参赛作品设计说明书

作品名称：PresentationGen-PPT 智能生成

完成时间：2024/5/30

目录

一、 简介

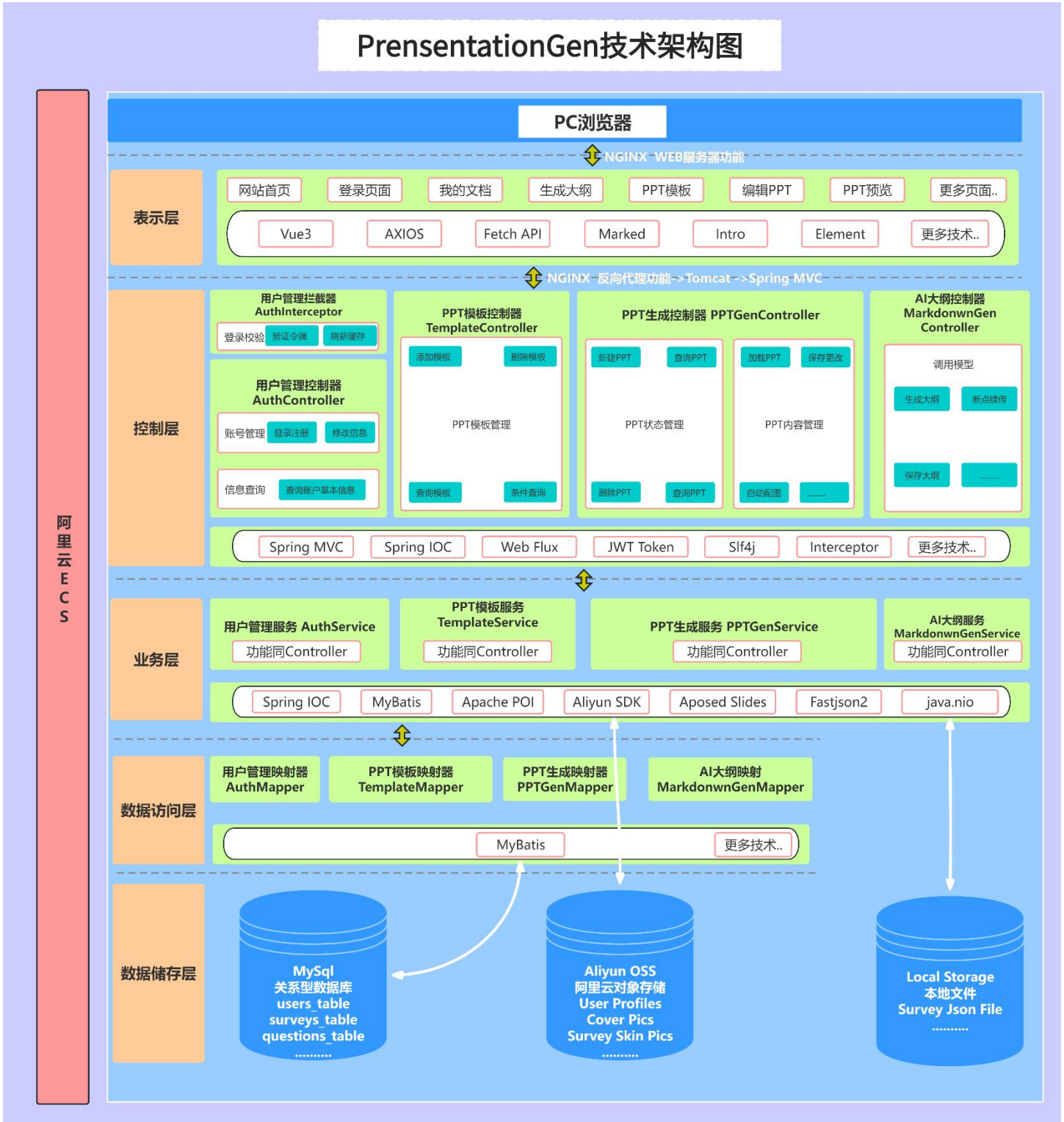
在现代工作和学习中，演示文稿（PPT）已经成为了沟通和展示思想的重要工具。然而，制作 PPT 往往需要花费大量的时间和精力，尤其是对于那些不擅长设计的人来说。为了解决这一难题，我们的应用——PresentationGen，它能够帮助用户轻松、快速地生成高质量的 PPT 演示。

PresentationGen 使用强大的文心一言语言模型，能够智能生成 Markdown 格式的大纲。无论用户要进行什么样的演示，只需简单输入用户的想法和主题，PresentationGen 就能够为用户生成清晰而完整的大纲，让用户可以更轻松地规划演示内容。并且通过 PresentationGen，用户不再需要手动将大纲转换为 PPT，因为它会自动将大纲解析转换为精美的 PPT 演示。这样，用户就可以节省大量的时间和精力，专注于演示内容的准备和完善。除此之外，应用还配备了丰富的预设模板和智能配图功能，让用户的演示更具吸引力和专业性。用户可以根据自己的喜好和演示内容选择合适的模板和配图，让用户的演示更加个性化和引人注目。除了演示制作功能之外，PresentationGen 还提供了完善和安全的用户管理功能，包括头像、密码、手机号设置等。同时，它还会记录用户的生成历史，方便用户随时查阅和管理自己的演示文稿。除了以上核心功能之外，PresentationGen 还提供了丰富的模板库、优秀的 PPT 编辑界面、流畅的打字机效果生成大纲页面，以及引人入胜的首页设计等特色功能，让用户的演示更加出彩和吸引人。

PresentationGen 采用了先进的前端和后端技术，包括 Vue.js、Spring Boot、MyBatis 等，保证了应用的流畅性、稳定性和安全性。同时，结合了第三方服务如 Aliyun OSS 和 Aliyun SMS，为用户提供了更全面的服务和保障。

