WebPerf Scout — 完整需求规格说明书

> **版本**：1.1 **主题**：具备爬虫能力的 Web 软件性能分析与测评工具 **目标**：通过声明式配置，**自动发现页面并执行性能测评**，输出可量化、可对比、可追溯的专业报告 **适用对象**：参赛开发者、AI 代码生成器、评审专家

一、整体功能描述（Overall Functional Description）

1.1 产品定位

WebPerf Scout 是一个**集智能爬虫与真实浏览器性能测评于一体的自动化工具**。用户可通过两种方式指定待测范围：

- **手动模式**：显式列出关键页面 URL
- **爬虫模式**：提供种子 URL，工具自动发现站点内页面并进行性能巡检

核心理念：**“从入口出发，自动覆盖，精准测评”**。

1.2 核心价值主张

- ✅ **自动发现**：内置轻量级爬虫，从首页递归提取有效页面链接
- ✅ **可控遍历**：支持深度限制、域名白名单、路径排除、页面数量上限
- ✅ **真实性能采集**：基于 Playwright 在模拟设备/网络下采集 Web Vitals
- ✅ **专业报告**：输出含评分环、截图、指标卡、资源瀑布图的 HTML 报告
- ✅ **CI/CD 友好**：单命令运行，输出结构化 JSON + 静态 HTML

1.3 用户工作流（End-to-End Flow）

1. 编写配置：选择以下任一方式
   - 方式 A（手动）：在 `targets` 中列出 URL
   - 方式 B（爬虫）：配置 `crawler.seedUrl` 等参数
2. **执行命令**：`webperf-scout run --spec perf-spec.yaml`
3. 系统自动执行：
   - 若为爬虫模式 → 启动爬取引擎，生成 URL 列表
   - 为每个页面 × 设备组合启动性能测评
4. **生成报告**：`report.html` + `report.json`
5. **查看/集成**：人工审阅或接入自动化流程

1.4 能力边界（In Scope / Out of Scope）

类别	包含（✅ In Scope）	不包含（❌ Out of Scope）
爬虫能力	单域内链接发现、深度控制、路径过滤、去重	跨域爬取、登录后内容、JavaScript 动态路由（如 React Router 无 SSR）
页面类型	静态页、SSR SPA、含 `<a href>` 的页面	需交互触发跳转的页面（如点击按钮才出现新链接）
性能测评	LCP/FCP/CLS/TTI/TBT/resourceLoadTime	自定义 RUM 指标（除非通过 JS 表达式定义）
部署	CLI 工具、Docker 容器	SaaS 服务、多用户管理

1.5 典型使用场景

- **全站性能巡检**：给定首页 URL，自动测评全站 50 个核心页面在手机 4G 下的表现
- **上线前回归测试**：PR 合并前，爬取 staging 环境关键路径，确保无性能退化
- **竞品分析**：对公开网站（如电商首页）进行自动发现 + 性能打分

1.6 一句话定义

> **WebPerf Scout 是一个“爬虫 + 性能双引擎”工具，从一个入口 URL 出发，自动发现页面并完成专业级 Web 性能测评，输出可视化报告。**

二、后端功能清单（Backend Modules）

2.1 CLI 入口模块（`cli/main`）

- **职责**：接收命令，启动测评流程
- **输入**：`--spec <path>`（必填），`--output-dir`（默认 `./reports/`），`--verbose`
- **输出**：`report.html` + `report.json`，或错误退出（code=1）

- 行为：
  - 解析参数 → 校验 spec → 初始化日志 → 调度任务 → 生成报告
- **约束**：无交互；路径相对当前工作目录

2.2 Spec 解析与校验模块（`spec/validator`）

- **职责**：加载并验证 `perf-spec.yaml`
- **输入**：YAML/JSON 文件路径
- **输出**：标准化内部对象，或 ValidationError
- 校验规则：
  - `version: "1.0"`
  - `targets` 和 `crawler` **互斥且必选其一**
  - 若存在 `crawler`，则必须包含 `seedUrl`
  - `url` 必须为 HTTPS
  - `device` ∈ ["Desktop", Playwright 内置设备名]
  - `metrics` ∈ 预定义集合 或 自定义 JS 指标
  - `crawler.maxDepth`：默认 2，最大 5
  - `crawler.maxPages`：默认 30，最大 100
  - `crawler.includeDomains`：默认 `[new URL(seedUrl).hostname]`
  - `crawler.excludePatterns`：字符串数组（支持子串匹配）
- **约束**：错误信息需人类可读（如 “第5行：缺少 url”）

