资深全栈工程师（偏后端）测试题

本次测试题要求：

1. 需要在现有 Perplexica 项目基础上进行二次开发。
    a. Perplexica 是一个开源的、基于 AI 的搜索引擎工具，使用 SearxNG 做元搜索，并通过 LLM（如 OpenAI、Ollama、Anthropic）来改进搜索结果的质量。项目包含前端（React）与后端（Node.js）组件，同时支持多种安装方式（Docker为主推荐）。现有功能包括多种 Focus 模式（如 Academic、YouTube、Wolfram Alpha）、API 支持、以及可插拔的模型后端。
2. 请在开发过程中保持代码整洁、结构清晰，并编写适当的说明文档和简单的测试。
3. 若需安装额外服务（如 Redis、Postgres），可使用 Docker 或本地安装方式自行配置。
4. 前端题需在 React 前端代码中实现新的功能组件或页面；后端题需在 Node.js 后端（可选用项目现有的结构或新增路由、模块）中实现相应逻辑；同时考察基础的数据库（Postgres）及缓存层（Redis）集成。
5. 务必保证所添加的功能可在本地开发环境下运行与验证。
6. 可使用 AI 辅助完成，重点在于最终代码质量与功能实现。
7. 请在收到题目后24小时后按提交说明提交作品

题目选择和提交说明：

1. **可选性**：应试者可根据时间与能力选择完成其中 1～3 道题目。全部完成者在测试中可能会加分。
2. **提交要求**：
    - 说明完成了哪些题目。
    - 提供代码仓库链接或打包文件。

- 提供简要的启动、测试步骤说明，以便验证功能。
- 提交录屏演示。

如需更多参考或信息，可查阅 Perplexica 项目文档 以及 MCP 官方文档。祝开发顺利，期待你的实现！

测试题列表

题目1（前端）：新增高级 Focus 模式管理界面

目标：在前端（React）中实现一个“Focus Mode 管理”页面，允许用户创建、编辑、删除自定义的 Focus 模式。

要求：

- 在 Perplexica 前端中新增一个路由与页面组件，比如 `/settings/focus-modes`。
- 页面中展示已有的 Focus 模式列表（从后端 API 获取或使用 mock 数据，您可在后端新增简单的内存存储接口）。
- 用户可在页面中新增自定义的 Focus 模式，每个模式包含以下字段：
  - 名称（必填）
  - 描述（可选）
  - API 来源（例如可配置 SearxNG 或其它后端服务的 endpoint）
- 支持模式的编辑与删除操作。
- 增加基本的表单验证（如名称必填且长度限制）。
- 异步加载和操作时增加 Loading 状态与错误处理（如创建失败的错误提示）。
- 确保页面 UI 简洁、易用，代码遵循 React 的最佳实践。

题目2（后端）：实现 Redis 缓存层优化搜索结果

目标：在 Node.js 后端为搜索结果加入 Redis 缓存，以提升重复查询的响应速度。

要求：

- 在后端（Node.js）中集成 Redis。例如在 `docker-compose` 中加入 Redis 服务，或提供在本地运行 Redis 的说明。
- 修改后端的搜索路由（例如 `/api/search`），在查询前先检查 Redis 缓存：
  - 如果存在对应 query 的缓存结果，直接返回缓存内容。
  - 如果缓存中无数据，则调用 SearxNG 获取搜索结果，并将结果存入 Redis（设置合理的过期时间，如 5 分钟）。
- 确保 Redis 缓存逻辑清晰、可维护，并在日志中能体现缓存命中与未命中状态。
- 提供简单的测试请求与结果验证说明。

题目3（后端）：整合 MCP 到 Perplexica，支持文字转图片功能

目标：将 MCP（Model Context Protocol）集成到 Perplexica，允许在用户与 AI 对话时，通过 MCP Client 调用一个自定义的 MCP Server 将文本转换为图片，并将该图片返回给用户（可在前端显示与下载）。

需求：

1. **实现 MCP Client**
   - 在 Perplexica 的后端（Node.js）中，新增或改造现有模块，充当 MCP Client，能够与一个 MCP Server 建立连接。
   - 当检测到用户请求“把回答转为图片”等类似操作时，后端会通过 MCP Client 调用服务器端的文字转图片功能。
   - 需保证网络通信流程符合 MCP 规范（可参考 MCP 官方文档）。
2. **实现 MCP Server**
   - 在本地或新开端口部署一个自定义的 MCP Server，该服务器暴露“文字转图片”的能力：
     - 接收客户端发送的文本请求
     - 将文本渲染为图片（可使用任意库，例如 Node.js 的 `canvas`、Python 的 `PIL` 等）
     - 返回对应的图片流或下载链接
   - 确保按照 MCP 协议对外暴露必要的元信息（如资源管理、工具定义等）。

- 可以使用 Docker 或本地方式部署，以方便测试与集成。

3. **触发逻辑**
   - 在 Perplexica 的查询或对话流程中（可能在回答阶段），如果用户表达了“请把回答转为图片”或“生成图片”等意图，后端在生成文本答案后，通过 MCP Client 调用 MCP Server，把文本转换为图片。
   - 将生成的图片在前端展示，并提供可下载链接。
4. **前端呈现**
   - 用户在 Perplexica 前端进行搜索或对话时，如果触发 MCP 图片生成操作，则在结果区或对话区插入一张图片预览。
   - 同时提供下载该图片的按钮或链接，让用户能保存到本地。