在 PresentationGen 的帮助下，制作 PPT 演示将变得轻松而高效。

二、 设计原理


PrensentationGen技术架构图
阿里云ECS
PC浏览器
NGINX -WEB服务器功能
表示层
网站首页
登录页面
我的文档
生成大纲
PPT模板
编辑PPT
PPT预览
更多页面..
Vue3
AXIOS
Fetch API
Marked
Intro
Element
更多技术..
NGINX -反向代理功能->Tomcat ->Spring MVC
控制层
用户管理拦截器 AuthInterceptor
登录校验
验证令牌
刷新缓存
用户管理控制器 AuthController
账号管理
登录注册
修改信息
信息查询
查询账户基本信息
PPT模板控制器 TemplateController
添加模板
删除模板
PPT模板管理
查询模板
条件查询
PPT生成控制器 PPTGenController
新建PPT
查询PPT
PPT状态管理
删除PPT
查询PPT
加载PPT
保存更改
PPT内容管理
自动配图
AI大纲控制器 MarkdonwnGen Controller
调用模型
生成大纲
断点续传
保存大纲
Spring MVC
Spring IOC
Web Flux
JWT Token
Slf4j
Interceptor
更多技术..
业务层
用户管理服务 AuthService
功能同Controller
PPT模板服务 TemplateService
功能同Controller
PPT生成服务 PPTGenService
功能同Controller
AI大纲服务 MarkdonwnGenService
功能同Controller
Spring IOC
MyBatis
Apache POI
Aliyun SDK
Aposed Slides
Fastjson2
java.nio
数据访问层
用户管理映射器 AuthMapper
PPT模板映射器 TemplateMapper
PPT生成映射器 PPTGenMapper
AI大纲映射 MarkdonwnGenMapper
MyBatis
更多技术..
数据储存层
MySql
关系型数据库
users_table
surveys_table
questions_table
.....
Aliyun OSS
阿里云对象存储
User Profiles
Cover Pics
Survey Skin Pics
.....
Local Storage
本地文件
Survey Json File
.....

(一) 系统架构图

（二） 技术路线

1. 前端

- **Vue.js & Vue-Router**: 构建单页面应用程序 (SPA)，使用 Vue-Router 管理路由以支持用户界面的多页面和状态管理。
- **Element UI**: 利用 Element UI 提供现成的 UI 组件如输入框、按钮、选择器等，加速开发并保持界面一致性。
- **Axios & Fetch API**: 用于发起 HTTP 请求，与后端服务进行交互。Axios 提供更丰富的 API 和错误处理功能。
- **Marked.js**: 将 Markdown 文本解析为 HTML，用于大纲的展示。
- **Intro.js：**提供友好的新手提示
- **SHA-256 & JWT Token**: 用于安全处理用户认证。SHA-256 用于密码散列，JWT 用于管理用户会话。
- **NPM & ESLint**: NPM 用于管理前端依赖，ESLint 用于保证代码质量。

2. 后端

- **Spring Boot**: 作为后端主框架，用于快速开发企业级应用。
- **MyBatis**: 数据库 ORM 工具，简化数据库操作。
- **WebClient**:非阻塞 HTTP 客户端，用于流式传输和大模型接口的文本流和下载图片平台的插图。
- **Interceptor**: 使用 Spring Boot 的 Interceptor 进行请求拦截，实现如 jwt 身份验证和日志记录等功能。
- **Fastjson2：**用解析和保存 json 格式的 PPTRecord 数据
- **Apache POI&Aposed Slides**: 用于处理 PPT 生成，支持从预设模板中填充数据。
- **Maven**: 用于项目管理和构建，自动化处理项目依赖。
- **SLF4J**: 为日志框架接口，与具体实现（如 logback）结合，提供日志管理。

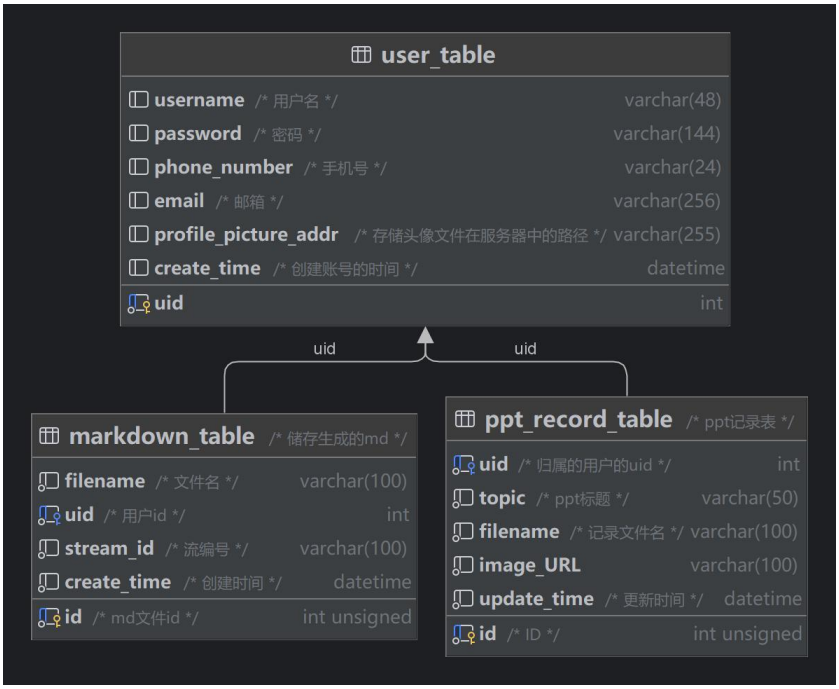
3. 第三方服务

- **Aliyun OSS**: 用于存储用户生成的 PPT 文件和其他网站静态资源。
- **Aliyun SMS**: 用于发送短信验证码，支持用户的手机验证功能。
- **Git**: Git 为版本控制工具。
- **Unsplash:** 作为无版权图片提供商

