



江西理工大学

本科毕业设计(论文)

题 目： 基于 Golang 的家庭媒体管理软件的设计与实现

专题题目：

学 院： 软件工程学院

专 业： 软件工程（工程造价方向）

班 级： 软件造价 16 级 1 班

学 号： 5720161117 学 生： 刘得根

指导教师： 邱晓红 职 称： 教授

指导教师： 职 称：

时 间： 2020 年 6 月

江西理工大学 软件工程 学院 2020 届 本科毕业设计（论文）任务书

题 目：基于 Golang 的家庭媒体管理软件的设计与实现

专题题目：

原始依据：

- **工作基础**

熟练使用 Golang 语言，学习 Vue.js、Javascrrip、ffmpeg，SQLite 等相关的开发技术有关软件的使用，同时还要学习 Linux 系统的使用。

- **研究条件**

通过网络资源和书籍资料等教育资源，了解实现此目标所需的技术和相关协助程序，并独立学习与软件开发相关的知识。

该项目所生成的软件对硬件的最低要求：

- ① 硬件要求：1 核 CPU，256M 内存以及 1G 硬盘。
- ② 软件要求：Windows7/8/10/server 操作系统、Linux 操作系统、各种内核浏览器

- **应用环境**

该软件的用户界面主要运行在各平台的浏览器上。

- **工作目的**

使用 Golang 语言和 Vue.js 搭建一个可以在浏览器中运行的家庭媒体管理系统。该软件主要致力于家庭媒体跨终端的共享，以及通过网络爬虫完善媒体的海报等内容。构建一个直观高效的媒体管理软件。

主要内容和要求：

- **研究内容**

该软件主要适用与绝大多数硬件条件下的家庭媒体共享与管理。在运行上对硬件的要求极低，在操做上，无论是手机端、PC 端还是智能电视，只要是在有浏览器的地方就能获得一致的操作体验。系统主要由用户操作、文件扫描、文件监听、转码、网络爬虫五个功能模块组成。

用户操作主要是在使用该软件时，由用户自己进行的注册、登录、修改系统配置。文件扫描主要是扫描用户媒体库下的文件，然后在界面中展示给用户以供用户管理和播放。文件监听模块是监听媒体库中文件的变动信息，及时反馈给系统。转码模块是在操作系统 CPU 空闲时，调用 ffmpeg 将媒体文件转成适合通过浏览器播放的视频编码格式。网络爬虫是用来获取网络公开信息，补充视频资料，使用户有更好的操作体验。

- **主要技术技术指标与技术参数**

主要使用 Golang 进行软件开发，前端使用 Vue.js 构建，使其可以在不同尺寸的屏幕上有一致的操作体验。转码使用了 Go 语言绑定的开源软件 ffmpeg。

- **具体要求**

巩固在校所学知识，进一步学习实现软件功能所需的应用背景知识，综合应用软件工程理论知识，独立完成项目系统开发研制过程，进行需求分析、设计、实现并测试，设计出符合需求的家庭媒体管理软件，并独立完成论文报告根据要求写好小论文。毕业论文接受学校的检查，查重率符合学校的查重要求。

日程安排:

2020.01.04~2020.01.06	完成开题报告
2020.01.07~2020.05.19	写作、修改、完善、定稿
2020.04.15~2020.04.21	中期检查
2020.05.26~2020.05.30	查重检测
2020.05.31~2020.06.03	答辩

主要参考文献和书目:

- [1] it之家. 网易网盘关闭部分入口，国内网盘行业“三足鼎立”[EB/OL]. <https://www.ithome.com/0/446/817.htm>, 2019-9-23/2019-12-27.
- [2] 什么值得买. 群晖中国区 CEO 陈予建访谈：NAS 市场仍在快速增长，中小企业用户成为增量主力[EB/OL]. <https://post.smzdm.com/p/az59p475/>, 2018-10-16/2019-12-27.
- [3] 维基百科. 服务器消息块[EB/OL]. <https://zh.wikipedia.org/wiki/伺服器訊息區塊/>, 2018-8-30/2019-12-29
- [4] nurdletech. Securing FTP using SSH[EB/OL]. <https://nurdletech.com/linux-notes/ftp/ssh.html>
- [5] 什么值得买. 原创 篇三：家庭多媒体中心软件 Emby 介绍[EB/OL]. <https://post.smzdm.com/p/735222/>, [2018-7-26]/2019-12-29
- [6] 高素春. UML 在面向对象软件开发中的应用[J/OL]. 河南科技:1-6[2019-12-28]. <http://kns.cnki.net/kcms/detail/41.1081.T.20191213.1548.002.html>.
- [7] 刘秋香, 刘振伟. 浅析几款主流的 UML 建模工具[J]. 电脑知识与技术, 2018, 14(32):245+253.
- [8] Andrawos M, Helmich M. Cloud Native Programming with Golang: Develop microservice-based high performance web apps for the cloud with Go[M]. Packt Publishing Ltd, 2017.
- [9] James Whitney, Chandler Gifford, Maria Pantoja. Distributed execution of communicating sequential process-style concurrency: Golang case study[J]. , 2019, 75(3).
- [10] Togashi N, Klyuev V. Concurrency in Go and Java: performance analysis[C]//2014 4th IEEE International Conference on Information Science and Technology. IEEE, 2014: 213-216.
- [11] 麦冬, 陈涛, 梁宗湾. 轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版), 2017(07):58-59.
- [12] Hong P. Practical Web Design: Learn the fundamentals of web design with HTML5, CSS3, Bootstrap, jQuery, and Vue.js[M]. Packt Publishing Ltd, 2018.
- [13] 万玛宁, 关永, 韩相军. 嵌入式数据库典型技术 SQLite 和 BerkeleyDB 的研究[J]. 微计算机信息, 2006 (012): 91-93.
- [14] 王珊, 萨师煊. 数据库系统概论(第 5 版)[M]. 高等教育出版社, 2014-09

- [15] medium. 5 Tips To Speed Up Golang Development With IntelliJ Or Goland. [EB\OL]. <https://medium.com/@keperry/5-tips-to-speed-up-golang-development-with-intellij-or-goland-6646110e9c5e>, 2019-5-2/2019-9-27
- [16] 邓正良. 基于 FFmpeg 和 SDL 的视频流播放存储研究综述[J]. 现代计算机, 2019 (22) :47-50.

指导教师（签字）：

年 月 日

江西理工大学 软件工程 学院 2020 届 本科毕业设计(论文)开题报告

学号: 5720161117

姓名: 刘得根

班级: 软件造价 161

题 目: 基于 Golang 的家庭媒体管理软件的设计与实现

专题题目:

本课题来源及研究现状:

● 课题来源

随着时代的发展,高清的电视设备越来越大众化。但是由于网络带宽的限制,优酷、腾讯视频、爱奇艺等内容提供商提供的流媒体服务并不能满足用户对清晰度的要求。对清晰度有要求的用户往往选择自己去网络获取高清晰度的媒体文件。这些文件以前被储存在网络中,但近几年国内网盘行业一片颓势,随着其它网盘的关闭和百度网盘的限速。越来越多的用户想收回自己对数据的控制权,而不再受制于网盘。再加上光纤的普及,带动了 NAS(Network Attached Storage)市场的增长。NAS 按字面简单说就是连接在网络上,具备资料存储功能的装置,因此也称为“网络存储器”。用户将视频、音频等媒体收回自己手里之后,市场上并没有一个适合国人使用的软件去更好的管理这些文件。所以很有必要设计一个家庭媒体管理软件,去解决这一困境,让用户观看储存在 NAS 中的视频媒体是能够拥有类似于优酷、爱奇艺的体验。

● 研究现状

目前,可用的家庭共享主要分为两种,一种是基于传统的文件共享协议 SMB、FTP 等方式手动管理文件,另一种是求助于专业的家庭媒体管理软件如 Plex、Emby、Jellyfin 等。但在实际体验中,这两者或多或少都有些不足。

1. 传统的文件管理

目前,内网中文件共享最常用的是 SMB 虽然在 Windows/Unix 中能够像访问本地文件一样访问 SMB 服务器上的文件,但是 SMB 也有占用网络带宽高、网络延迟大等问题,并且在 Android/iOS 等移动平台,SMB 协议的支持情况却不如人意。在这些平台要访问 SMB 服务器往往得借助于第三方软件。安卓端甚至要通过将 SMB 协议转换为 FTP 协议才能访问文件,而在这转换过程中很大程度上将会降低文件的传输效率。

相较于 SMB,FTP 虽然支持公网、允许直接通过浏览器访问。但是安卓端的一些国产浏览器如腾讯的手机 QQ 浏览器、360 浏览器等却阉割了对 FTP 协议的支持。并且大部分 FTP 的作用供用户上传和下载文件,在线播放显然不是他的主

要功能。虽然一些软件做到了直接播放 FTP 服务器上的内容，但是效果却不尽人意。再者 FTP 并不是一个安全的协议，近年来暴露出许多安全漏洞。因此对于家庭媒体服务器来说，FTP 甚至不如 SMB。

由此，可以看出传统的文件共享协议由于协议本身的局限性以及实现协议的各个软件自身标准不一，并不能保证各个设备之间具有统一的体验。所以 SMB、FTP 并不适合做为一个家庭媒体文件共享的服务。再者，用管理文件的方式管理视频等媒体文件并不是一个聪明的选择。用户需要的是一个直观的体验，最好能够直接看到媒体文件的海报，分类，上映日期等信息。而且，如果能够自动获取字幕等信息，以及在各个不同设备之间拥有统一的使用逻辑，媒体管理服务将会有一个绝佳的体验。

2. 现存媒体管理软件

目前专注于媒体服务器的软件有 Plex、Emby 与 Jellyfin，其中 Plex 与 Emby 是商业软件，某些服务需要付费才能使用。Jellyfin 是开源的自由软件。这三个软件都实现了所需的功能但却各有各的缺陷。

