《Web 应用开发》课程大作业报告评阅表

| 项目 | 报告写作和格式规范，无抄袭情况（30%） | 符合课程要求（20%） | 功能合理且完善，代码无抄袭情况（30%） | 具有特色或创新性（10%） | 按时提交材料，材料无遗漏（10%） | 总分（100%）<br>签名： |
|---|---|---|---|---|---|---|
| 得分 | | | | | | |
| 得分说明/评语 | | | | | | |

# 2023-2024 第 2 学期

# 《Web 应用开发》

# 课程大作业报告

大作业题目：基于 Gin 框架实现简历解析系统 （命题/自选）

所在教学班：XX 班 姓名：张三 学号：123

## 一、功能概述

这是一个基于 Gin 框架的 Web 应用程序，主要用于处理 PDF 简历文件的上传和解析。以下是该程序的功能概述：

主要组件和功能：

1. 导入依赖：程序开始处导入了必要的 Go 包，包括处理 HTTP 请求、文件操作、正则表达式、模板渲染、PDF 解析等所需的库。

2. 定义数据结构：Resume 结构体：用于存储简历的 HTML 内容，其中`Content`字段用于存放处理后的 HTML 文本。

3. 辅助函数：SaveContentAsHTML：将 HTML 内容保存为 HTML 文件，用于持久化存储或后续操作。HighlightKeywords：在简历文本中查找并高亮关键词，如“姓名”、“专业”等，提高简历内容的可读性。

4. 核心功能：parsePDF：从指定路径的 PDF 文件中提取文本内容，并进行处理。该函数负责打开 PDF 文件、读取每一页的内容、提取文本、高亮关键词，并最终生成结构化的`Resume`对象。

5. HTTP 处理器：uploadHandler：处理文件上传和结果显示的 HTTP 处理器函数。它根据请求方法（GET 或 POST）执行不同的操作:GET 请求：渲染并返回上传页面`upload.html`。POST 请求：处理文件上传，创建临时文件存储上传的 PDF 文件，调用`parsePDF`函数解析 PDF 文件，并将结果显示在`result.html`页面。

6. 路由设置：在`main`函数中，设置路由规则，将`/upload`路由的请求映射到`uploadHandler`函数。

7. 静态文件服务：设置静态文件目录，用于提供 CSS、JavaScript 等静态资源。

8. 服务器启动：启动 Gin 服务器，监听 8080 端口，并处理传入的 HTTP 请求。

工作流程：

1. 用户访问服务器的`/upload`页面，看到一个文件上传表单。

2. 用户选择 PDF 简历文件并提交表单，触发 POST 请求。

3. `uploadHandler`接收请求，处理文件上传逻辑。

4. 服务器创建临时文件，保存上传的 PDF 文件。

5. `parsePDF`函数被调用，解析 PDF 文件，提取并高亮文本内容。

6. 解析结果通过`result.html`页面展示给用户。

## 二、实现原理和方法

实现一个基于 Gin 框架的 PDF 简历解析 Web 应用的过程可以分为几个主要阶段：初始化项目、搭建服务器、创建处理函数、实现 PDF 解析、错误处理、结果展示和服务器运行。以下是详细的实现过程描述：

