Arithmetic Practice Assistant
面向儿童的智能算术练习桌面应用

Development Team

2026 年 2 月 18 日

Outline

1. Sales Pitch
2. 开发与代码
3. 总结

产品定位

我们是什么?

一款面向 **K-12 儿童**的桌面算术练习工具:

- 可配置的四则运算 & 混合运算
- 支持括号、难度梯度
- **手写识别**输入
- 即时反馈 + 成绩统计

简洁·高效·有趣

核心卖点

简洁界面

儿童友好大字体 ($\geq$14pt)
Material Design 风格
中英文一键切换

流畅手写识别

画布手写 $\rightarrow$ 即时识别
多后端可选
本地/云端灵活切换

丰富历史记录

按姓名检索
正确率颜色标识
逐题详情回顾

方案对比：为什么不用 AI 出题？

	本项目（代码生成）	纯 AI 生成
正确性	✅ 100% 确定性正确	❌ 可能出错
成本	✅ 零边际成本	❌ 按 token 计费
延迟	✅ 毫秒级本地生成	❌ 网络延迟
离线可用	✅ 完全离线	❌ 依赖网络

结论

对于**结构化、规则明确**的算术题目，代码生成在正确性和成本上**完胜** AI。

手写识别：多后端策略

本地后端（零成本）

- `sklearn-svm` —内置 SVM 分类器
- `tesseract` —Tesseract OCR
- `paddle-ocr` —PaddleOCR

云端后端（高精度）

- Google Cloud Vision
- 百度 OCR
- 腾讯云 OCR

用户可在设置页面自由选择后端—离线环境用本地，追求精度用云端

历史记录功能

- 每次练习自动保存到 CSV
- 按学生姓名筛选
- 正确率颜色编码：
  - $\geq 80\%$：绿色
  - $\geq 60\%$：黄色
  - $< 60\%$：红色
- 点击查看逐题详情
- 便于家长/教师追踪学习进度

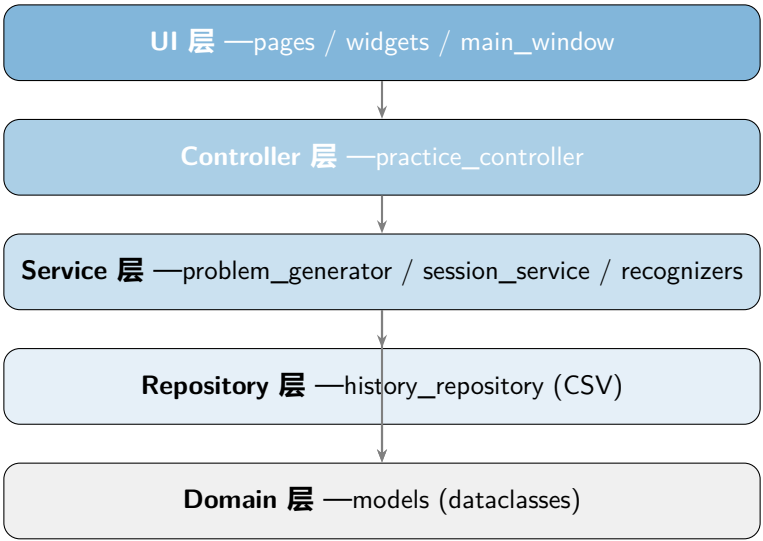
历史列表

Session A —95%

Session B —72%

Session C —48%

面向对象分层架构


UI 层 —pages / widgets / main_window
Controller 层 —practice_controller
Service 层 —problem_generator / session_service / recognizers
Repository 层 —history_repository (CSV)
Domain 层 —models (dataclasses)

项目目录结构

```
arithmetic_helper/
  main.py # 入口 + .env 加载
  app/
    domain/models.py # PracticeConfig, PracticeQuestion,
                           # AnswerRecord, SessionResult
    services/
      recognizer_backend.py # RecognizerBackend (ABC)
      handwriting_recognizer.py # sklearn SVM
      google_vision_recognizer.py
      baidu_ocr_recognizer.py
      tencent_ocr_recognizer.py
      tesseract_recognizer.py
      paddle_ocr_recognizer.py
      problem_generator.py # 题目生成引擎
      session_service.py # 会话生命周期管理
    repositories/
      history_repository.py # CSV 持久化
    controllers/
      practice_controller.py # 信号驱动的 UI 编排
    ui/
      main_window.py # 主窗口 + 依赖注入
      pages/ widgets/ # 页面 & 组件
    i18n/localizer.py # 国际化
```

关键设计模式

策略模式（Strategy）

`RecognizerBackend` 抽象基类
6 种具体实现可即插即用
运行时按用户选择切换

依赖注入（DI）

`MainWindow` 统一创建并注入
Service → Controller → Page
方便单元测试 mock

工厂模式（Factory）

`MainWindow._build_recognizer()`
根据 key 字符串实例化后端
内置缓存避免重复创建

信号/槽通信

PyQt 信号驱动 UI 更新
Controller 发射信号
Page 订阅响应

识别后端：抽象与扩展

```python
class RecognizerBackend(ABC):
    @abstractmethod
    def recognize(self, image: QImage) -> int | None:
        """识别手写内容，返回整数或 None"""

    @property
    @abstractmethod
    def name(self) -> str:
        """后端显示名称"""

    @property
    @abstractmethod
    def available(self) -> bool:
        """依赖/密钥是否就绪"""

    @staticmethod
    def _qimage_to_png_bytes(image: QImage) -> bytes | None:
        """共享工具方法：QImage -> PNG bytes"""
```

扩展新后端只需继承此类，实现 3 个抽象成员即可接入系统

通用 API-Key 接口

机制

1. `main.py` 启动时加载 `.env`
2. 写入 `os.environ`
3. 各 Recognizer 读取所需变量
4. `available` 属性检查密钥

```
# .env 示例
GOOGLE_VISION_API_KEY=xxx
BAIDU_API_KEY=xxx
BAIDU_SECRET_KEY=xxx
TENCENT_SECRET_ID=xxx
TENCENT_SECRET_KEY=xxx
```

用户只需修改 `.env` 文件即可切换 OCR 提供商—**零代码改动**

技术栈

层面	技术
GUI 框架	PyQt5 + qt-material 主题
语言	Python 3.10+
本地识别	scikit-learn (SVM)、Tesseract、PaddleOCR
云端识别	Google Vision、百度 OCR、腾讯云 OCR
数据持久化	CSV（轻量、可迁移）
国际化	自研 Localizer（zh_CN / en_US）
样式	Material Design + 自定义 CSS

可扩展性

当前支持

- 四则运算 + 混合运算
- 6 种识别后端
- 中文 / 英文界面
- CSV 历史存储

扩展方向

- 新题型（分数、方程 ...）
- 新识别后端（继承 ABC）
- 新语言（扩展字典）
- 新存储（SQLite / 云端）
- 自适应难度算法

分层解耦 + 抽象接口 = **低成本扩展**

总结

1. **产品层面**：简洁 UI + 手写识别 + 丰富历史 → 儿童友好的练习体验
2. **成本优势**：代码确定性生成题目，零边际成本，100% 正确
3. **工程质量**：五层解耦架构、策略/工厂/DI 设计模式
4. **灵活扩展**：抽象识别接口 + 通用 API-Key 管理 → 即插即用

Thank You!

Questions & Discussion