Plex 与 Emby 都是商业软件，这两者都有强大的功能以及不错的体验，也都支持主流的平台。但是由于他们的功能过于繁杂，对性能的要求都比较高，在低性能的设备上（如我的群晖 DS119J）上流畅地运行。这两个软件都是国外公司开发，遵循的是国外用户的使用体验，虽然软件有中文的版本，但是在软件的某些地方，还保留着英文。而且这两个软件使用的媒体数据是 IMDB 的数据，并不是国内用户常用的豆瓣。因此这两个软件在一定程度上并不是非常契合国内用户需求。并且这两个软件在使用过程种或多或少地暴露出了某些问题。再者，这两个软件的免费授权都有一些限制，解锁高级用户却要价不菲，最便宜的都要 4.99 美元一个月。

相比于 Plex 与 Emby，Jellyfin 是免费的自由软件。他也是由国外用户主导开发，也有 Plex 和 Emby 不接地气的问题。Jellyfin 使用 C# 开发并不能很好地支持各大平台，并且性能上也略微有些问题。

在如今的市场中，找不到一个高性能，高效率，契合用户所需功能的软件。传统文件共享服务，实用但却并不优雅；三大媒体共享软件，优雅但却或多或少地有某些缺陷。找遍这三个软件的官网，并没有找到它们对 MIPS 架构的支持。而如今国产 CPU 获得进展最大的龙芯使用的恰恰是 MIPS 架构。Golang 得益于它超强的交叉编译能力，不需要拥有 MIPS 设备就可以编译出能够在 MIPS 平台的软件。不费吹灰之力解决了多平台支持能力。

本课题的目标是提供一个高性能、多平台支持的媒体管理软件。由于 Go 语言天生有较高的性能与跨平台能力。这个软件将会有很大的竞争力。

课题研究目标、内容、方法和手段：

- **目标**

研究目标是构建一个性能要求低、支持主流操作系统、支持主流 CPU 指令集、安装方便、配置简单、易于使用的 B/S 模式开源家庭媒体管理软件。

具体分为以下三个目标：

- 1). 用 Vue 构建一个美观、简洁、现代化的前端界面。
- 2). 在功能上实现用户登陆，管理和播放媒体文件。软件能够扫描媒体文件并且从网络中下载 相对应的封面，字幕文件等。
- 3). 软件能够在空闲时间对不支持浏览器播放的媒体文件进行转码。实现软件跨平台、跨操作系统的能力。提高软件的运行性能，力求安装配置简单。

● 内容

根据软件功能预计分为以下模块：

- (1) 初始化模块：处理第一次启动时进行用户创建，媒体库本地路径的指定等工作。
- (2) 用户登陆模块：处理用户的登陆和修改密码等工作。
- (3) 媒体文件扫描模块：是本课题的核心模块之一，主要负责扫描文件，根据网络下载模块查询文件信息对文件进行分类。
- (4) 文件展示模块：是本课题核心模块之一，主要根据分类展示媒体文件，以供用户选择。
- (5) 文件管理模块：对文件进行删除，修改信息等
- (6) 视频播放模块：本课题核心模块之一，主要通过 http 协议将用户所选择的内容推送至浏览器视频播放软件，以供用户播放。
- (7) 网络爬虫模块：从网络获取相关视频信息，从公共数据库获取媒体信息以及字幕文件，当数据库不可用或者缺失时，通过网络爬虫从特定的网站获取媒体文件信息以及字幕文件。
- (8) 转码模块：当服务器 CPU 空闲时，对媒体不被支持的编码格式的媒体文件进行转码。

● 方法与手段

软件的开发过程中主要采用面向对象的软件开发方法。这是一种自底向上和自顶向下相结合的方法，而且它以对象建模为基础，从而不仅仅包含了软件输入输出的数据结构，而是包含了所有对象的数据结构。

统一建模语言（UML）是面对对象软件的标准化建模语言，是一种用于说明、可视化、构建和编写一个正在开发的、面向对象的、软件密集系统制品的开放方法。通过利用 UML 建模技术，利用模型元素来组建整个系统的模型，模型元素包括系统中的类、类和类之间的关联、类的实例相互配合实现系统的动态行为等。本课题研究利用了 Rational Rose 建模工具，为媒体管理系统进行逻辑梳理和开发准备。

在功能实现上主要采用 Golang、Vue.js、SQLite、ffmpeg 等技术。

软件使用 Vue.js 构建精美的响应式前端页面，能够自适应电视、手机等不同尺寸的屏幕。而且能够兼容主流浏览器。

软件的后台主要由 Golang 来构建。Golang 是 Google 开发的一种静态强类型、编译型、并发型，并具有垃圾回收功能的编程语言。它是一种快速的，静态类型的编译语言，但写起来感觉就像是一种动态类型的解释语言。本项目之所以选择 Go 主要看中了其交叉编译的能力和并发机制，不错的性能以及越来越完善的软件生态。这是我们能够快速高效构建软件的保证。

数据库主使用常用的嵌入式数据库 SQLite。占用资源非常的低，能够支持 Windows/Linux/Unix 等等主流的操作系统。由于是集成在应用中，契合了设计目标中“安装部署简单”的特点。

视频扫描模块会对特定的目录进行媒体文件的扫描。对扫描到的每个文件赋予一个独一的 ID。调用 ffmpeg 对文件进行分析，得到其编码等详情信息，并使用了一个由 Golang 编写的爬虫对网络上的公开数据进行采集，将采集到的信息和文件的信息关联起来存入数据库。

在视频展示时，会从数据库中提取视频的数据和网络爬虫所采集的数据，并且展示到前端页面。当用户点击播放按钮时，转到相应的界面对本地文件进行播放。

当扫描到的文件并不适合通过浏览器播放器进行播放时，软件会在合适的时间自动调用 ffmpeg 进行转码操作，将视频转换成合适的编码格式。

该软件项目测试过程，主要采用黑盒测试。黑盒测试是按照软件的规格进行测试，也就是说，这种测试方法不像 白盒测试那样需要考虑内部工作原理，而是以用户的角度去看待一款软件产品，用这种方式去测试可能发现很多有价值的问题。

设计（论文）提纲及进度安排：

设计（论文）提纲：

第一部分：绪论（介绍背景、意义，研究现状）

第二部分：相关技术与方法（介绍相关技术背景）

第三部分：系统分析（系统需求分析）

第四部分：系统设计（系统各模块设计）

第五部分：系统实现（介绍实现方法）

第六部分：系统测试（各功能模块的测试）

第七部分：总结

进度安排：

2019.12.24～2019.12.30

选题

2020.01.04~2020.01.06	完成开题报告
2020.01.07~2020.05.19	写作、修改、完善、定稿
2020.05.31~2020.06.03	答辩

主要参考文献和书目:

- [1] it之家. 网易网盘关闭部分入口, 国内网盘行业“三足鼎立”[EB/OL]. <https://www.ithome.com/0/446/817.htm>, 2019-9-23/2019-12-27.
- [2] 什么值得买. 群晖中国区 CEO 陈予建访谈: NAS 市场仍在快速增长, 中小企业用户成为增量主力[EB/OL]. <https://post.smzdm.com/p/az59p475/>, 2018-10-16/2019-12-27.
- [3] 维基百科. 服务器消息块[EB/OL]. <https://zh.wikipedia.org/wiki/伺服器訊息區塊/>, 2018-8-30/2019-12-29
- [4] nurdletech. Securing FTP using SSH[EB/OL]. <https://nurdletech.com/linux-notes/ftp/ssh.html>
- [5] 什么值得买. 原创 篇三: 家庭多媒体中心软件 Emby 介绍[EB/OL]. <https://post.smzdm.com/p/735222/>, [2018-7-26]/2019-12-29
- [6] 高素春. UML 在面向对象软件开发中的应用[J/OL]. 河南科技:1-6[2019-12-28]. <http://kns.cnki.net/kcms/detail/41.1081.T.20191213.1548.002.html>.
- [7] 刘秋香, 刘振伟. 浅析几款主流的 UML 建模工具[J]. 电脑知识与技术, 2018, 14(32):245+253.
- [8] Andrawos M, Helmich M. Cloud Native Programming with Golang: Develop microservice-based high performance web apps for the cloud with Go[M]. Packt Publishing Ltd, 2017.
- [9] James Whitney, Chandler Gifford, Maria Pantoja. Distributed execution of communicating sequential process-style concurrency: Golang case study[J]. , 2019, 75(3).
- [10] Togashi N, Klyuev V. Concurrency in Go and Java: performance analysis[C]//2014 4th IEEE International Conference on Information Science and Technology. IEEE, 2014: 213-216.
- [11] 麦冬, 陈涛, 梁宗湾. 轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版), 2017(07):58-59.
- [12] Hong P. Practical Web Design: Learn the fundamentals of web design with HTML5, CSS3, Bootstrap, jQuery, and Vue.js[M]. Packt Publishing Ltd, 2018.
- [13] 万玛宁, 关永, 韩相军. 嵌入式数据库典型技术 SQLite 和 BerkeleyDB 的研究[J]. 微计算机信息, 2006 (01Z): 91-93.
- [14] 王珊, 萨师煊. 数据库系统概论(第 5 版)[M]. 高等教育出版社, 2014-09
- [15] medium. 5 Tips To Speed Up Golang Development With IntelliJ Or Goland. [EB\OL]. <https://medium.com/@kepererry/5-tips-to-speed-up-golang-development-with-intellij-or-goland-6646110e9c5e>, 2019-5-2/2019-9-27
- [16] 邓正良. 基于 FFmpeg 和 SDL 的视频流播放存储研究综述[J]. 现代计算机, 2019(22):47-50

指导教师审核意见:

指导教师 (签字): 年 月 日

摘 要

随着时代的发展,高清的电视设备越来越大众化。但是由于网络带宽的限制,优酷、腾讯视频、爱奇艺等流媒体运营商提供的内容参差不齐。对内容有更高追求的用户往往会选择通过种子等媒介下载质量更高的影视资源。但是缺少了运营商的服务,这些没有统一格式的视频显得杂乱不堪。因此对这些用户来说,一个专注于家庭媒体管理的软件至关重要。

系统使用谷歌公司推出的 Go 语言构建,并使用了 Vue.js 简化前端页面构建的复杂度。为了适应不同屏幕大小的设备,在网页开发上使用了 B/S 架构。系统的大部分功能都可以通过以上技术实现。系统在初步扫描文件时,会通过辨别文件的 MD5 码来识别重复的文件。为了补充视频信息,使用了网络爬虫技术在网络的公开数据中采集媒体的海报、简介等信息。得益于这些信息,能够实现媒体的分类整理与字幕等信息的进一步获取。通过浏览器播放视频时会出现一个问题——浏览器往往只能解码少数几种公开的视频编码格式。为了让用户不同的视频能够在浏览器上有统一的播放体验,使用了 ffmpeg 在 CPU 空闲时对文件转码,从而使软件更加人性化。

关键词: Go 语言; 媒体管理; B/S 框架; ffmpeg

ABSTRACT

With the development of the technology, high-definition videos are becoming more and more popular. However, due to the limitation of network bandwidth, the content provided such as Youku, Tencent Video, iQiyi, etc. is uneven. Users who has a higher pursuit of videos often choose to download higher quality film or television resources by other where such as torrents. But without the service of the company, these videos without a unified format appear to be messy. Therefore, for these users, a software focused on home media management is essential.

The system is built by the Golang witch introduced by Google, and Vue.js is used to simplify the complexity of front-end page construction. In order to adapt to devices with different screen sizes, B / S architecture is used in web page development. Most functions of the system can be realized by the above technology. When the system initially scans the file, it will identify the duplicate file by identifying the MD5 code of the file. In order to supplement video information, web crawler technology is used to collect media posters, profiles and other information from public data on the network. Thanks to this information, it is possible to further obtain information such as media sorting and captioning. There is a problem when playing video through a browser-browsers can often only decode a few public video encoding formats. In order to allow users to have a unified video playback experience on different browsers, ffmpeg is used to transcode files when the CPU is idle, thereby making the software more user-friendly.

Keywords:Golang; Media Management Software;ffmpeg;B/S

目 录

第一章 绪论	1
1.1 研究背景与意义	1
1.2 研究现状	1
1.3 论文结构	2
1.4 本章小结	3
第二章 相关技术与方法	4
2.1 架构概述	4
2.2 关键技术简介	4
2.3 开发工具	5
2.4 本章小结	6
第三章 系统分析	7
3.1 可行性分析	7
3.2 需求分析	7
3.3 本章小结	10
第四章 系统设计	11
4.1 系统类分析	11
4.2 关键业务设计	14
4.3 数据库设计	16

4.4 本章小结	20
第五章 系统实现	21
5.1 网页登陆	21
5.2 媒体智能管理	22
5.3 语音控制	25
5.4 文件展示	25
5.5 监听模块	27
5.6 设置	29
5.7 交叉编译	30
5.8 本章小结	32
第六章 系统测试	33
6.1. 系统测试综述	33
6.2. 测试用例	33
6.3. 本章小结	40
第七章 总结	42
参考文献	43

第一章 绪论

1.1 研究背景与意义

随着时代的发展,高清的电视设备越来越大众化。但是由于网络带宽的限制,优酷、腾讯视频、爱奇艺等内容提供商提供的流媒体服务并不能满足用户对清晰度的要求。对清晰度有要求的用户往往选择自己去网络获取高清晰度的媒体文件。这些文件以前被储存在网络中,但近几年国内网盘行业一片颓势,随着其它网盘的关闭和百度网盘的限速。越来越多的用户想收回自己对数据的控制权,而不再受制于网盘。再加上光纤的普及,带动了 NAS(Network Attached Storage)市场的增长。NAS 按字面简单说就是连接在网络上,具备资料存储功能的装置,因此也称为“网络存储器”。用户将视频、音频等媒体收回自己手里之后,市场上并没有一个适合国人使用的软件去更好的管理这些文件。所以设计一个家庭媒体管理软件,去解决这一困境,让用户观看储存在 NAS 中的视频媒体是能够拥有类似于优酷、爱奇艺的体验。

1.2 研究现状

目前,可用的家庭共享主要分为两种,一种是基于传统的文件共享协议 SMB、FTP 等方式手动管理文件,另一种是求助于专业的家庭媒体管理软件如 Plex、Emby、Jellyfin 等。但在实际体验中,这两者或多或少都有些不足。

1. 传统的文件管理

目前,内网中文件共享最常用的是 SMB 虽然在 Windows/Unix 中能够像访问本地文件一样访问 SMB 服务器上的文件,但是 SMB 也有占用网络带宽高、网络延迟大等问题,并且在 Android/iOS 等移动平台,SMB 协议的支持情况却不如人意。在这些平台要访问 SMB 服务器往往得借助于第三方软件。安卓端甚至要通过将 SMB 协议转换为 FTP 协议才能访问文件,而在这转换过程中很大程度上将会降低文件的传输效率。相较于 SMB,FTP 虽然支持公网、允许直接通过浏览器访问。但是安卓端的一些国产浏览器如腾讯的手机 QQ 浏览器、360 浏览器等却阉割了对 FTP 协议的支持。并且大部分 FTP 的作用供用户上传和下载文件,在线播放显然不是他的主要功能。虽然一些软件做到了直接播放 FTP 服务器上的内容,但是效果却不尽人意。在者 FTP 并不是一个安全的协议,近年来暴露出许多安全漏洞。因此对于家庭媒体服务器来说,FTP 甚至不如 SMB。

由此,可以看出传统的文件共享协议由于协议本身的局限性以及实现协议的各个软件自身标准不一,并不能保证各个设备之间具有统一的体验。所以 SMB、

FTP 并不适合做为一个家庭媒体文件共享的服务。再者，用管理文件的方式管理视频等媒体文件并不是一个聪明的选择。用户需要的是一个直观的体验，最好能够直接看到媒体文件的海报，分类，上映日期等信息。而且，如果能够自动获取字幕等信息，以及在各个不同设备之间拥有统一的使用逻辑，媒体管理服务将会有一个绝佳的体验。

2. 现存媒体管理软件

目前专注于媒体服务器的软件有 Plex、Emby 与 Jellyfin，其中 Plex 与 Emby 是商业软件，某些服务需要付费才能使用。Jellyfin 是开源的自由软件。这三个软件都实现了所需的功能但却各有各的缺陷。

Plex 与 Emby 都是商业软件，这两者都有强大的功能以及不错的体验，也都支持主流的平台。但是由于他们的功能过于繁杂，对性能的要求都比较高，在低性能的设备上（如我的群晖 DS119J）上流畅地运行。这两个软件都是国外公司开发，遵循的是国外用户的使用体验，虽然软件有中文的版本，但是在软件的某些地方，还保留着英文。而且这两个软件使用的媒体数据是 IMDB 的数据，并不是国内用户常用的豆瓣。因此这两个软件在一定程度上并不是非常契合国内用户需求。并且这两个软件在使用过程种或多或少地暴露出了某些问题。再者，这两个软件的免费授权都有一些限制，解锁高级用户却要价不菲，最便宜的都要 4.99 美元一个月。

相比于 Plex 与 Emby，Jellyfin 是免费的自由软件。他也是由国外用户主导开发，也有 Plex 和 Emby 不接地气的问题。Jellyfin 使用 C# 开发并不能很好地支持各大平台，并且性能上也略微有些问题。

在如今的市场中，找不到一个高性能，高效率，契合用户所需功能的软件。传统文件共享服务，实用但却并不优雅；三大媒体共享软件，优雅但却或多或少地有某些缺陷。找遍这三个软件的官网，并没有找到它们对 MIPS 架构的支持。而如今国产 CPU 获得进展最大的龙芯使用的恰恰是 MIPS 架构。得益于 Golang 超强的交叉编译能力，使得不需要拥有 MIPS 设备就可以编译出能够在 MIPS 平台的软件。不费吹灰之力解决了多平台支持能力。

本课题的目标是提供一个高性能、多平台支持的媒体管理软件。Go 语言是类似于 C 语言的编译型语言，天生有较高的性能与跨平台能力。这个软件将会有很大的竞争力，特别是在低性能设备上以及国产 CPU 平台将会是独一无二的。

1.3 论文结构

第一章：绪论。阐述课题的研究背景，分析媒体管理软件的研究现状并与市场上的现有软件进行比较，叙述了其在生活中的重要性以及该课题的研究目标。

第二章：相关技术与方法。介绍了项目开发过程中应用到的语言，网络爬虫实现方法和 ffmpeg 软件，并简述了系统开发过程中所使用的软件。

第三章：系统分析。使用用例图分析系统每个模块的需求，并对系统在技术、经济、操作方面的可行性进行简要分析。

第四章：系统设计。通过给出类图和类的功能表简要叙述每个类的实现方法，并给出了数据库设计的详细数据。

第五章：系统实现。展示项目实现对应功能的关键流程和代码，重点叙述了媒体智能管理和软件性能方面内容。

第六章：系统测试。针对用例对每个功能进行功能上的测试，其中重点测试了视频播放模块。

第七章：总结。结合系统的功能和使用的技术阐述了在系统开发过程中做了什么样的工作。

1.4 本章小结

在本章中，主要分析了家庭媒体管理软件在现代互联网化的家庭生活中的重要意义，分析了国内外 该类型软件的研究现状，提出了当前需要解决的问题以及对该问题的解决思路 and 希望达到的目标。