1. 初始化项目

- 创建一个新的 Go 模块目录。

- 导入所需的包：Gin 框架用于 Web 服务，`unidoc`用于 PDF 解析，以及其他标准库。

2. 搭建服务器

- 使用 Gin 框架初始化 Web 服务器。

- 设置静态文件目录，用于服务 CSS、JavaScript 等资源。

- 定义路由，例如`/upload`用于文件上传。

3. 创建处理函数

- 实现`UploadHandler`函数，用于处理文件上传的 POST 请求和显示上传页面的 GET 请求。

4. 实现 PDF 解析

- 定义`ParsePDF`函数，用于打开、读取和解析 PDF 文件。

- 使用`unidoc`库遍历 PDF 的每一页，提取文本。

- 定义`HighlightKeywords`函数，使用正则表达式高亮简历中的关键词。

5. 错误处理

- 在文件操作和 PDF 解析过程中，检查错误并返回适当的 HTTP 状态码和错误信息。

6. 结果展示

- 将解析后的简历内容传递给模板引擎，渲染成 HTML 页面。

- 使用 Gin 的模板功能，将结果展示给用户。

7. 服务器运行

- 在`main`函数中，启动 Gin 服务器，监听指定端口（如 8080）。

技术细节：

- Gin 框架：用于快速搭建 Web 应用，处理 HTTP 请求。

- unidoc 库：用于处理 PDF 文件，提取文本内容。

- 正则表达式：用于匹配简历中的关键词。

- 模板引擎：用于渲染 HTML 页面，展示解析结果。

逻辑流程：

1. 服务器启动：程序运行，Gin 服务器启动，监听端口。
2. 路由设置：定义`/upload`路由，关联到`UploadHandler`函数。
3. 文件上传：
   - 用户访问`/upload`，GET 请求返回上传表单。
   - 用户填写表单并上传 PDF 文件，POST 请求触发`UploadHandler`。
4. 文件处理：
   - `UploadHandler`接收文件，保存为临时文件。
   - 调用`ParsePDF`解析 PDF 文件，提取文本。
   - 使用`HighlightKeywords`高亮关键词。
5. 结果保存：可选地，将结果保存为 HTML 文件。
6. 页面渲染：将解析结果传递给`result.html`模板，渲染页面。
7. 结果展示：浏览器显示渲染后的 HTML 页面，展示简历内容。

错误处理：

- 文件上传失败：返回 400 错误。
- 临时文件创建失败：返回 500 错误。
- 文件打开失败：返回 500 错误。
- PDF 解析失败：返回 500 错误。
- 模板渲染失败：返回 500 错误。

这个流程图展示了从服务器启动到结果展示的整个过程，包括错误处理和用户交互。通过这种方式，用户可以上传 PDF 简历，系统将解析并高亮简历中的关键词，最后将结果展示给用户。

流程图:

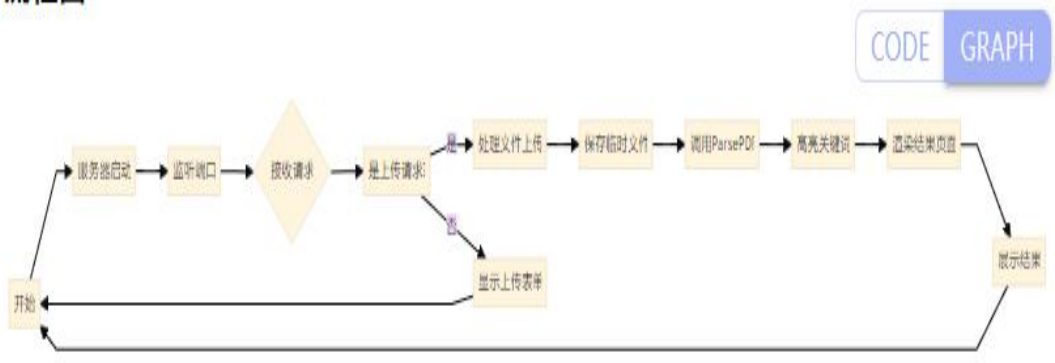


图 1 gin 简历解析系统流程图

## 三、实验和测试环境

GoLand

Go 1.22.2


图 2 生产环境以及配置图

## 四、运行和测试结果

描述一下结果，附带效果截图

简历上传功能


图 3 简历上传功能运行效果


图 4 简历解析功能运行效果

```
2024/05/28 12:58:50 Server is running on :8080
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :8080
[GIN] 2024/05/28 - 12:59:01 | 200 |       530.6µs |             ::1 | GET      "/upload"
Unlicensed copy of unidoc
```

图 5 简历解析功能运行效果

## 五、结论和总结

这是一个一个基于 Go 语言和 Gin 框架开发的简历解析 Web 应用，旨在自动化地从 PDF 简历中提取关键信息，并通过用户友好的 Web 界面展示。该系统采

用模块化设计，易于扩展和维护。在技术选型上，选用了轻量级的 Gin 框架处理 Web 请求，以及`unidoc`库进行 PDF 文件的解析。然后调用阿里云的通义千问大模型接口来进行简历解析，从而系统实现了文件上传、PDF 内容提取、关键词高亮显示等功能，并通过 HTML 模板渲染结果。安全性方面，进行了基本的文件验证以防止恶意上传。性能上，虽然当前实现适用于常规文件大小，但需进一步优化以应对高并发场景。用户体验方面，简洁的界面和自动化处理流程大大提升了便捷性。未来，系统将考虑集成更高级的文本分析技术，增加用户认证，以及部署到云平台，以提高服务的质量和可靠性。