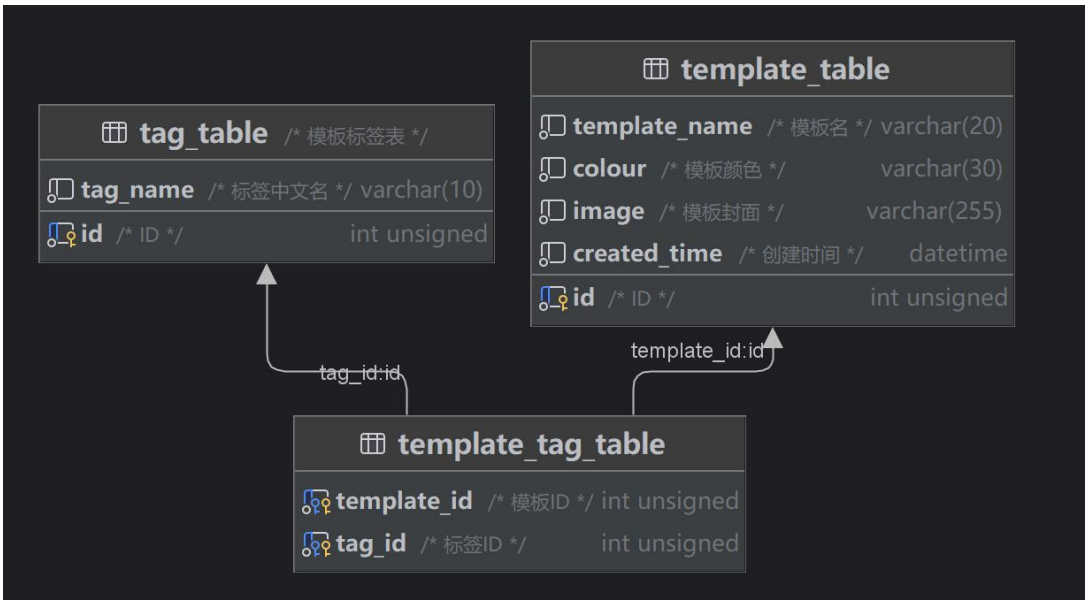
- **NGINX**：作为反向代理服务器，提高网站的加载速度和安全性，同时处理静态资源。

(三)　数据库设计

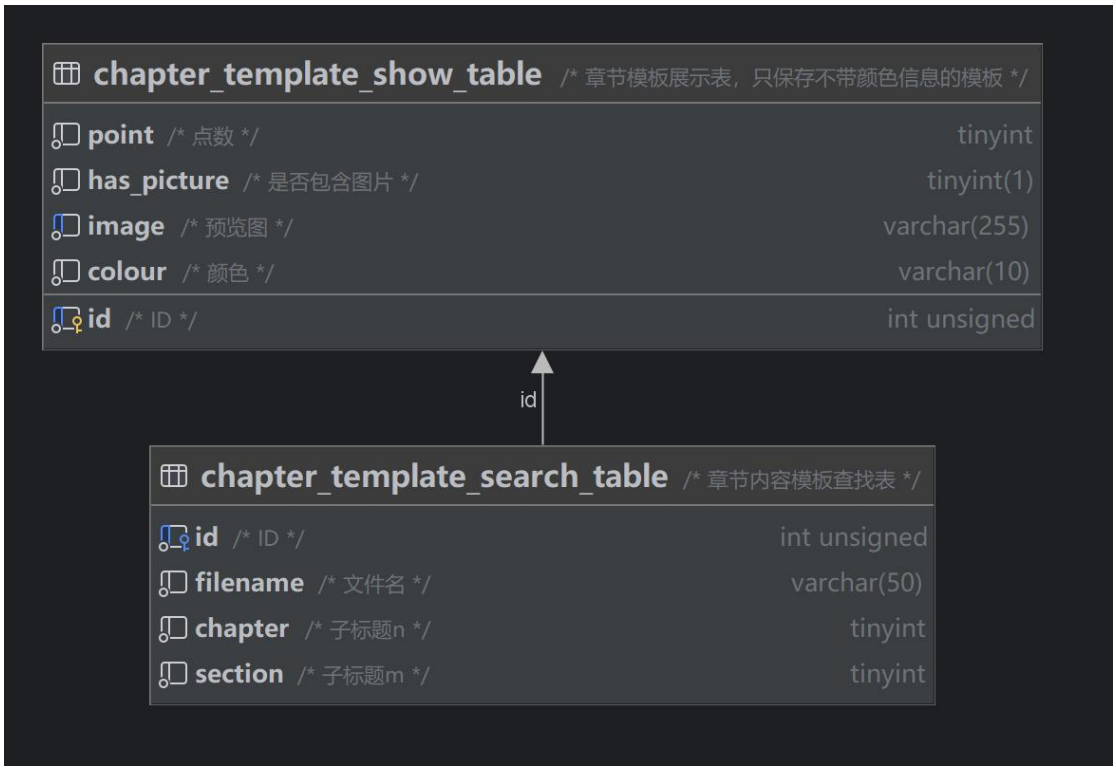
1. PPT 模板(template_table)与标签表(tag_table)（多对多关系，用中间表维护）



2. 用户和有关生成记录（大纲和 PPT 生成）表

3. 

4. 章节模板前端展示表(chapter_templates_show_table)和后端查询表(chapter_templates_search_table)