第二章 相关技术与方法

2.1 架构概述

媒体管理系统的开发应用了 Golang 语言，它是一种开发的一种静态强类型、编译型、并发型，并具有垃圾回收功能的编程语言。Go 语言程序编译为系统可执行文件运行，不依赖其他库、无需运行时环境。极大方便了软件的分发和用户的部署。针对 WEB 开发，Go 语言内置了丰富的 net 库，不依赖 Nginx、Tomcat 等 WEB 服务器便运行 WEB 程序。

在媒体软件的网页开发上采用了 B/S 结构，B/S 是 Browser/Server 的缩写（浏览器/服务器结构）。在这种结构中，媒体软件不需要开发任何用户界面，只需要通过 Web 浏览器向服务器发送请求，由 Web 服务器进行处理，并将处理结果逐级传回客户端。B/S 结构主要采用了结合浏览器的多种脚本语言 ActiveX 技术，简化了系统的开发、维护以及使用。

2.2 关键技术简介

在本系统中主要应用了 Golang、Vue、B/S 架构、SQLite 嵌入式数据库技术，其中在本系统特色的智能转码和信息获取采用了 ffmpeg 和网络爬虫技术。

2.2.1 Golang

Go 语言（又称 Golang）是 Google 开发的一种静态强类型、编译型、并发型，并具有垃圾回收功能的编程语言。

Go 语言的语法类似于 C，但是变量的声明不同。Go 语言支持垃圾回收。Go 语言的并行计算模型基于 Tony Hall 的通信排序过程（CSP）。采用类似模型的其他语言包括 Okam 和 Limbo，但也具有例如通道传输的管道模型功能。在 Go 语言的 1.8 版中打开了插件支持，这意味着现在可以从 Go 语言动态加载某些任务。

与 C++ 相比，Go 语言不包含诸如枚举，异常处理，继承，泛型，保证，虚拟函数等功能，但添加了诸如切片类型，构造函数，管道，垃圾收集函数，接口等语言级别的支持。Go 2.0 版本将支持泛型，并且对试剂的存在持否定态度，同时也捍卫自己免受提供类型继承的困扰。Go 语言的这些特性，极大地简便了媒体软件的开发过程。Go 语言于 2009 年 11 月发布之后，在 Google 产品中得到了广泛使用，并得到了广泛的反响，越来越完善的生态为软件提供了良好的支持。

2.2.2 爬虫技术

网络爬虫又称网络蜘蛛、网络蚂蚁、网络机器人等，它可以自动化浏览网络中的信息，当然浏览信息的时候需要按照我们制定的规则进行，这些规则我们称之为网络爬虫算法。使用 Go 语言可以很方便地编写出爬虫程序，从网络中检索出媒体软件所需的信息。

2.2.3 Vue

Vue.js 是一个 JavaScript 前端应用程序，旨在简化和改进网站。Vue 分析的核心部分是 MVC 模型中的视觉层，通过各节中的特殊节，可以同时带来最新信息并了解视觉与模型之间的相互作用。

组件是 Vue 最为强大的特性之一。为了更好地管理一个大型的应用程序，往往需要将应用切割为小而独立、具有复用性的组件。在 Vue 中，组件是基础 HTML 元素的拓展，可方便地自定义其数据与行为。

2.2.4 B/S

Browser-Server（浏览器/服务器）模型称为 B/S 模型，不同于 C/S 模型。用户不需要安装专门的软件，只需要一个浏览器与一个 Web 服务器即可与后端数据进行交互。即可以轻松地在不同平台上工作，该服务器可以使用任意平台的计算机，并支持主流的 Oracle 数据库，DB2，MySQL 和其各种数据库。B/S 模型简化了客户端的操作，该结构已经成为了当今软件应用的主流结构模式。

2.2.5 Google Speech-to-Text 语言识别 API

Google Speech-to-Text 是 google 基于神经网络开发的语音识别库，开发者可以方便地通过谷歌提供的 API，运用强大的神经网络模型将语音转换为文字。该 API 可为全球超过 120 种语言和语言变体的用户群提供有力的支持。以实现语音指令和控制、呼叫中心音频转录等多种使用场景。它可以将用户输入的语音指令转换成对应的问题指令，从而实现媒体文件的语言控制。

2.2.6 ffmpeg 转码

ffmpeg 是一个免费的开源软件，可以记录，录制视频和音频，转换音频和流音频，支持各种音频和视频的解码器。在世界上许多与视频有关的项目中，基本上都有 ffmpeg 或者其组件在背后默默支持。

2.3 开发工具

该系统所用到的开发工具如下：

1. Go-1.3
2. Golang
3. Sqlite3

4. Google-Chrome

2.4 本章小结

在本章中，列出了开发过程中所用的开发工具，并详细描述了该项目采用的 ffmpeg 转码技术和网络爬虫技术以及 Go 语言独特的特点和简要说明以及用于实现其他功能的技术。

第三章 系统分析

3.1 可行性分析

3.2.1 技术可行性

通过计算机程序配置系统，以达到项目的最低标准。软件开发涉及很多技术技能。在该系统中，为了避免技术集成引起的问题，无论是硬件安装还是系统专用技术这个角色，是我们一直使用的技术。// 语句不通畅

本系统采用 Go 语言和 SQLite 数据库分别作为前端与后端的开发，这两种工具能与计算机系统很好的兼容，因此该系统在技术上可行。

3.2.2 经济可行性

在系统的所有开发阶段，仅需一台可以运行 Linux 的计算机设备即可进行开发，无需投资开发项目。系统完成后，家庭媒体管理系统将会为生活提供便利。

3.2.3 操作可行性

该系统的目的是简化家庭媒体管理的要求。对于大多数用户而言，该系统提供了一个简单，易于理解，易于使用的界面。所以这是可行的。

3.2 需求分析

3.2.4 总体需求

表 3-1 总体需求

主要质量属性	详细要求
健壮性	在项目设计上要具有较好的容错能力
性能效率	在网络正常的前提下，要在规定时间内响应用户的请求
易用性	软件开发过程中形成的所有文档语言都是简洁、一致和易于理解的
安全性	保护软件免受意外或故意访问、使用、修改、破坏或泄漏的软件属性，其数据应集中存储在数据库服务器中
可扩展性	能方便的进行二次开发，满足对功能的扩展或提高并能提高相应的安全机制
兼容性	不易与其他软件发生冲突
正确性	正确的执行各个功能的需求，并达到每个模块相对应的功能要求

3.2.5 用例图分析

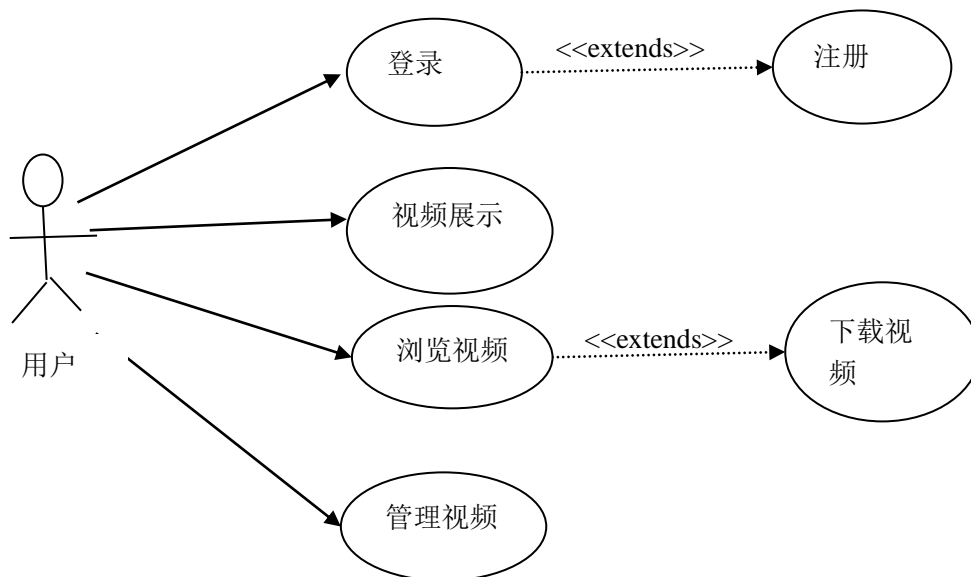


图 3.1 用户用例图

用户用例有用户登录、用户注册、用户浏览资源、用户播放资源、用户管理资源、用户下载资源六种。

表 3-2 用户信息注册用例

用例编号	User01	
用例名称	用户注册	
用例概述	用户通过此用例进行注册	
主要参与者	未经过初始化的设备	
前置条件	设备没有经过初始化	
基本事件流	步骤	活动
	A1	用户向系统发出注册请求
	A2	根据要求录入用户信息
	A3	判断数据库中用户名是否存在
	A4	将用户存入数据库，并且完成初始化
扩展事件流	1a	用户已经存在，提示重新录入用户
	1b	确认密码与前密码不同，提示重新录入密码
	1c	用户名含有不合法字符，提示重新录入用户名
	1d	用户点击取消，用例试用结束

表 3-3 用户登录用例

用例编号	User02
用例名称	用户登录
用例概述	用户通过该试用用例进行登录
主要参与者	以完成初始化的用户
前置条件	该设备已经完成初始化

	步骤	活动
基本事件流	A1	用户向系统发出登录请求
	A2	系统提示录入用户名以及密码
	A3	系统将收到的信息与数据库对比，判断其信息是否准确
	A4	用户名以及密码比对正确，用户成功登录
扩展事件流	1a	账户不存在，提示账户不存在，重新录入
	1b	密码错误，提示密码错误，重新录入
	1c	用户点击取消，用例试用结束

