Readme-cn

Before reading

1. 为了让展示的效果更加还原实际，我按照 Mock AI 界面开发了几乎一模一样的 Header 和 Footer 做为 dashboard 的 Header 和 Footer。同时，我使用了一个 iframe 来模拟 Header 中的其他 Item 的点击后效果。这样可以保证视觉上 Dashboard Page 仿佛集成在了现有的 Web 站点。
2. 里面的所有的评价指标、数据信息都是 mock 的，全部存放在 `/mock` 文件夹下，通过 API 请求在前端渲染。这些评价指标都是暂时想的，还有很大的修改空间。

How to run

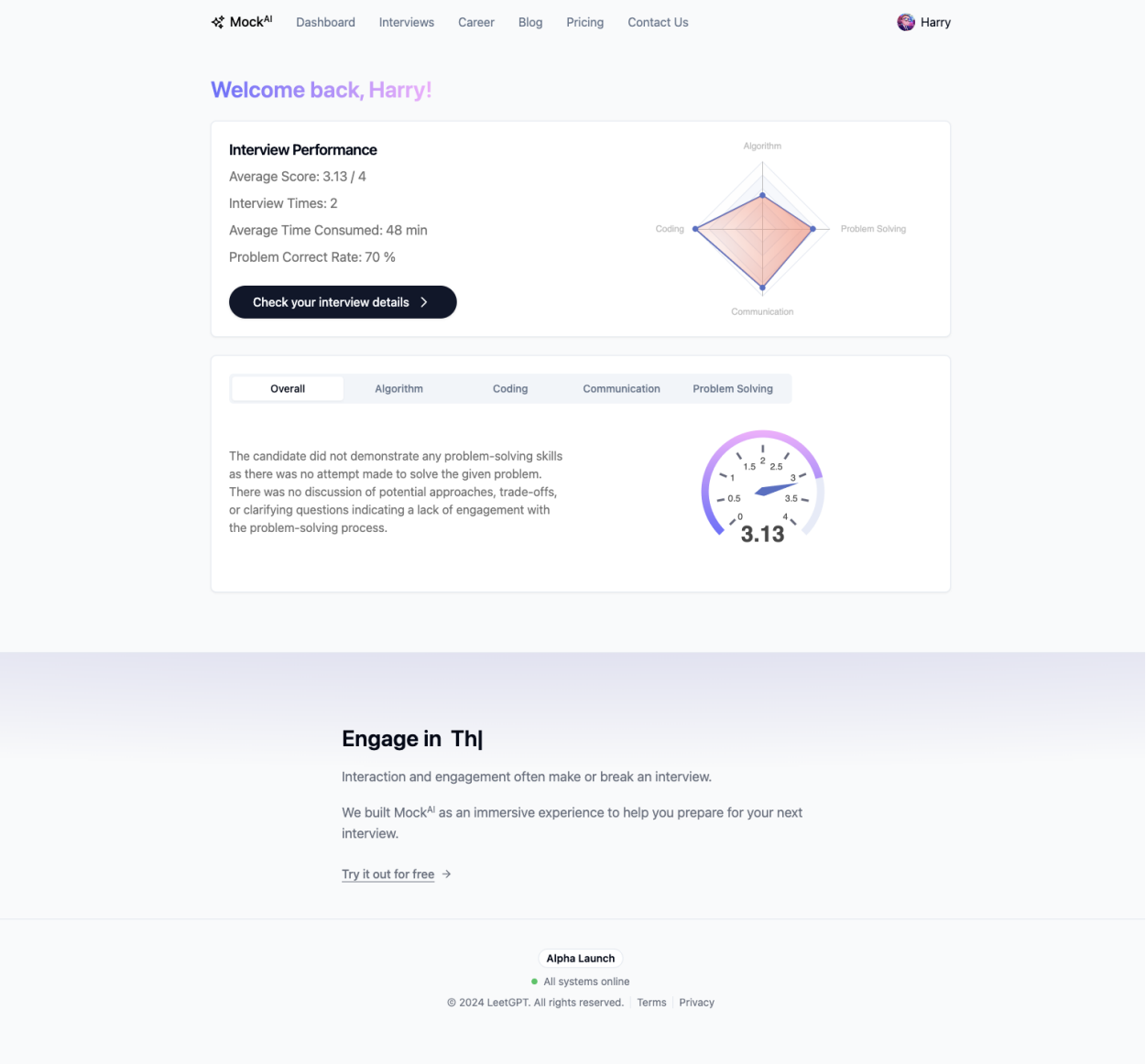
```
pnpm install

pnpm run dev
```

Design choices

1. 开发框架采用规定的 Nextjs + shadcn/ui + tailwind.css
2. 页面样式设计主要依据现有的 Web 站点设计风格
3. API 采用了 API Routes + Server Functions
4. 音频播放组件基于仓库 *wavesurfer*
5. 图表组件基于仓库 echarts
6. Code Editor 组件基于仓库 monaco-editor