(四)　数据结构设计

核心数据结构：PPT 记录，格式化大纲，模板数组

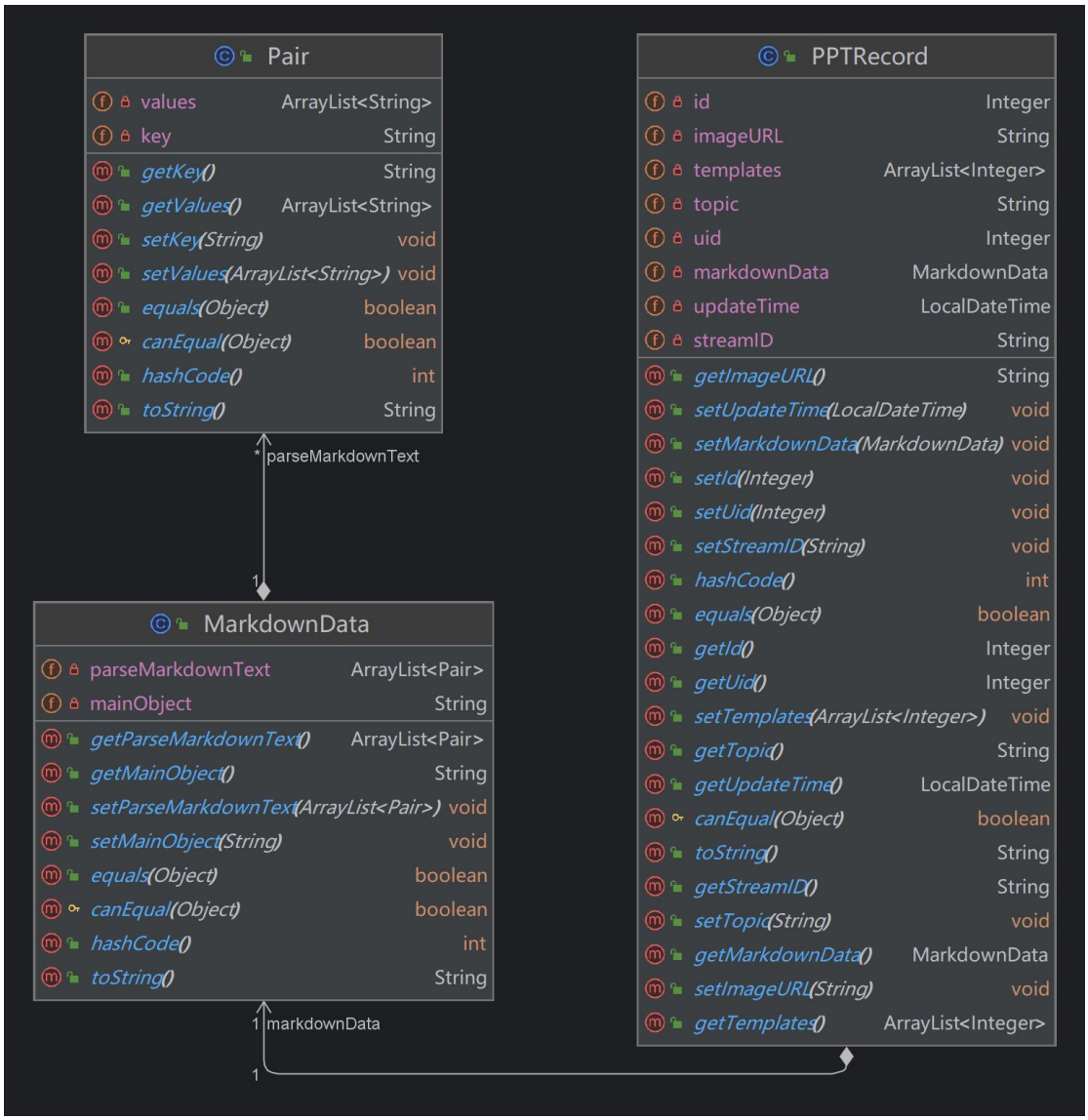


图　三个主要数据结构的依赖关系

1. PPTRecord

PPTRecord 用于记录一条保存过的 PPT，其中主要包含了对所属用户的标识 uid，保存了 PPT 文本大纲信息的 markdownData，保存了 PPT 模板选择信息的

templates。

Field Name	Type	Description
id	Integer	PPT ID
uid	Integer	用户编号 uid
topic	String	主题
imageURL	String	保存图片 url
updateTime	LocalDateTime	更新时间
streamID	String	流编号，未保存的 PPT 的临时编号
markdownData	MarkdownData	保留大纲信息
templates	ArrayList<Integer>	保留模板信息

2. MarkdownData 和 Pair

MarkdownData 保存了 PPT 文本大纲信息，包括了 mainObject，配图用的关键词，和 parseMarkdownText。parseMarkdownText 展开后的数据结构是 ArrayList<pair<String, ArrayList<String>>>,实际意义如下：

```
"parseMarkdownText": [
  {
    "key": "主标题",
    "values": [
      "副标题"
    ]
  },
  {
    "key": "第一章标题",
    "values": [
      "第一章副标题",
     "第一章第一节标题",
     "第一章第一节内容",
      "第一章第二节标题",
     "第一章第二节内容",
      "第一章第三节标题",
     "第一章第三节内容",
      "第一章第四节标题"
     "第一章第四节内容",
    ]
  },
```

```
        {
          "key": "第二章标题",
          "values": [
            "第二章副标题",
            "第二章第一节标题",
            ……
          ]
        }……..
    }
```

示例如图：


```
    "markdownData": {
      "parseMarkdownText": [
        {
          "key": "跨域问题详解",
          "values": [
            "原理、挑战与解决方案"
          ]
        },
        {
          "key": "1. 跨域问题的背景与定义",
          "values": [
            "了解跨域访问的限制",
            "(1) 跨域问题的基本概念",
            " **跨域定义**：当浏览器从一个域名的网页去请求另一个域名的资源时，由于安全策略限制，浏览器会阻止这样的请求，这就是跨域问题。\n **同源策略**：浏览器安全策略的核心，要求协议、域名和端口三者必须完全一致，否则即视为跨域。\n **跨域问题的普遍性**：在Web开发中，跨域问题普遍存在，尤其在前后端分离、微服务架构等场景中更为常见。",
            "(2) 跨域问题的历史与发展",
            " **早期Web开发**：由于Web应用规模较小，跨域问题并不突出。\n **现代Web应用**：随着Web应用复杂度的提升，跨域问题逐渐成为开发中的一大挑战。\n **跨域解决方案的演进**：从JSONP到CORS，再到现代的代理服务器方案，跨域解决方案不断演进。",
            "(3) 跨域问题的分类与示例",
            " **简单请求与预检请求**：根据HTTP请求类型和头部字段的不同，跨域请求可分为简单请求和预检请求。\n **常见跨域场景**：如API接口调用、图片加载、字体文件加载等。\n **跨域问题的具体表现**：如浏览器控制台中的错误提示、网络请求的失败等。",
            "(4) 跨域问题的安全性考量",
            " **跨域攻击的风险**：跨域问题可能导致数据泄露、恶意脚本注入等安全风险。\n **安全策略的重要性**：同源策略等安全策略是保护Web应用安全的重要手段。\n **跨域解决方案的安全性评估**：在解决跨域问题时，需充分评估各种方案的安全性。"
          ]
        },
```