表 3-4 浏览视频用例

用例编号	User03	
用例名称	浏览视频	
用例概述	用户浏览页面上展示的视频	
主要参与者	已经初始化的设备	
前置条件	设备已经经过初始化并且用户登录成功	
基本事件流	步骤	活动
	A1	用户向系统发出主页请求
	A2	系统将数据库中的视频信息取出并发送给用户
扩展事件流	1a	目录下没有视频文件时，提示没有找到视频

表 3-5 播放视频用例

用例编号	User04	
用例名称	播放视频	
用例概述	用户播放选中的视频	
主要参与者	已经初始化的设备	
前置条件	设备已经经过初始化并且用户登录成功	
基本事件流	步骤	活动
	A1	用户点击选中的视频
	A2	系统在数据库中找出视频目录
	A3	系统将视频对应目录下的文件以文件流的形式发送给用户
扩展事件流	1a	该视频不存或者文件损坏在时，显示警告信息同时用例结束

表 3-6 下载视频用例

用例编号	User05	
用例名称	下载视频	
用例概述	用户下载选中的视频到本地	
主要参与者	已经初始化的设备	
前置条件	设备已经经过初始化并且用户登录成功	
基本事件流	步骤	活动
	A1	用户选中视频
	A2	用户点击下载按钮
	A3	系统将视频对应目录下的文件以文件流的形式发送给用户
扩展事件流	1a	该视频不存或者文件损坏在时，显示警告信息同时用例结束

表 3-7 设置用例

用例编号	User06	
用例名称	系统设置	
用例概述	用户修改系统设置	
主要参与者	已经初始化的设备	
前置条件	设备已经经过初始化并且用户登录成功	
基本事件流	步骤	活动
	A1	用户进入设置界面
	A2	系统展示设置选项
	A3	用户录入设置信息
	A4	系统将对应信息存入数据库
扩展事件流	1a	用户输入的信息含有非法字符，提示重新录入
	1b	用户点击取消按钮，该用例结束

3.3 本章小结

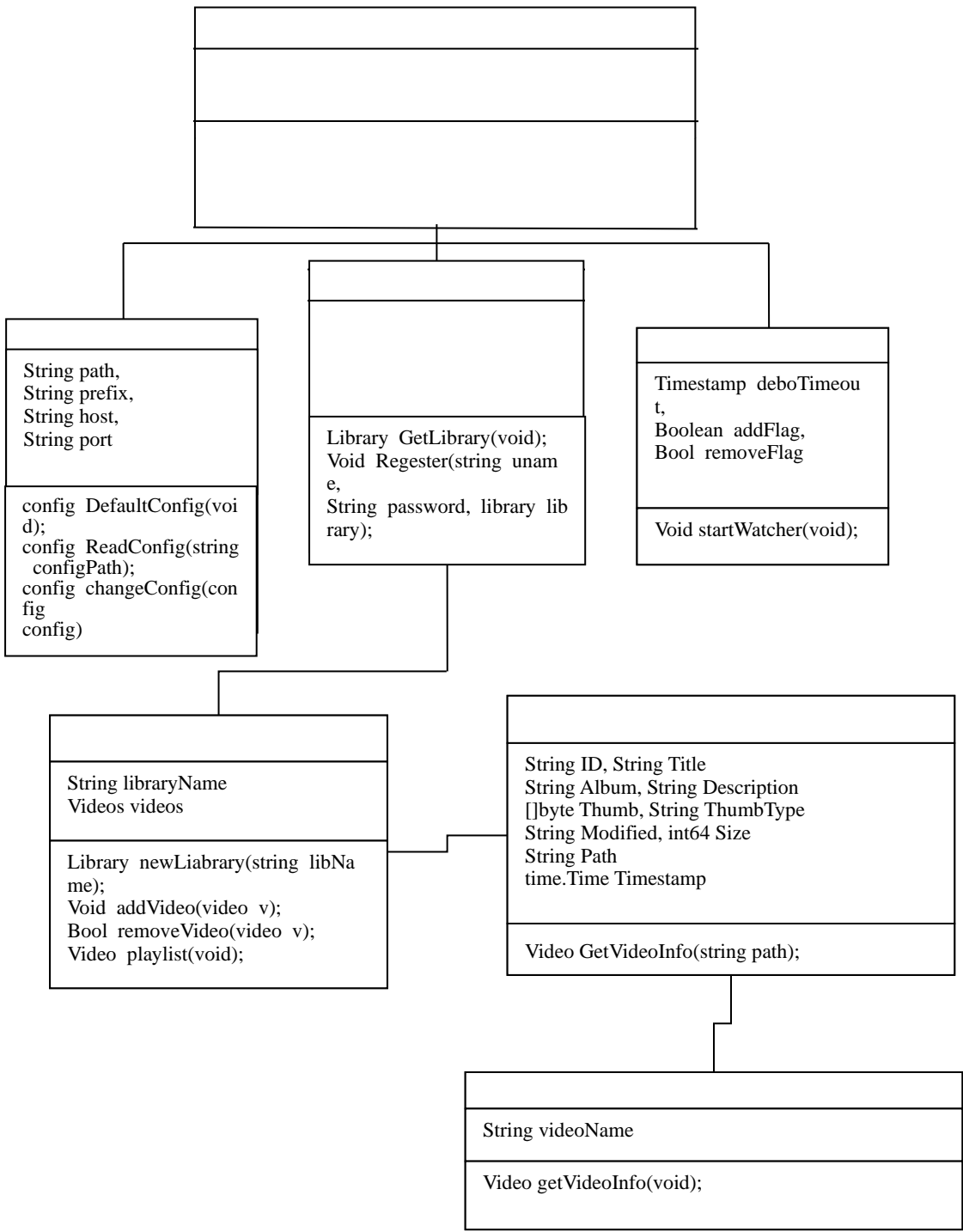
在可行性分析部分，对整个系统的需求以及它的经济性，技术可行性和操作性三个方面进行了简单分析。在需求分析中，使用了用例图和用例描述对执行系统中出现的相关任务进行分析。

第四章 系统设计

4.1 系统类分析

通过对项目进行系统分析，得出类间关系如图 4-1 所示：

图 4-1 系统类图



4.1.1 APP 类

APP（应用）程序的启动器与装载器，通过 APP 调用系统的各个模块。

表 4-1 APP 类功能表

类功能描述	程序启动器			
所在包名称	main			
继承对象	无			
实现对象	Application			
主要实现方法				
保护属性	方法名	输入参数	输出参数	方法功能描述
Public	Run	无	无	程序启动器
Public	NewApp	Config config	App	生成一个程序启动器

4.1.2 CONFIG 类

CONFIG(配置) 管理程序的配置文件，保存默认配置信息；修改中储存配置信息。

表 4-2 CONFIG 类功能表

类功能描述	管理程序配置信息			
所在包名称	config			
继承对象	无			
实现对象	AppConfig			
主要实现方法				
保护属性	方法名	输入参数	输出参数	方法功能描述
Public	DefaultConfig	无	config	默认情况下的配置信息
Public	ReadConfig	String configPath	config	读取配置文件，得到配置信息
Public	ChangeConfig	Config config	无	修改配置文件

4.1.3 USER 类

USER（用户）类用于保存用户在注册时输入的所有数据，并完成对用户的一系列业务操作。

表 4-3 USER 类功能表

类功能描述	管理程序配置信息			
所在包名称	user			

继承对象	无			
实现对象	AppUser			
主要实现方法				
保护属性	方法名	输入参数	输出参数	方法功能描述
Public	RegisterUser	String username, String password, Library library	bool	用户注册功能
Public	getLibrary	无	Library	读取用户的媒体库

4.1.4 WATCHER 类

WATCHER（监听）类用户定时监听用户文件库的修改，并将修改信息存入数据库。

表 4-4 WATCHER 类功能表

类功能描述	监听用户文件				
所在包名称	util				
继承对象	无				
实现对象	Watcher				
主要实现方法					
保护属性	方法名	输入参数	输出参数		方法功能描述
Public	StartWatcher	无	无		启动监听

4.1.5 LIBRARY 类

LIBRARY（库）类用来储存用户的所有媒体文件。

表 4-5 LIBRARY 类功能表

类功能描述	存储用户媒体文件信息			
所在包名称	user			
继承对象	无			
实现对象	UserLibrary			
主要实现方法				
保护属性	方法名	输入参数	输出参数	方法功能描述
Public	NewLibrary	String libraryName	library	生成新的媒体库
Public	AddVideo	Video video	无	往媒体库添加视频
Public	removeVideo	Video video	Bool	从库中删除

				视频
Public	PlayList	无	Video[]	返回文件库中的所有的视频

4.1.6 VIDEO 类

VIDEO 类存储媒体文件的详细信息，并对视频文件进行相应的操作。

表 4-6 VIDEO 类功能表

类功能描述	存储用户媒体文件信息			
所在包名称	media			
继承对象	无			
实现对象	MediaVideo			
主要实现方法				
保护属性	方法名	输入参数	输出参数	方法功能描述
Public	GetVideoInfo	String videoName	Video	调用爬虫获取视频信息并储存

4.1.7 SPIDER 类

SPIDER（网络爬虫）类通过爬虫程序对网上公开信息进行采集，并返回对应信息。

表 4-7 SPIDER 类功能表

类功能描述	存储用户媒体文件信息			
所在包名称	spider			
继承对象	无			
实现对象	Spider			
主要实现方法				
保护属性	方法名	输入参数	输出参数	方法功能描述
Public	GetVideoInfo	String videoName	Video	从网络采集 视频信息
Protect	getConnect	String url	String	获取网页
Protect	ParseHtml	无	Video	解析 html，获 取相应信息