2.3 任务调度器（`engine/orchestrator`）

- **职责**：展开任务矩阵并并发执行
- **输入**：已校验 spec
- **输出**：`TaskResult[]`（每个 target × deviceProfile 一个结果）
- 行为：
  - 若 spec 含 `targets` → 直接使用
  - 若 spec 含 `crawler` → 调用 `discoverUrls(crawlerConfig)` 获取 URL 列表，并转换为 targets 格式（`name = pathname`）
  - 生成 K = N×M 个任务
  - 并发执行（默认 ≤ CPU 核数，上限 8）
  - 独立浏览器上下文（无状态污染）
  - 失败重试 ≤2 次（指数退避）
- **约束**：单任务总超时 ≤30 秒

2.4 Playwright 性能采集器（`engine/playwright-runner`）

- **职责**：在真实浏览器中加载页面并采集指标
- **输入**：`url`, `deviceProfile`（含 device/network/cpuThrottle）
- **输出**：`{ metrics, screenshotBase64, resources }`
- 行为：
  - 启动 Chromium via Playwright
  - 应用设备模拟（viewport/userAgent）
  - 应用网络节流（4G/3G/WiFi）
  - 通过 CDP 应用 CPU 节流（`Emulation.setCPUThrottlingRate`）
  - 导航到 URL
  - 注入指标脚本采集 Web Vitals
  - 截图 + 获取资源列表
- 指标采集方式：
  - `lcp/fcp/cls`：Google `web-vitals@4` 库
  - `tti`：实现 Lighthouse TTI 算法
  - `totalBlockingTime`：Long Tasks API
  - `resourceLoadTime`：`performance.navigation.loadEventEnd`
- 安全约束：
  - 禁止执行任意用户 JS
  - 禁止 `file://`
  - 爬取深度 = 1

2.5 性能评分与分析器（`engine/analyzer`）

- **职责**：计算综合评分并标记异常
- **输入**：原始指标值
- **输出**：`score: 0–100`, `alerts: []`
- 评分公式：

```
score = 100
  - max(0, lcp - 2500) * 0.03
  - max(0, fcp - 1800) * 0.055
  - max(0, tti - 3500) * 0.028
  - totalBlockingTime * 0.1
  - cls * 1000 * 0.15
score = clamp(round(score), 0, 100)
```

- 告警阈值：

- LCP > 2500 ms
- FCP > 1800 ms
- TTI > 3500 ms
- TBT > 200 ms
- CLS > 0.1

2.6 报告生成器（`report/generator`）

- **职责**：生成 HTML 和 JSON 报告
- **输入**：`TaskResult[]` + 时间戳
- 输出：
  - `report.html`：嵌入数据的静态页面
  - `report.json`：结构化数据
- HTML 生成：
  - 读取 `public/report-template.html`
  - 替换 `{{REPORT_DATA_JSON}}` 为序列化数据
- JSON 结构：

```
{
  "specVersion": "1.0",
  "runAt": "2025-12-11T14:00:00Z",
  "results": [
    {
      "targetName": "/",
      "deviceProfile": "Mobile_4G",
      "metrics": {
        "lcp": 2800,
        "fcp": 1200,
        "tti": 3100,
        "totalBlockingTime": 180,
        "cls": 0.05,
        "resourceLoadTime": 3200
      },
      "score": 72,
      "screenshotBase64": "...",
      "resources": [
        { "url": "https://example.com/", "duration": 3200, "startTime": 0 },
        { "url": "https://example.com/main.js", "duration": 800, "startTime": 400 }
      ]
    }
  ]
}
```

2.7 智能爬取模块（`crawler/discoverer`）

- **职责**：从种子 URL 自动发现待测页面集合
- 输入：

```
{
  seedUrl: string;
  maxDepth: number;       // 默认 2
  maxPages: number;       // 默认 30
  includeDomains: string[]; // 默认 [seedUrl 域名]
  excludePatterns: string[]; // 如 ["/admin", "/api"]
}
```

- **输出**：`Promise<{ url: string; name: string }[]>`（最多 `maxPages` 个）
- 行为逻辑：
  1. 初始化队列：`[{ url: seedUrl, depth: 0 }]`
  2. 初始化已访问集合（Set）
  3. While 队列非空 且 结果数 < maxPages：
     - 出队一个 URL
     - 若已访问 → 跳过
     - 若域名不在 `includeDomains` → 跳过
     - 若路径匹配任意 `excludePatterns`（子串匹配）→ 跳过