字段表：

MarkdownData

Field Name	Type	Description
parseMarkdownText	ArrayList<Pair>	解析后的 ppt 结构化文本
mainObject	String	解析后 ppt 的中心对象。用于配图

Pair

Field Name	Type	Description
key	String	键
values	ArrayList<String>	值列表

(五)　核心模块设计

2. 用户管理模块 Auth

- **功能**: 处理用户注册、登录、信息更新（头像、密码、手机号）和用户验证。
- **技术栈**：
    - **前端**：Vue.js, Axios, SHA-256 for password hashing
    - **后端**：MyBatis 操作数据库, JWT for authentication 和 interceptor 用于用户认证, AliyunSMS 发送验证码 ,AliyunOSS 储存头像
    - **数据库**：用户信息存储

3. 大纲和文本解析模块

- **功能**：使用语言模型生成 Markdown 格式的大纲，解析大纲生成的文本。
- **技术栈**：
    - **前端**：Vue.js, Marked.js,Fetch Api
    - **后端**：WebClient , WebFlux 用于传输异步流式数据
    - **工具**：文心一言 JAVA SDK

4. PPT 自动生成模块

- **功能**: 根据解析的大纲和用户选择的模板及图片初始化，保存和生成 PPT。
- **技术栈**:
    - **前端**：使用 element-ui 构建编辑交互界面，调用微软的 office 在线预览框架实现 PPT 在线预览和放映
    - **后端**：使用 Apache POI 处理 PPT 文件文本替换，Aposed Slides 进行 PPT 合并，Fastjson2 保存和加载 PPT 记录，WebClient 请求外部图片服务。
    - **工具**：使用阿里云 OSS 储存 PPTX 文件
- **第三方服务**: Aliyun OSS 用于存储生成的 PPT 文件和静态资源。

三、 创新点

1. 后端大纲生成流式传输(MarkdownGenService#sendRequest)

使用 WebClient 和反应式编程模式 Flux 来处理异步 HTTP 请求。该方法的目的是向特定的 URI 发送 POST 请求，并处理返回的流式响应数据。并且判断本次请求是否得到了完整的回复，若不完整则进行下一次请求并拼接在流尾部。

```
public Flux<String> sendRequest(String topic, String sup, Integer uid) {  1 个用法   Shelter
    String streamID = "stream" + UUID.randomUUID().toString().replace( target: "-",  replacement: "");

    log.info("第一次请求发送成功");
    String accessToken = accessTokenService.getAccessToken();
    log.info("当前accesstoken为{}", accessToken);
    String requestBody = buildRequestBody(topic, sup);
    StringBuilder completeResponse = new StringBuilder();

    return this.webClient.post() RequestBodyUriSpec
            .uri( uri: botURL + "?access_token=" + accessToken) RequestBodySpec
            .contentType(MediaType.APPLICATION_JSON)
            .accept(MediaType.APPLICATION_JSON)
            .bodyValue(requestBody) RequestHeadersSpec<capture of ?>
            .retrieve() ResponseSpec
            .bodyToFlux(ApiResponse.class) Flux<ApiResponse>
            .flatMap(apiResponse -> {
                // 累积当前响应结果
                completeResponse.append(apiResponse.getResult());
                // 检查是否结束并且被截断
                if ("true".equals(apiResponse.getIsEnd())) {
                    if ("true".equals(apiResponse.getIsTruncated())) {
                        // 如果满足条件，记录日志并发起后续请求
                        log.info("Initial request is truncated, starting follow-up request...");
                        return Flux.concat(
                                Flux.just(apiResponse.getResult()), // 确保包含当前结果
                                sendFollowUpRequest(accessToken, topic, sup, completeResponse.toString(), uid, streamID)
                        );
                    } else {
                        log.info("第一次请求中结束流");
                        saveMarkdownToFile(completeResponse.toString(), topic, uid, streamID);
                        return Flux.just( data: apiResponse.getResult() + "\nstreamID:" + streamID);

                    }
                } else {
                    // 否则，只返回当前结果
                    return Flux.just(apiResponse.getResult());
                }
            });
}
```

2. 前端大纲生成打字机效果（Src.components.StreamDisplay.vue）

这段代码使用 Fetch API 发送 POST 请求到一个后端服务，并以流的形式接收和处理数据。通过读取流中的数据来动态更新和处理文本内容。该方法利用 ReadableStream 和 TextDecoder 处理流数据，实时解析和渲染 Markdown 文本，并在检测到特定模式时（如 streamID），触发事件。这种实现方式支持高效的数据处理，适用于需要实时数据传输和更新的应用场景。避免了用户盲目等待接近一分钟的大纲生成，提供更好的用户体验。

3. 后端解析 Markdown 文本（utils.MarkdownParser# parseMarkdownFile）

这个函数实现了一个解析 Markdown 文件的功能，将 Markdown 文本解析成结构化的 PPT 文本和图片链接。以下是这段代码的创新点和作用：

(1)**正则表达式匹配图片链接：**

- 使用正则表达式 !\\(([^)]+)\\)! 来匹配 Markdown 文本中的图片链接。
- 通过 Pattern 和 Matcher 来提取图片 URL，方便后续处理。

(2)**分层解析：**

- 代码能够解析不同层级的标题（#、##、###），并将这些标题及其对应的内容分层存储在不同的结构中。

- 通过 Pair 类存储标题和对应内容的键值对，结构清晰，易于后续使用。

(3)**段落处理：**

- 使用 StringBuilder 来收集和处理段落内容，并确保在遇到新标题时正确处理之前的段落。
- 这种方式能够高效地处理长文本并减少内存碎片。

4. 后端多线程优化合并 PPT 页面以及自动配图

(utils.MergePPT#mergePPT）

这段代码展示了一个后端多线程优化合并 PPT 页面的实现，同时还可以根据关键字调用 Unsplash 的接口为需要配图的 PPT 页面自动配图。以下是该实现的创新点：