4.2 关键业务设计

4.2.1 系统登陆

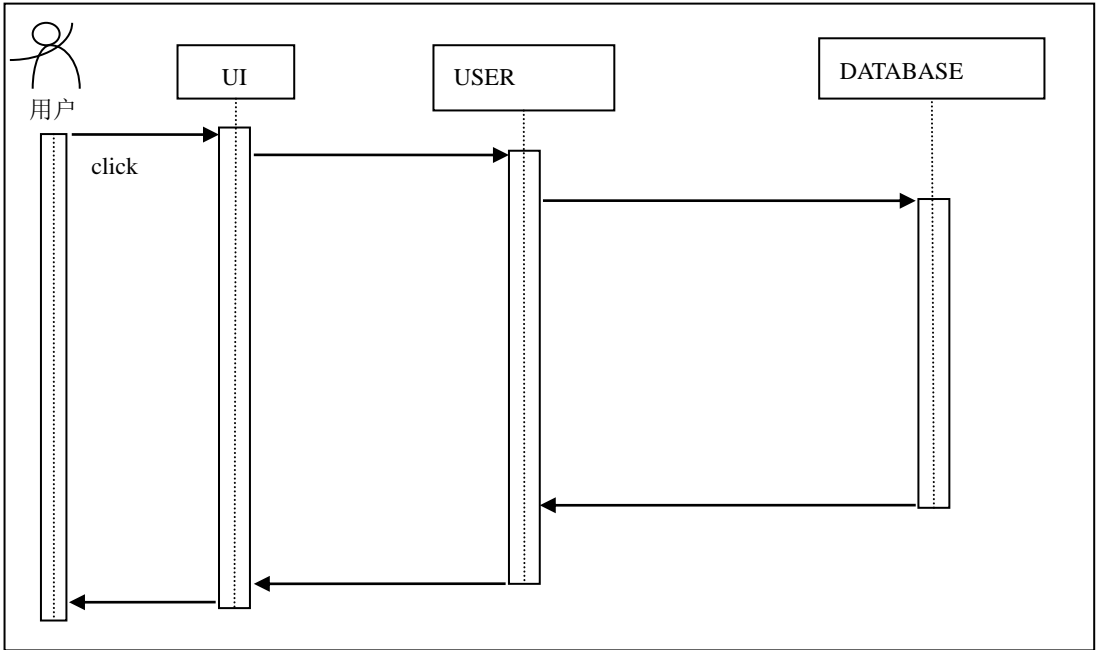


图 4.2 系统登录时序图

用户通过系统 UI 界面输入用户名和密码后,调用 User 类的 login 方法,login 方法会调用数据库的 select 方法,从数据库中查询到所需的账号信息,然后将账户返回给前端。

4.2.2 视频播放

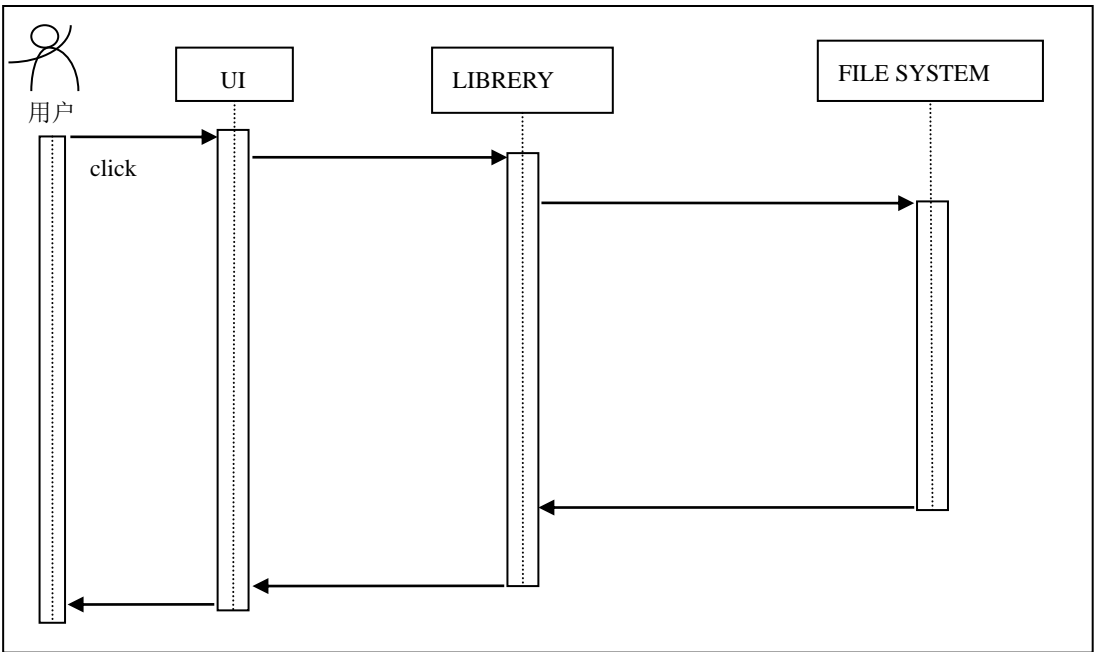


图 4.3 视频播放时序图

用户通过点击 UI 界面的视频,调用 APP 类的 Play 放法,然后从媒体库中调用媒体文件,最后在 UI 中播放。

4.3 数据库设计

4.3.1 概述

本系统设计以下四个数据库

- ① 用户表。用户基本信息表，用于储存用户基本信息，有用户名、密码、用户 ID、初始化信息四项数据。
- ② 路径表。用于储存媒体库的路径信息，有路径 ID、路径、前缀、用户名四项数据。
- ③ Session 表。用户储存用户登录的浏览器 session 信息，有 session_id、用户名、过期时间（TTL）三个数据项。
- ④ Video 表。用与储蓄视频的详细信息。