## 附：关键代码（不超过 100 行）

```
// uploadHandler 是处理文件上传和结果显示的 HTTP 处理器函数
func uploadHandler(c *gin.Context) {
    // 检查请求方法是否为 GET
    if c.Request.Method == "GET" {
        // 如果是 GET 请求，渲染并返回 upload.html 页面
        c.HTML(http.StatusOK, "upload.html", gin.H{})
        return
    }

    // 如果请求方法是 POST，则处理文件上传
    if c.Request.Method == "POST" {
        // 解析 multipart 表单数据
        form, err := c.MultipartForm()
        if err != nil {
            // 如果解析表单失败，返回错误信息
            c.JSON(http.StatusBadRequest, gin.H{"error": "Failed to parse form"})
            return
        }

        // 从表单中获取文件列表
        files := form.File["file"]
        if len(files) == 0 {
            // 如果没有文件被上传，返回错误信息
            c.JSON(http.StatusBadRequest, gin.H{"error": "No file uploaded"})
            return
        }
        // 处理第一个上传的文件
        file := files[0]

        // 创建一个临时文件，用于存储上传的 PDF 文件
        tempFile, err := os.CreateTemp("", "upload-*.pdf")
        if err != nil {
            // 如果创建临时文件失败，返回错误信息
```

```
            c.JSON(http.StatusInternalServerError, gin.H{"error": "Temporary file creation failed"})
            return
        }
        defer tempFile.Close() // 确保在函数结束时关闭临时文件
        defer os.Remove(tempFile.Name()) // 确保在函数结束时删除临时文件

        // 打开上传的文件内容
        fileContent, err := file.Open()
        if err != nil {
            // 如果打开文件失败，返回错误信息
            c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to open file"})
            return
        }
        defer fileContent.Close() // 确保在函数结束时关闭文件内容

        // 将上传的文件内容复制到临时文件
        _, err = io.Copy(tempFile, fileContent)
        if err != nil {
            // 如果文件保存失败，返回错误信息
            c.JSON(http.StatusInternalServerError, gin.H{"error": "File saving failed"})
            return
        }

        // 调用 parsePDF 函数解析 PDF 文件
        resume, err := parsePDF(tempFile.Name())
        if err != nil {
            // 如果 PDF 解析失败，返回错误信息
            c.JSON(http.StatusInternalServerError, gin.H{"error": "PDF parsing failed"})
            return
        }

        // 渲染 result.html 页面，并传递解析后的简历数据
        c.HTML(http.StatusOK, "result.html", gin.H{"resume": resume})
    }
}

func main() {
    // 创建一个默认配置的 Gin 路由器实例
    router := gin.Default()

    // 加载所有 templates 目录下的 HTML 模板文件
    router.LoadHTMLGlob("templates/*")
```

```
    // 设置静态文件目录，用于提供静态资源如 CSS、JavaScript 等
    router.Static("/static", "./static")

    // 设置/upload 路由的处理函数为 uploadHandler
    // POST 请求用于文件上传，GET 请求用于显示上传页面
    router.POST("/upload", uploadHandler)
    router.GET("/upload", uploadHandler)

    // 打印服务器正在监听的端口信息
    log.Println("Server is running on :8080")
    // 运行 Gin 路由器，监听并服务于 8080 端口
    if err := router.Run(":8080"); err != nil {
        log.Fatalf("Failed to start server: %v", err)
    }
}
```

请务必提交以下电子版文件（缺一不可）：

1.word 版本报告（.doc 或.docx）。除了电子版外，还需要打印并提交纸质版本。

2.将 word 版本报告转为 PDF 格式（.pdf），请确保与 word 版本内容一致。

3.全部源代码打包到一个文件中（.zip）。

4.关键代码（100 行左右）。加注释!

5.运行效果截图（png 或 jpg 格式），选择 1 张你认为效果最好的提交。更多的效果截图可以放在 word 版本报告中。

文件命名要求：

以上所有文件请以学号开头进行命名：

例如：

32019070299-张三.doc

32019070299-张三.pdf

32019070299-张三.zip

32019070299-张三.go

32019070299-张三.png

以上文件请分别以附件方式上传。

注意：

1.源代码中请使用相对路径，不要使用绝对路径（绝对路径会导致老师这里无法运行）。

2.请务必包含全部需要的文件，资源文件等（但不要包含没有用的多余文件），尽量不要使用 CDN（有可能也会无法运行）。

3.如果你的代码中包含或依赖可能在老师这里无法运行的环境（例如有数据库等），请在 word 版报告中的"实验和测试环境"部分中说明清楚。

4.请在 word 版本报告中包含你的运行效果截图（以免老师这里无法运行导致看不到你的效果），截图请涵盖你实现的所有功能，截图可以是整个页面的布局图，也应当包含局部细节图（用来表明你实现的具体功能，局部图能更好地展现细节，同时能让老师重点关注到你想表现的内容，而不是让老师在一幅图中去猜你到底想展现什么内容）。

5.word 版报告（及 PDF）务必阐述清晰（你要把老师看作是完全不了解你做了什么的人，然后想办法把你做的内容说清楚，说明白，阐述请用书面语，不要过于口语化），排版规范（字体，字号，图下方要有图 1.xxx，图 2.xxx 这样的题注）。题注是"指出现在图片下方的一段简短描述"，表格的题注要放在表格的上方。题注也会帮助老师理解你的图和表想表达的内容。

6.word 版报告没有必要粘贴源代码（如有，请加注释）。

7.请提交最终版本程序（删除代码中没有用的内容，删除不需要和没用的多余文件）。