        - 使用 Playwright 加载该页面（快速模式，禁用图片/媒体）
        - 执行 JS 提取所有 <a[href]>：

          ```
          Array.from(document.querySelectorAll('a[href]'))
            .map(a => new URL(a.href, document.baseURI).href)
            .filter(href => href.startsWith('http'))
          ```

        - 对每个新链接：
            - 若 depth + 1 ≤ maxDepth → 入队
        - 将当前 URL 加入结果列表（`name = new URL(url).pathname`）
        - 标记为已访问
    4. 返回结果列表（按发现顺序）
- 安全约束：
    - 单页面加载超时 ≤ 10 秒
    - 总爬取时间 ≤ 2 分钟
    - 通过 `page.route()` 拦截非 text/html 资源（提升速度）
    - User-Agent 标识为 `WebPerfScout-Crawler/1.0`

三、前端功能清单（HTML 报告 UI）

3.1 整体设计

- **响应式**：适配桌面/手机浏览器
- **风格**：现代卡片式，绿色/橙色/红色表示性能状态
- **字体**：系统无衬线字体（如 `-apple-system, 'Segoe UI'`）

3.2 核心组件

(1) 标题与时间

- 显示："WebPerf Scout 性能报告"
- 显示运行时间（如 "2025年12月11日 14:00"）

(2) 综合评分环

- 圆形进度环，显示 0–100 分
- 颜色：
    - ≥90：绿色
    - 50–89：橙色
    - <50：红色

(3) 页面切换 Tab

- 每个 "页面 × 设备" 一个 Tab
- 默认激活第一个
- 点击切换内容（前端 JS 动态渲染）

(4) 指标卡片区

- 网格布局（响应式）
- 每卡含：指标名（LCP）、值（2.8 s）、状态色（根据阈值）

(5) 页面截图

- 显示 base64 截图
- 最大宽度 100%，圆角边框

(6) 资源瀑布图（简化）

- 列出前 20 个资源
- 每行：URL（截断）+ 耗时（如 "1.2 s"）
- 按 startTime 排序

3.3 交互逻辑

- 页面加载时读取内联 `REPORT_DATA` 变量
- 渲染默认页面内容
- Tab 切换时更新指标/截图/资源列表

- 自动应用颜色类（good/warning/bad）

3.4 数据绑定

- 后端注入：

```
<script>
  const REPORT_DATA = { /* report.json 内容 */ };
</script>
```

- 前端直接使用该变量渲染

四、非功能性要求

类别	要求
爬虫安全性	默认限深=2、限页=30；禁止跨域；自动排除 `/admin`、`/api` 等路径
可复现性	相同 spec + 相同环境 → 相同输出（含截图）
性能	爬取阶段：单页 ≤10s；测评阶段：单页×设备 ≤15s（4C8G 机器）
安全性	仅 HTTPS；无任意 JS 执行；无文件写入风险
日志	结构化 JSON 日志，输出 stdout，含 level/task/message
部署	提供 Dockerfile（`node:20-alpine`）
兼容性	支持 Chromium、Firefox（Playwright）

五、交付物（MVP）

1. **CLI 工具**：支持 `targets`（手动）和 `crawler`（自动）两种模式
2. **静态报告**：`report.html`（含所有被测页面性能数据）
3. **结构化数据**：`report.json`
4. 示例配置
    - `examples/manual-spec.yaml`（手动列表）
    - `examples/crawler-spec.yaml`（自动爬取）
5. **使用文档**：`README.md`（含安装、运行、CI 集成示例）

六、附录：Spec 示例

手动模式（manual-spec.yaml）

```
version: "1.0"
targets:
  - name: 首页
    url: https://example.com
  - name: 关于我们
    url: https://example.com/about
deviceProfiles:
  - device: "Desktop"
    network: "WiFi"
metrics: ["lcp", "fcp", "cls"]
```

爬虫模式（crawler-spec.yaml）

```
 version: "1.0"
crawler:
  seedUrl: https://example.com
  maxDepth: 2
  maxPages: 20
  excludePatterns: ["/admin", "/api", "/login"]
deviceProfiles:
  - device: "iPhone 12"
    network: "4G"
metrics: ["lcp", "fcp", "tti", "cls"]
```