4.3.2 概念设计

用户表有用户名、用户 ID，用户密码，初始化信息。

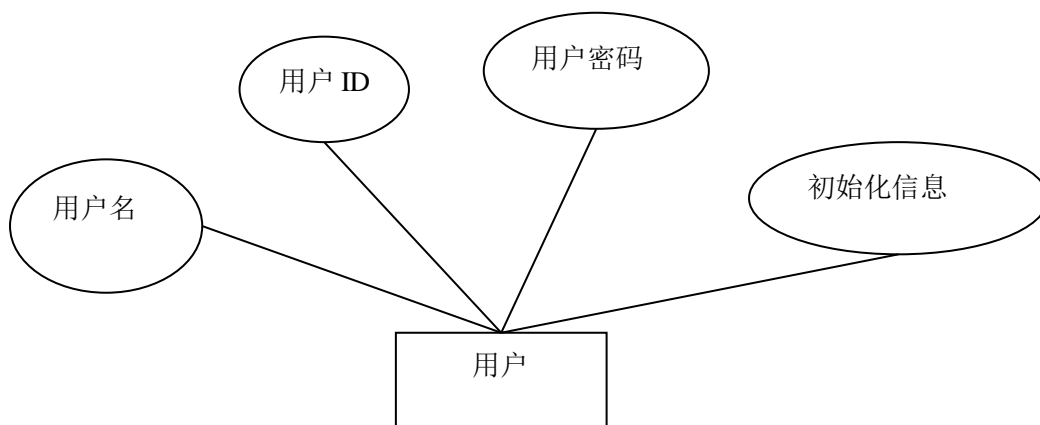


图 4.4 用户 ER 图

路径表有用户名、路径 ID，路径，路径前缀。

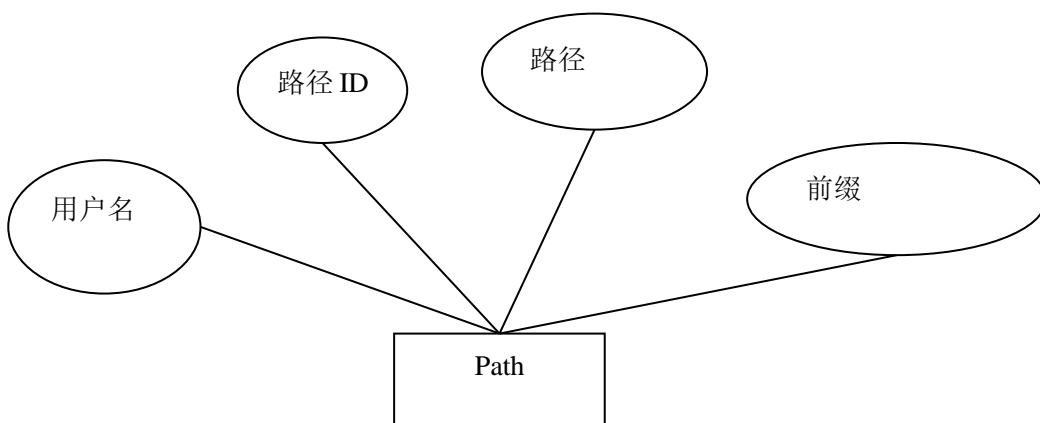


图 4.5 路径 ER 图

Session 有用 sessionId、TTL，用户名。

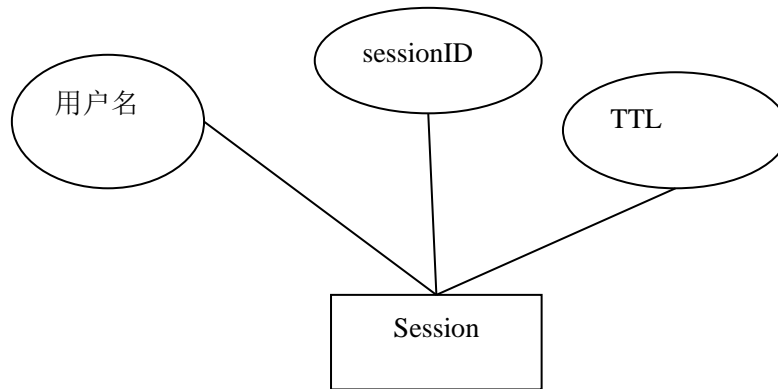


图 4.6 Session ER 图

视频表有视频 ID，用户名，视频标题，相册，视频描述，视频海报，海报文件类型，移动标记，大小，格式类型。

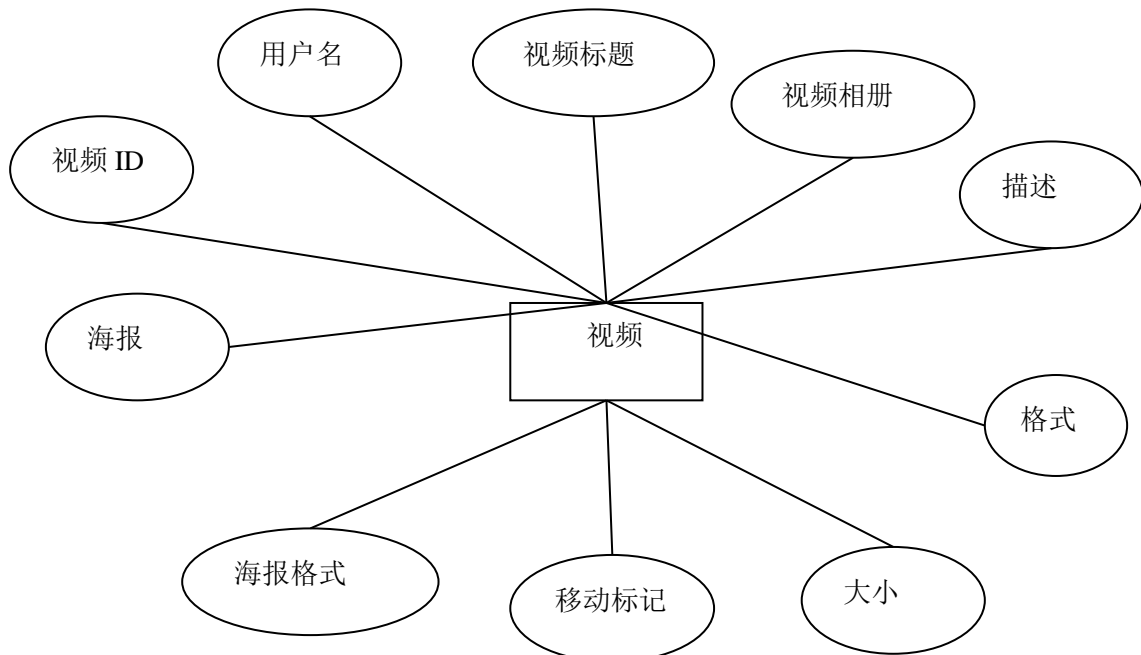


图 4.7 Video ER 图

数据库总体 ER 图如下：

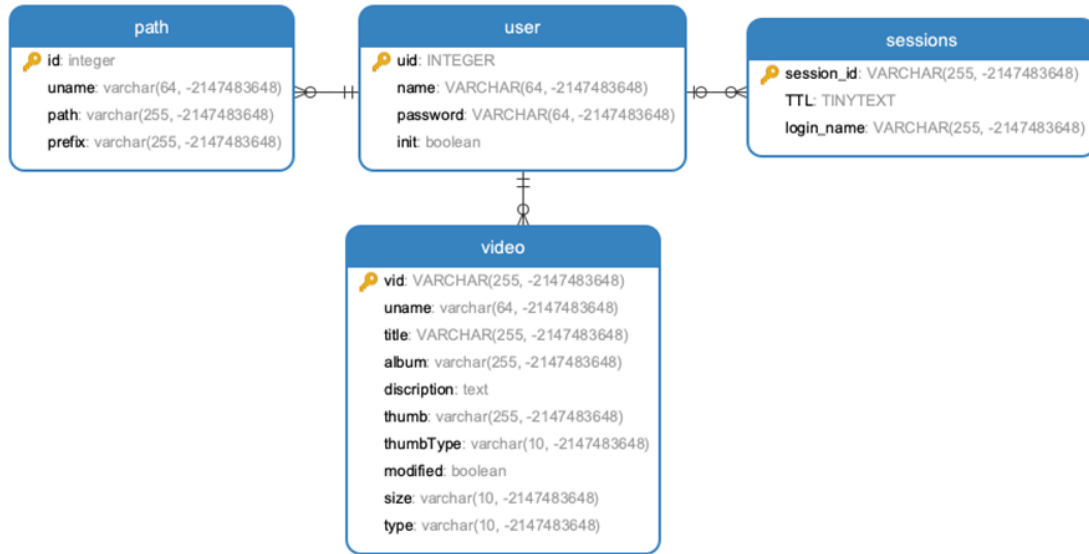


图 4.8 系统总体 ER 图

4.3.3 数据库表

数据库中使用到的数据库基本如表 N 所示

表 4-8 总体数据表

编号	表名	描述
1	user	用户注册表
2	session	session 状态表
3	path	媒体库路径表
4	video	视频信息表

用户表用来记录该系统用户的信息，对于家庭媒体管理软件来讲，一般一个系统只对应一个用户。该用户储存着用户的用户名，密码信息和初始化标记。用户名和密码是用来登录时的校验；初始化标记用来标记账户创建时是否经过了初始化。Session 表用来储存用户的 session 信息，通过设置 session 的 TTL 属性可以设定账户的存活时间。当在 session 有效范围内，用户不需要重复登录账户。路径表中储存着用户媒体库的路径，在视频展示时，系统通过读取路径的属性得到媒体文件在文件系统中的绝对路径。Video 表储存着每个视频文件的详细信息。

(1) 用户数据表

用户表在用户登录、注册、修改密码时使用。

表 4-9 用户数据表

字段	字段描述	类型/长度	约束	备注
uid	用户编号	int	主键	
name	用户名称	varchar(64)	NOT NULL	
password	用户密码	varchar(64)	NOT NULL	32 位 MD5
init	初始化信息	boolean	NOT NULL	

用户 ID 作为本表的主键来确定唯一的用户，用户名、密码和初始化标记都是非空字段。用户名和密码是用户登录时的唯一凭证。密码使用了 MD5 加密储存，有效地保证了密码的安全性。初始化使用了布尔值来标记用户是否经历了初始化。

(2) Session 表

session 表掌管着用户 session 在浏览器上的生命周期。

表 4-10 Session 数据表

字段	字段描述	类型/长度	约束	备注
session_id	Session 编号	varchar(255)	主键	UUID
TTL	过期时间	Tinytext	NOT NULL	Session 存活时间
login_name	所属用户	varchar(64)	外键	

Session 编号 使用随机生成的 UUID 作为唯一标识符；TTL 储存着 session 的存活时间不能为空；所属用户作为外键储存着该 session 所属的用户。

(3) 媒体库路径表

媒体库路径表储存着媒体库的路径。

表 4-11 媒体库路径数据表

字段	字段描述	类型/长度	约束	备注
id	编号	Int	主键	
uname	所属用户	varchar(64)	外键	
Path	路径	varchar(255)	NOT NULL	
Prefix	前缀	varchar(255)	NOT NULL	

编号字段作为主键标识一个媒体库，所属用户作为外键标识媒体库所属的用户，路径和前缀都不能为空。

(4) 视频信息表

视频表中储存着视频的所有详细信息。

表 4-12 视频信息数据表

字段	字段描述	类型/长度	约束	备注
vid	编号	int	主键	
uname	所属用户	varchar(64)	外键	
title	视频标题	varchar(255)	NOT NULL	
album	相册	varchar(255)		
discription	描述	text		
thumb	海报	varchar(255)		
thumbType	海报类型	varchar(10)		
size	视频大小	varchar(10)		
type	视频类型	varchar(10)		

