June (Month 9):

- Advanced analytics
- Compliance reporting
- Integrations (LangSmith, W&B)
- **Target**: 10 paying customers, $10K MRR

Q3 2026: Scale & Enterprise

Milestone: Enterprise-Ready

July-September (Months 10-12):

- Enterprise features (SSO, RBAC)
- Professional services launch
- Sales team hiring
- Marketing automation
- **Target**: 30 paying customers, $50K MRR

2027 & Beyond

Year 2 Goals:

- 50+ enterprise customers
- $1.2M ARR
- 10,000 GitHub stars
- 500+ production deployments
- Industry standard for agent evaluation

Long-term Vision (3-5 years):

- Become the "MLflow for agents"
- Acquire 100+ enterprise customers
- $10M+ ARR
- Potential acquisition target ($50M-100M)

---

12. Open Questions & Risks

Open Questions

Product Questions

1. **Metric standardization**: How do we ensure metrics are comparable across different agent types?
2. **Benchmark datasets**: Should we create public benchmarks? How to incentivize contributions?
3. **Pricing model**: Should SaaS be per-user, per-evaluation, or hybrid?
4. **Integration depth**: How deep should framework integrations go?
5. **Multi-language support**: When to add JavaScript/TypeScript support?

Technical Questions

1. **Scalability limits**: What's the max evaluation size on single machine?
2. **Real-time evaluation**: Can we evaluate streaming agents?
3. **LLM-as-judge**: When to use LLM for evaluation vs rules?
4. **Data privacy**: How to handle sensitive test data in SaaS?

Business Questions

1. **Freemium conversion**: What % of open-source users will convert to paid?
2. **Enterprise sales**: Build sales team or stay product-led?
3. **Vertical focus**: Should we specialize in specific industries (finance, healthcare)?
4. **Competition**: What if OpenAI/Anthropic launch competing products?

Risks & Mitigations

Risk 1: Low Adoption

Risk: Open-source launch doesn't gain traction
Probability: Medium (30%)
Impact: High
Mitigation:

- Launch with compelling use case (refund agent)
- Multi-platform launch (PH, HN, Reddit simultaneously)
- Leverage 16 years ML experience for credibility
- Offer implementation consulting to drive adoption
- Have 3-5 beta users lined up for social proof

Risk 2: Framework Fragmentation

Risk: Different agent frameworks too different to standardize
Probability: Medium (40%)
Impact: High
Mitigation:

- Design abstraction layer for framework differences
- Start with most popular frameworks (LangChain, LlamaIndex)
- Allow custom adapters for niche frameworks
- Focus on common evaluation needs first

Risk 3: Competitive Response

Risk: Major players (OpenAI, LangChain, W&B) launch competing products
Probability: High (60% within 18 months)
Impact: High
Mitigation:

- Move fast on open-source to build community moat
- Focus on comprehensive evaluation (hard to replicate)
- Build deep integrations and partnerships
- Establish thought leadership early
- Consider acquisition as successful exit

Risk 4: Technical Complexity

Risk: Evaluation too complex for average user
Probability: Medium (35%)
Impact: Medium
Mitigation:

- Prioritize excellent documentation and examples
- Offer pre-configured evaluation templates
- Build helper tools (test dataset generator)
- Provide consulting for complex cases
- Create video tutorials and workshops

Risk 5: Monetization Challenges

Risk: Free users don't convert to paid
Probability: Medium (40%)
Impact: Medium
Mitigation:

- Make free tier genuinely valuable (builds trust)
- Clear value proposition for paid features
- Target enterprise with high willingness to pay
- Diversify revenue (SaaS + consulting + training)
- Freemium is customer acquisition, not primary revenue

Risk 6: Sustainability

Risk: Unsustainable to maintain open-source + build SaaS
Probability: Medium (35%)
Impact: High
Mitigation:

- Start with strong MVP, don't overcommit on features
- Build community of contributors early
- Generate consulting revenue to fund development
- Hire contractors for specific features
- Consider venture funding if traction strong

Risk 7: Regulatory Changes

Risk: New AI regulations make evaluation requirements more complex
Probability: Medium (50%)
Impact: Medium (could be positive!)
Mitigation:

- Monitor regulatory landscape (EU AI Act, etc.)
- Build compliance features proactively
- Position as solution for regulatory compliance
- Partner with legal/compliance firms
- Actually an opportunity if positioned correctly

Appendices

Appendix A: Competitor Analysis

(Detailed analysis of LangSmith, Confident AI, W&B, custom solutions)

Appendix B: Technical Specifications

(Detailed API specs, data models, architecture diagrams)

Appendix C: Market Research

(Survey results, interview transcripts, market size calculations)

Appendix D: Financial Projections

(Detailed revenue models, cost structure, break-even analysis)

Document Approval

Role	Name	Date	Signature
Product Owner	[Your Name]	[Date]	_____
Engineering Lead	[TBD]	[Date]	_____
Design Lead	[TBD]	[Date]	_____

End of PRD

This is a living document. Updates will be versioned and logged in the revision history.