(1)**多线程处理：**

- 使用 ExecutorService 创建一个固定线程池，线程数为系统可用处理器数量。通过多线程并发处理多个 PPT 文件，提高了处理效率。
- 每个线程提交的任务是从路径列表中读取 PPT 文件并创建 Presentation 对象。

(2)**线程安全的页码计数：**

- 使用 AtomicInteger 实现线程安全的页码计数。每个线程在处理 PPT 页面时都会增加计数，确保不同线程处理的页面不会出现冲突。

(3)**按顺序合并 PPT：**

- 使用 Future 数组存储各个线程处理的结果（即 Presentation 对象）。
- 在主线程中，通过遍历 Future 数组，按顺序获取每个 Future 的结果，并将其页面合并到最终的 Presentation 对象中。

(4)**自动配图功能：**

- 在处理 PPT 页面时，检查页面内容是否需要配图。
- 如果页面需要配图，通过调用 Unsplash 的 API，根据指定的关键字获取相关图片。
- 将获取到的图片插入到对应的 PPT 页面中，提升演示文档的视觉效果和专业性。

```java
public static byte[] mergePPT(ArrayList<Path> paths, String mainObject) {  1 个用法  Shelter *
    long begin = System.currentTimeMillis();

    if (paths == null || paths.isEmpty()) {
        log.error("没有提供PPTX文件。");
        return null;
    }

    ExecutorService executor = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors());
    Future<Presentation>[] futures = new Future[paths.size()];
    AtomicInteger page = new AtomicInteger( initialValue: 1); // 初始化页面计数器
    log.info("主题物品是{}",mainObject);
    mainObject=mainObject.replace( target: " ",  replacement: "-");
    String finalMainObject = mainObject;
    for (int i = 0; i < paths.size(); i++) {
        final int index = i;
        futures[i] = executor.submit(() -> {
            log.info("处理文件：{}", paths.get(index).getFileName());
            Presentation pres = new Presentation(Files.newInputStream(paths.get(index)));
            if (paths.get(index).getFileName().toString().contains("有图")) {...}
            return pres;
        });
    }

    Presentation finalPres = new Presentation();
    ByteArrayOutputStream baos = null;
    try {
        finalPres.getSlideSize().setSize( v: 1920,  v1: 1080, SlideSizeScaleType.EnsureFit);
        for (Future<Presentation> future : futures) {
            try {
                Presentation pres = future.get();
                for (ISlide slide : pres.getSlides()) {...}
                pres.dispose();
            } catch (InterruptedException | ExecutionException e) {...}
        }
        if (finalPres.getSlides().size() > 1) {...}
        baos = new ByteArrayOutputStream();
        finalPres.save(baos, SaveFormat.Pptx);
    } catch (Exception e) {...} finally {
        finalPres.dispose();
        executor.shutdown();
    }

    long end = System.currentTimeMillis();
    log.info("合并完成，共耗时 " + ((end - begin) / 1000.0) + " 秒。");
    return baos.toByteArray();
}
}
```

四、 实用点

```
public boolean preHandle(HttpServletRequest request, @NotNull HttpServletResponse response, @NotN
    String jwtToken = request.getHeader( s: "token"); //获取jwt token
    log.info("开始校验token:{}", jwtToken);
    int uid;
    try {
        uid = jwtUtils.getJwtPayloadUid(jwtToken);
    } catch (Exception e) {
        log.info("Interceptor校验jwt失败");
        log.error(e.getMessage());
        response.setContentType("application/json");
        response.getWriter().write(JSON.toJSONString(Result.error( msg: "invalid jwt token!")));
        return false;
    }
    log.info("Interceptor校验jwt成功");
    request.setAttribute( s: "uid", uid);
    return true; //放行
}
```

1. 后端拦截器鉴权(intercept.JwtInterceptor#preHandle)

这部分代码是用户管理模块 Auth 中拦截器配置的 JwtInterceptor 类的 preHandle 方法，它的主要功能是校验 JWT（JSON Web Token），并将校验出的 uid 添加到请求的属性中，给后面的服务提供可信的 uid。以下是这一部分代码的具体实用点：

(1)**JWT 令牌验证：**

- 从请求头中获取 JWT 令牌。
- 调用 jwtUtils.getJwtPayloadUid(jwtToken) 方法解析令牌，获取用户 ID (uid)。
- 如果令牌无效或解析失败，返回错误信息并终止请求，避免未经验证的用户访问系统资源。

(2)**请求属性设置：**

- 成功解析令牌后，将 uid 添加到请求的属性中，供后续处理使用，确保在整个请求处理流程中都能访问到已验证的用户身份信息。

2. 前端密码加密(Src.views.AuthView.vue)