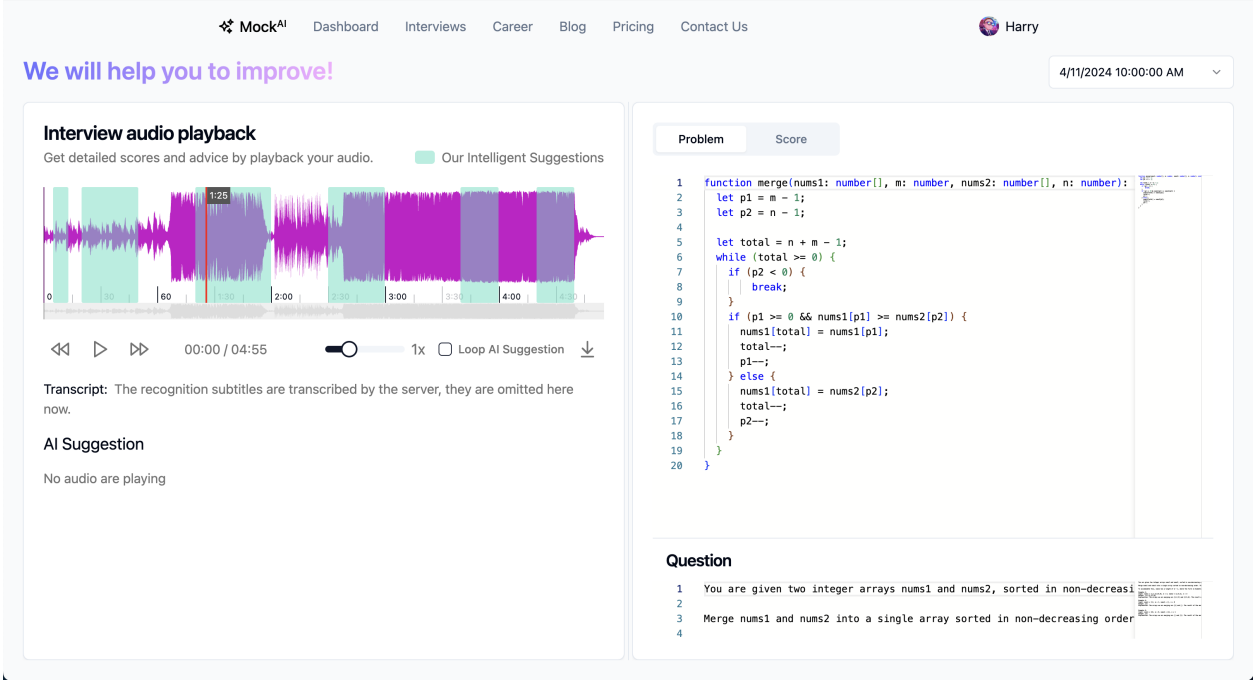
Dashboard features

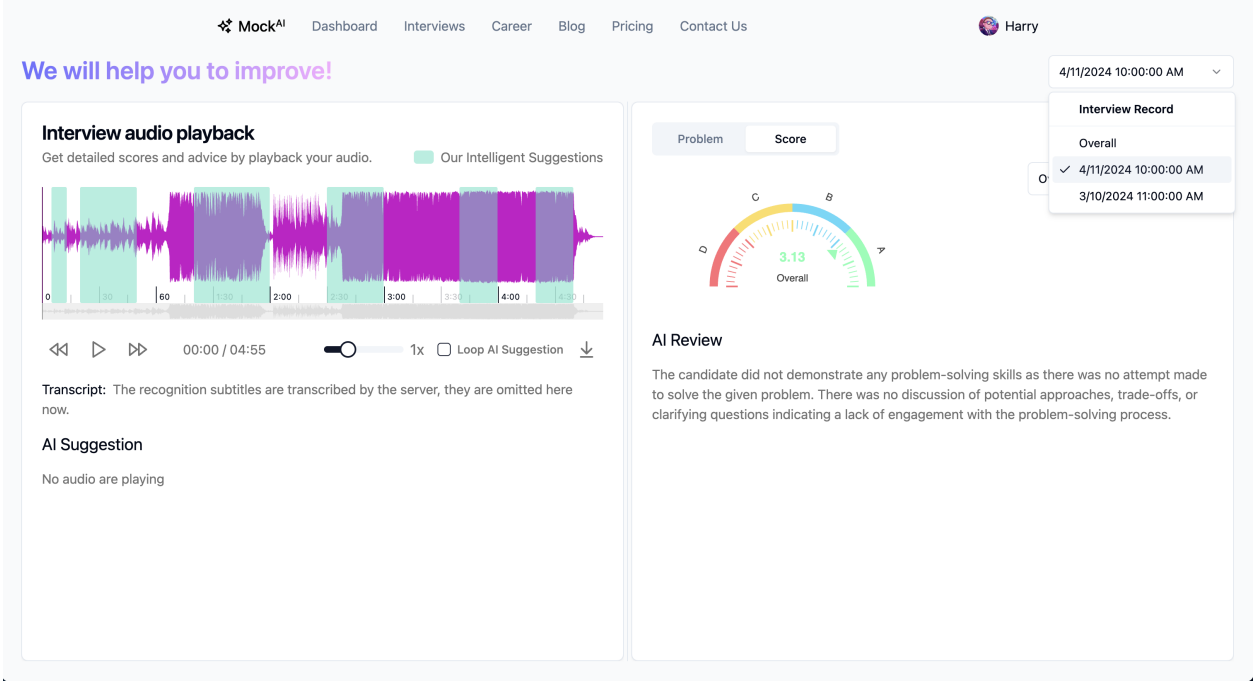
Main Page（对应路由 /dashboard）



1. Hearder，Footer，以及动画、间距、等一些全局的特征与现有网站保持一致。
2. 由于一个用户会存在若干个面试记录，所以主页面主要集中展现该用户面试的整体信息，包括但不限于：系统给出的平均分、面试次数、每次面试平均消耗时间、解题的正确率、以及 Algorithm、Coding、Communication、Problem Solving 四大维度的平均分和 AI 给出的总体反馈和建议（这些整体的评价指标都是暂时想的，不够还可以增加）。
3. 有一个 Check your interview detials 的按钮，点击之后，进入每一次面试的详情页面。

Interview Detail Page（对应路由 /dashboard/detail/[id]）





首先一些全局的特征，还是与现有网站保持一致。同时界面支持左右板块的大小滑动。Problem Tab 区域，也支持 Code 和 Question 板块的上下滑动。

在页面的右上方，有一个 Selection 下拉框，用于选择用户需要查看的面试（因为一个用户会有许多面试记录，需要有一个下拉框切换），默认值是最新的一次面试记录。

然后详情页分为几个模块：

录音 + AI 分析模块

1. 可视化音波，支持播放/暂停、快进/回退、倍速、下载。也允许拖动、放大/缩小、点击音轨任意位置播放等功能。
2. 绿色部分为系统分析出来的 AI Suggestion 部分（这里 mock 几组时间段的数据）。当音轨进行到绿色方块部分的时候，下面的 AI Suggestion 会显示出 Suggestion。如果用户点击 Loop AI Suggestion 的单选框，则会重复播放当前绿色块的部分。
3. 预留了音频转录为文本的设计位置。音频转录为文本的功能实现为从服务端将音频传送至类似 Google Cloud Speech-to-Text 的语音识别服务中，转录成文本，再传给前端。

