**产品需求文档 (PRD)：工厂工时效率管理系统**

**1. 项目背景与目标**

作为精益管理和工业工程的核心，本项目旨在建立一个基于Web的工时效率评估系统 。通过规范化每日投入与产出工时的采集，实时核算各部门、各生产线的整体效率与异常损失，从而为精益改善提供数据支撑。

**2. 核心业务逻辑与公式**

系统所有的自动计算逻辑需严格遵循以下公式：

* **整体效率** = 产出工时 / 投入工时
* **异常工时率** = 异常工时 / 投入工时
* **产出工时** = $\sum(产品产出数量 \times 机型标准工时)$
* **投入工时** = (实际出勤人数 $\times$ 正常工作时数) + 加班时数 - 调整工时
  + 注：投入工时需区分直接人力与间接人力 。

+4

**3. 功能模块设计**

3.1 权限与账号管理

+3

系统采用两级权限体系：

* **管理员 (Admin)**：拥有所有配置表（人力、标工、异常类型）的维护权限及账号管理权限 。

+3

* **生产组长 (Lead)**：仅负责所属线别的每日工时日报录入与查询 。

**3.2 基础配置模块 (管理员专用)**

| **模块名称** | **关键字段 +2** | **逻辑说明** |
| --- | --- | --- |
| **部门人力配置** | 部门、线别、应出勤直接/间接人力、正常工时 | 校验“线别”唯一性，防止重复维护 。 |
| **标准工时 (ST)** | 产品系列、机型、直接/间接ST (HR/Unit)、适用线别 | 支持一个机型关联多个生产线 。 |
| **异常类型维护** | 异常类型名称（如：待料、机障、品质） | 用于日报下拉选择 。 |

3.3 每日工时日报模块

这是系统的核心输入端，采用“三步走”填写模式 ：

1. **初始化**：选择日期、部门、线别 。
2. **数据填报**：
   * **投入部分**：系统自动带出“应出勤人力”，组长填写加班与调整时数 。
   * **产出部分**：拉选机型（最多10个），输入各机型产出数量 。

+1

* + **异常部分**：拉选异常类型并输入损失工时 。

1. **即时反馈**：页面下方实时计算并呈现当天的“整体效率”与“异常工时率” 。

3.4 可视化看板与查询

+1

* **实时看板**：
  + 各部门/线别效率推移图（折线图） 。
  + 异常工时分类占比图（饼图） 。
* **报表查询**：支持按日期、部门、线别筛选明细，并提供 **Excel导出** 功能 。

**4. 技术架构与部署需求**

**4.1 技术栈**

* **后端**：Python 3.x + Flask 框架。
* **数据库**：SQLite 3 (轻量级，无需额外安装服务)。
* **前端**：原生 HTML5 + JavaScript (推荐使用 Chart.js 处理图表) + CSS (响应式布局)。

**4.2 数据库表结构 (Schema)**

* users: 存储账号、密码、角色。
* configs\_dept: 存储部门人力参数。
* configs\_st: 存储产品标工数据。
* configs\_abnormal: 存储异常分类。
* daily\_reports: 存储日报主表数据（日期、部门、线、总投入等）。
* daily\_output\_details: 存储日报关联的各机型产出明细。
* daily\_abnormal\_details: 存储日报关联的异常明细。

**4.3 生产环境部署 (Windows)**

1. **内网访问**：需配置 Flask 运行在 0.0.0.0，并开启 Windows 防火墙相应端口（如 5000）。
2. **生产服务器**：建议使用 Waitress 作为生产级 WSGI 服务器，以保证多用户同时访问的稳定性。
3. **便捷启动**：编写一个 .bat 批处理脚本，点击即可启动服务。

**5. 开发建议**

"请根据上述PRD，首先建立 SQLite 数据库模型。然后使用 Flask 编写 API 接口，实现基础配置表的 CRUD 功能。接着开发日报录入页面，要求在前端实现产出工时的动态计算。最后使用 Chart.js 开发实时看板。代码请保持简洁，不使用复杂的 node.js 依赖，全部使用原生 JS 或 CDN 引入。”