前端使用 CryptoJS 对密码进行 sha256 散列后使用 axios 发送到后端，保障用户隐私和密码安全。

3. 后端保存和读取 PPT 工作区状态（savePPT&loadPPT）

这段代码展示了两个方法：`savePPT` 和 `loadPPT`，用于序列化和反序列化 `PPTRecord` 对象到文件。这两个方法有效地管理了 PPT 生成记录的持久化和恢复，支持 PPT 生成过程中的状态保持和历史数据回溯

```java
public boolean savePPT(PPTRecord pptRecord) {  2 个用法   Shelter *
    //获取主题
    String topic = pptRecord.getMarkdownData().getParseMarkdownText().get(0).getKey();
    //获取ppt封面
    String url = templateMapper.selectTemplateImage(pptRecord.getTemplates().get(0));

    //设置序列化文件名
    String filename = "";
    if (pptRecord.getStreamID() != null) {
        filename = topic + "_" + pptRecord.getStreamID();
        pptGenMapper.initializePPT(pptRecord.getUid(), topic, url, filename, LocalDateTime.now());
        pptRecord.setStreamID(null);
    } else {
        filename = topic + "_" + UUID.randomUUID();
        if (pptGenMapper.updatePPTRecord(pptRecord.getId(), topic, url, filename, LocalDateTime.now()) != 1) {
            return false;
        }
    }

    String jsonString = JSON.toJSONString(pptRecord, JSONWriter.Feature.PrettyFormat);
    // 将 JSON 字符串写入到文件
    Path path = Paths.get( first: recordPath + filename);
    try {
        Files.write(path, jsonString.getBytes());
    } catch (IOException e) {
        log.error(Arrays.toString(e.getStackTrace()));
        return false;
    }
    return true;
}

/**
 * 从文件中反序列化pptrecord
 *
 * @param id
 * @return
 */
public PPTRecord loadPPT(Integer id, Integer uid) {  1 个用法   Shelter *
    try {
        Path path = Paths.get( first: recordPath + pptGenMapper.getPPTRecordPathByID(id, uid));
        String jsonString = new String(Files.readAllBytes(path));
        return JSON.parseObject(jsonString, PPTRecord.class);
    } catch (IOException e) {
        log.error("Error reading from file: {}", e.getMessage());
        return null;
    }
}
```

4. 后端替换 PPT 模板占位符并保留格式

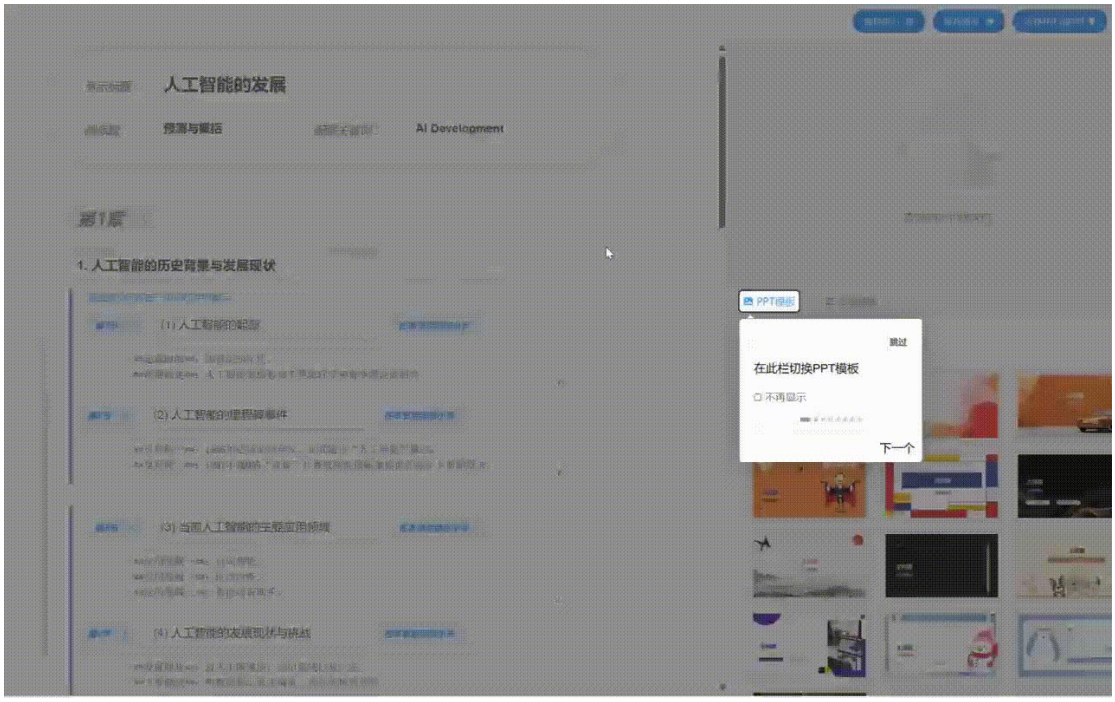
（utils.PPTXDataModifier#updateTextShape）

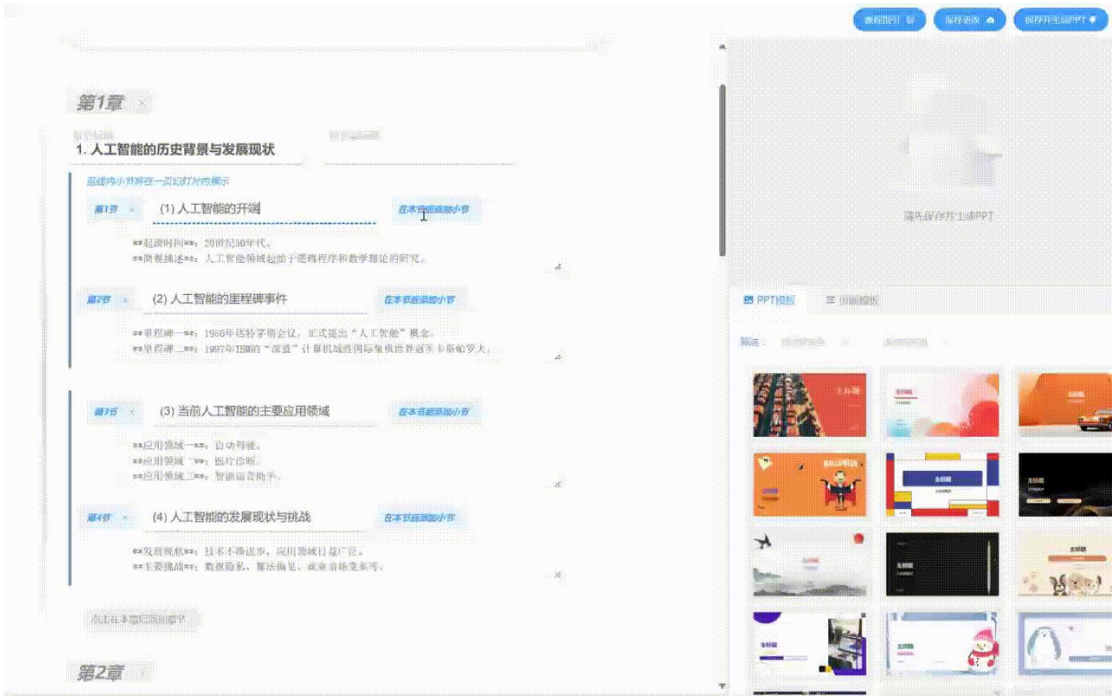
通过多次调用这个函数实现更新 PowerPoint 幻灯片中文本框的内容。它通过 Apache POI 库操作文本框，首先获取原有的文本样式（如字体大小、颜色、是否加粗或斜体、字体族），然后将新文本按照双星号标记加粗。

```
/**
 * 更新文本框内容
 */
private static void updateTextShape(XSLFTextShape textShape, String newText) {
    XSLFTextParagraph originalParagraph = textShape.getTextParagraphs().get(0);
    XSLFTextRun originalTextRun = originalParagraph.getTextRuns().get(0);

    double fontSize = originalTextRun.getFontSize();
    PaintStyle fontColor = originalTextRun.getFontColor();
    boolean isBold = originalTextRun.isBold();
    boolean isItalic = originalTextRun.isItalic();
    String fontFamily = originalTextRun.getFontFamily();

    String[] parts = newText.split( regex: "\\*\\*");
    for (int i = 0; i < parts.length; i++) {
        XSLFTextRun newTextRun = originalParagraph.addNewTextRun();
        newTextRun.setText(parts[i]);
        newTextRun.setFontColor(fontColor);
        newTextRun.setFontSize(fontSize);
        newTextRun.setBold(i % 2 == 1 || isBold);
        newTextRun.setItalic(isItalic);
        newTextRun.setFontFamily(fontFamily);
    }
    originalTextRun.setText("");
}
```