题目展示模块

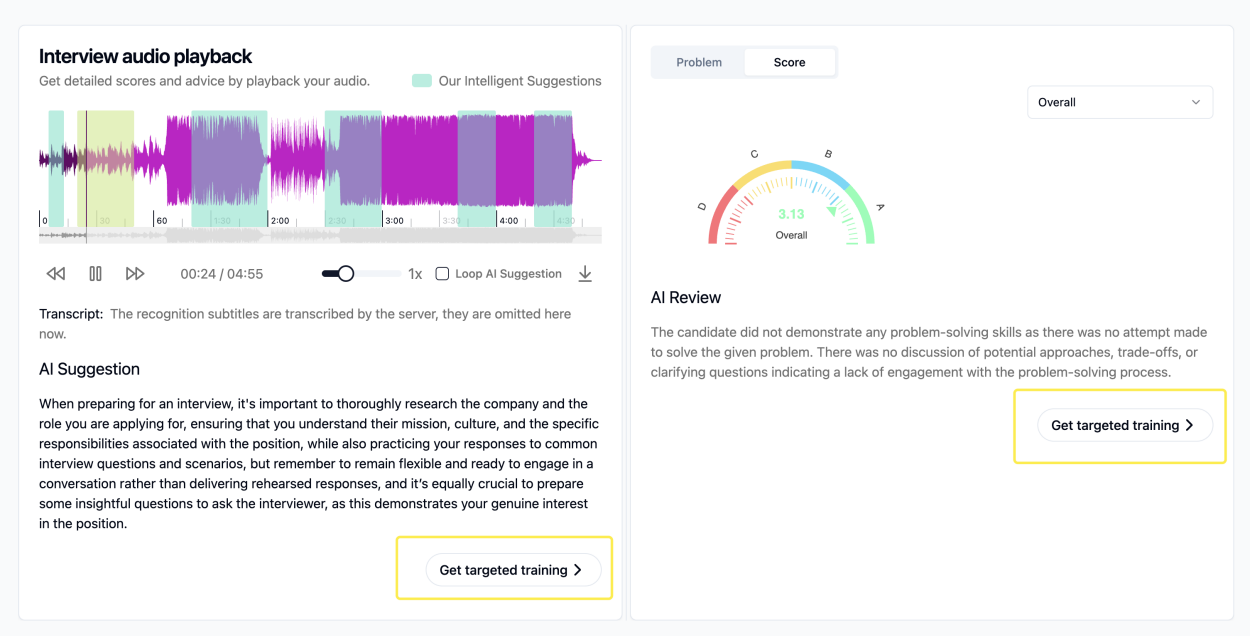
右半边的顶部的 tab 如果切换到 Problem， 则进入到题目展示模块。

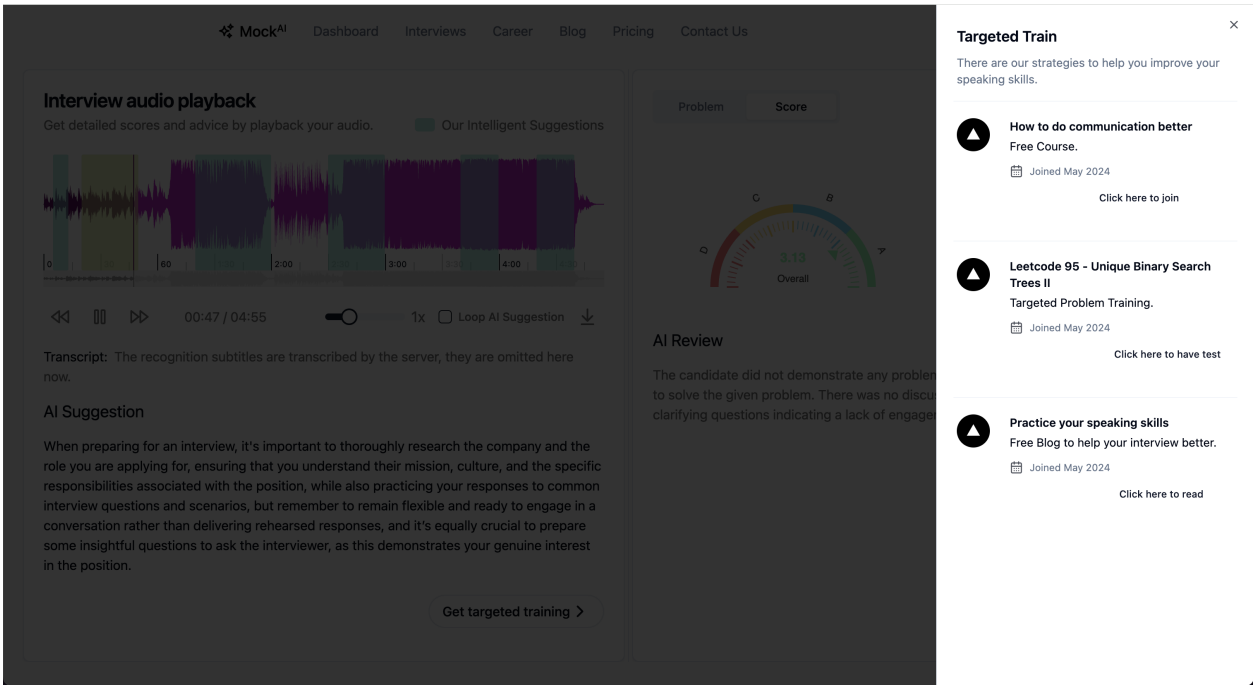
用户如果只听音频，而不能在当前的 page 看到对应面试的题目和自己的答案，那么将会很不直观和不方便操作。所以这里设计了一个题目展示模块。由于不确定原本站点的面试界面是基于什么库实现的，所以这里暂时使用了 monaco-editor 替代。

指标展示模块

右半边的顶部的 tab 如果切换到 Score， 则进入到指标展示模块。这里会显示这一次面试，AI 给出的Algorithm、Coding、Communication、Problem Solving 四大维度的分数和 AI 给出的反馈和建议。

One more thing: Targeted Train





这是一个要求之外的设计。数据都是 mock 的。

触发时机：

1. 在音频生成 AI Suggestion 的时候，会显示出按钮
2. 在 Score Tab 的时候，会显示出按钮

该功能设计的主要目的：

针对 AI Suggestion 和系统给用户面试的打分，针对性的推荐一系列的题目、课程或者文章，供用户更深入的提升自己的能力。同时，这也是今后我们业务拓展的一个可能的思路，可以进行付费，或者与第三方相关平台合作等等。

Q&A

这里主要回答一些 Assignment 文档里的一些问题。

Q：While actual recordings will not be provided due to security constraints, you should outline how the application would retrieve and process this data.

虽然没有提供音频数据，但是我使用了两首 wav 格式的歌来模拟面试录音。并且实现了使用 api routes 来获取和下载音频。代码在 /src/pages/api 里面。

其中检索和处理音频数据的主要逻辑是：处理请求和参数 → 从数据库读取音频文件（这里模拟成从项目的文件夹读取文件）→ 处理音频文件(包括错误处理) → 返回数据

这里特别注意的是，因为 assignment 里说录音长达一小时。所以可能会出现由于音频太大，需要很长加载时间，导致用户无法一开始播放就随心所欲的调整音频的位置（因为可能还没加载完毕）。所以这里需要实现范围请求，即从指定的范围开始加载文件，确保音频能在任意时长位置都即时播放。

简单的代码如下：

```
if (range) {
  const parts = range.replace(/bytes=/, "").split("-");
  const start = parseInt(parts[0], 10);
  const end = parts[1] ? parseInt(parts[1], 10) : fileSize - 1;
  const chunksize = end - start + 1;
  const file = fs.createReadStream(filePath, { start, end });
  const head = {
    "Content-Range": `bytes ${start}-${end}/${fileSize}`,
    "Accept-Ranges": "bytes",
    "Content-Length": chunksize,
    "Content-Type": "audio/wav",
  };
  res.writeHead(206, head);
  file.pipe(res);
```

```
    res.status(200);
  } else {
    const head = {
      "Content-Length": fileSize,
      "Content-Type": "audio/wav",
    };
    res.writeHead(200, head);
    fs.createReadStream(filePath).pipe(res);
  }
```

详细实现请参看 /src/pages/api/get-audio/index.ts 代码。

Q: Security Consideration: While you won't be working with actual audio files, include a brief description of how data security and privacy would be handled when fetching and displaying data.

回答这个问题需要从几个角度。

首先是技术层面：

1. 确保所有传输的音频数据使用 HTTPS 进行加密传输，这可以防止数据在传输过程中被截获。
2. 使用 POST 请求向 node 端请求数据。
3. 确保向数据库请求的 api 是由 nextjs 的 node 服务端发出的，而不是由前端直接去请求。这样可以隐藏真实的请求 url。
4. 确保对服务器上存储的音频数据进行加密，确保即使数据被非法访问，也无法被轻易解读。

其次是应用层面：

1. 确保所有访问音频数据的请求都经过用户身份验证（即登录）。原站点已经实现了用户登录和鉴权的功能。
2. 音频的权限管理，我们可以确保音频的权限是基于用户级别实现细粒度的访问控制。例如，只允许音频的上传者或有授权用户才能访问音频数据。
3. 埋点和监控，记录和监控对音频数据的访问，以便在出现安全问题时进行追踪和响应。

最后：

我们也应当告知用户我们的隐私政策。向用户明确说明应用如何收集、使用和存储音频数据。

The end

如果您有任何问题或建议，请随时告诉我！ 您的反馈会对我很有帮助。

非常感谢！

Haowei Xiong

harryxiong24@gmail.com | haoweix3@uci.edu