5. 前端易上手且自由灵活的 PPT 编辑界面(Src/EditView.vue)

通过 Introjs 通过七步教程带用户上手 PPT 编辑操作，并且通过复杂的 js 逻辑，支持了自由增删章节和小节，自由编辑所有文本。


人工智能的发展
预测与规划
AI Development
第1章
1. 人工智能的历史背景与发展现状
(1) 人工智能的起源
(2) 人工智能的里程碑事件
(3) 当前人工智能的主要应用领域
(4) 人工智能的发展现状与挑战
PPT模板
跳过
在此栏切换PPT模板
不再显示
下一个


第1章
1. 人工智能的历史背景与发展现状
(1) 人工智能的开端
(2) 人工智能的里程碑事件
(3) 当前人工智能的主要应用领域
(4) 人工智能的发展现状与挑战
第2章
PPT模板

五、 人员分工

队员 1（全栈人员）：

- 负责编写项目设计文件。
- 建立前后端代码仓库及接口文档。
- 设计并实现后端大纲生成模块 MarkdownGen。
- 设计并实现后端 PPT 生成模块 PPTGenerate。
- 设计并实现前端页眉组件 HeaderView。
- 实现前端登录页面 AuthView。
- 设计并实现前端大纲流式传输组件 ModalView 及 StreamDisplay。
- 设计并实现前端 PPT 编辑页面 EditView 的主要功能。
- 搭建并部署前端服务器。
- 协调团队工作和资源分配。

队员 2（后端人员）：

- 建立并维护数据库中的用户管理表 user_table。
- 编写用户管理模块 Auth。
- 实现阿里云短信验证接口。
- 负责后端服务器的搭建和日常维护。

队员 3（后端人员）：

- 参与项目前期的可行性验证和技术路线选择。
- 实现 PPT 生成模块中的文本替换功能，主要负责 PPTXDateModifier.java。
- 参与优化 MergePPT.java 的性能。

队员 4（前端人员）：

- 实现个人信息页 UserView 和 PersonView。
- 参与设计 PPT 编辑页面的新手指引。
- 参与构建首页 HomeView。
- 完成帮助页面 HelpView 及关于我们部分。
- 实现前端的头像、手机号和密码修改功能。
- 负责部分 PPT 模板的上传。
- 编写项目使用说明文档。

队员 5（前端人员）：

- 参与制作首页 HomeView。
- 实现“我的文档”页面 HistoryView。
- 制作 PPT 库页面 PPTView。
- 负责制作网页宣传视频。

- 负责部分 PPT 模板的上传工作。
- 设计部分图标和图片等静态资源。

六、 总结

本项目设计书展示了一个高度创新和实用的 PPT 自动生成系统的设计和实现过程。系统采用了前后端分离的架构，前端使用 Vue.js 实现用户界面，后端利用 Spring Boot 提供服务支持，并集成了第三方服务如 Unsplash API，以自动为 PPT 页面配图。

数据库设计方面，项目采用了合理的表结构设计，有效地管理 PPT 模板、用户信息和生成记录，确保数据的一致性和完整性。

系统在数据结构设计上，定义了核心数据结构如 PPTRecord、MarkdownData 和 Pair，为处理和存储各种格式的数据提供了可靠的基础。核心模块设计上，系统实现了用户认证、文本解析和 PPT 生成等关键功能，确保了系统的高效运行和用户友好的使用体验。

创新方面，本项目引入了多项前沿技术：包括通过流式传输技术实现的后端大纲生成，前端大纲生成的打字机效果，以及多线程优化合并 PPT 页面和自动配图功能。这些创新点不仅提升了系统的性能和用户体验，还显著提高了 PPT 生成的效率和质量。

在实用性方面，系统通过 JWT 鉴权机制确保了安全性，前端密码加密保护用户隐私，后端的 PPT 工作区状态保存与读取功能保证了用户工作的持久性和恢复性。此外，系统提供了一个易上手且灵活的 PPT 编辑界面，极大地方便了用户的操作。

总之，本项目通过先进的技术手段和创新的设计理念，实现了一个功能强大、性能优越且用户体验友好的 PPT 自动生成系统。该系统不仅满足了用户在 PPT 生成和编辑方面的需求，还具备高效、安全和易用的特点，具有广泛的应用前景和推广价值。