vid 字段作为主键来标识一个视频，用户名作为外键标识视频所属的用户，除了视频名称之外，其他字段都可为空。

4.4 本章小结

本章内容主要介绍了媒体管理软件的系统类分析、部分关键业务设计、数据库设计三个部分。其中在数据库设计中给出了数据库的所有数据表。

第五章 系统实现

5.1 网页登陆

(1) 功能描述

网页登录是对注册用户身份验证的过程。只有用户身份验证通过，才能获得整个系统的操作权限，每个用户都有一个属于自己的独立账户名和密码。界面如下图所示：

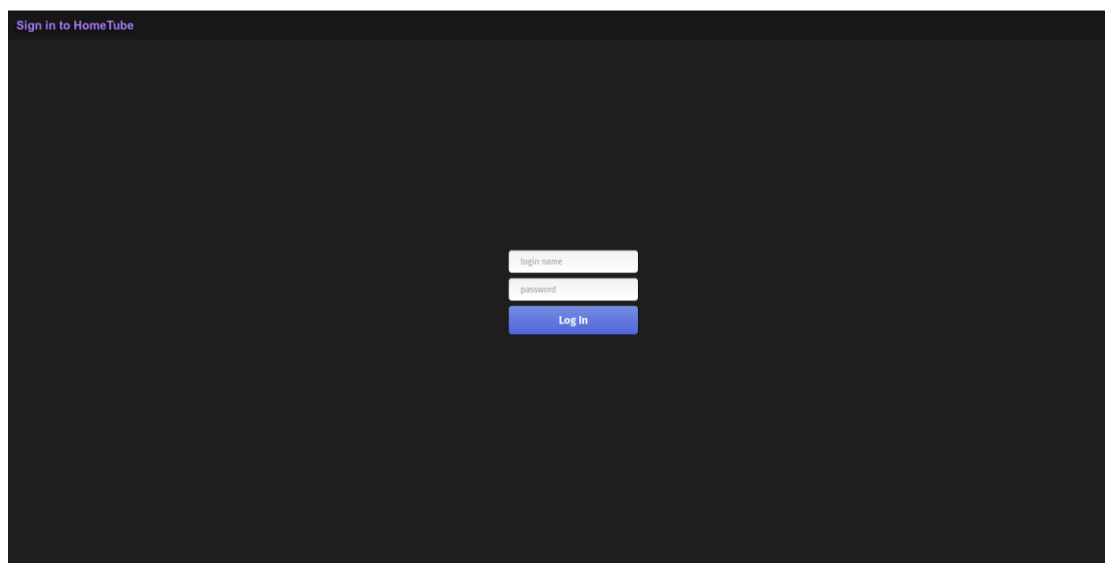


图 5.1 登录界面

(2) 实现流程

用户登录时，点击输入用户的用户名和密码，当用户输入不合法的字符或者某项为空时，前端会对输入变量做一个基础的判断，并提示用户输入的数据不合法。在输入完成并点击登录按钮后，登录按钮所绑定的方法会将所填写的数据与数据库中的信息进行核验，如果数据用于一致的对比结果，就会提示登录成功，否则提示登录失败。主要代码实现如下：

```
res, _ := ioutil.ReadAll(r.Body)
ubody := &defs.UserCredential{}

if err := json.Unmarshal(res, ubody); err != nil {
    sendErrorResponse(w, defs.ErrorRequestBodyParseFailed)
    return
}
```

```
if err := dbops.AddUserCredential(ubody.Username, ubody.Pwd); err != nil {  
    sendErrorResponse(w, defs.ErrorDBError)  
    return  
}  
  
id := session.GenerateNewSessionId(ubody.Username)  
su := &defs.SignedUp{Success: true, SessionId: id}  
  
if resp, err := json.Marshal(su); err != nil {  
    sendErrorResponse(w, defs.ErrorInternalFaults)  
    return  
} else {  
    sendNormalResponse(w, string(resp), 201)  
}
```

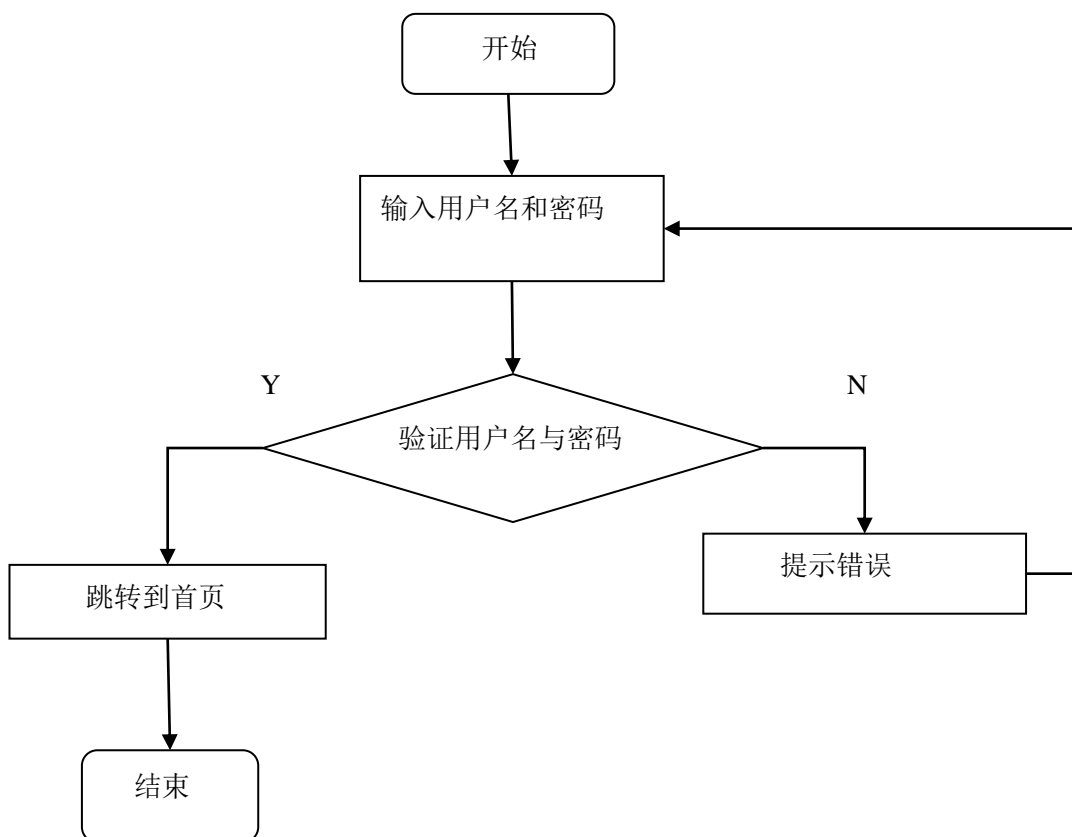


图 5.2 登录流程图

5.2 媒体智能管理

(1) 功能描述

在用户初始化界面或当用户添加新的媒体库路径时，扫描所给的目录的所有媒体文件，并标记重复文件。加入数据。并调用 ffmpeg 解码模块解析视频格式，尺寸等信息。同时启动爬虫，爬取视频的海报，描述等

(2) 实现流程

文件扫描是读取当前用户的配置文件，根据 config 的 path 属性和 prefix 属性拼凑出媒体文件的绝对路径。然后通过系统的 io 流目录下的所有媒体文件，并存入数据库，同时启动 ffmpeg 在后台对文件转码，分析等工作，并启动爬虫程序，在适当条件下从网络获取视频的海报，描述等信息。该功能主要实现部分代码如下：

```
lib.mu.Lock()
defer lib.mu.Unlock()
fp = filepath.ToSlash(fp)
d := path.Dir(fp)
p, ok := lib.Paths[d]
if !ok {
    return errors.New("media: path not found")
}
n := path.Base(fp) // 读文件
v, err := ParseVideo(p, n) //解析文件 解析失败则返回 err
if err != nil {
    return err
}
lib.Videos[v.ID] = v
log.Println("Added:", v.Path)
AddSpider(v) // 调用爬虫
return nil
```

其流程如图 5.3：

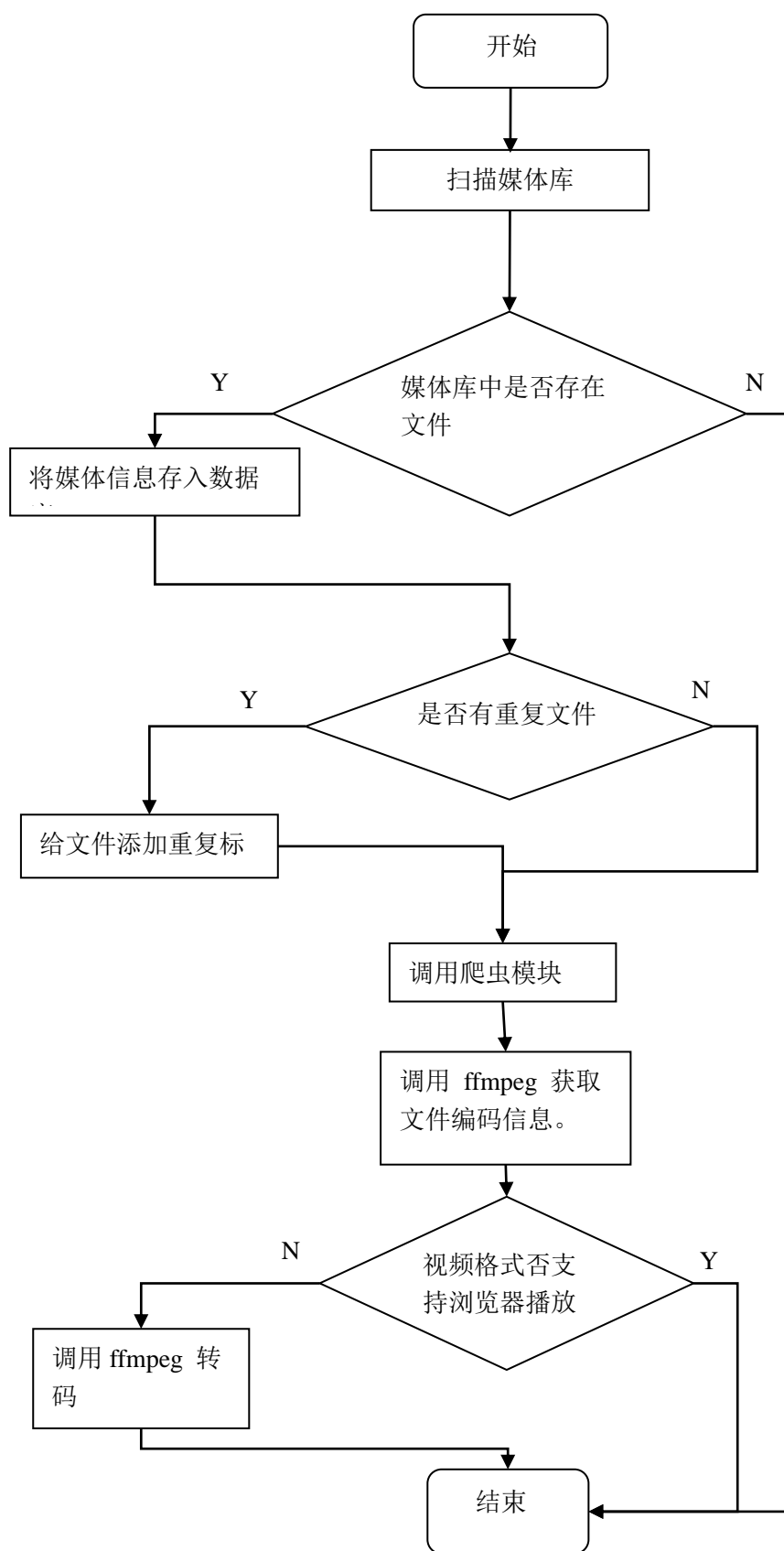


图 5.3 智能管理流程图

5.3 语音控制

(1) 功能描述

通过 html5 的 `getUserMedia` api 获取用户录入的音频，使用 Go 语言调用谷歌的 `speech-to-text` API，根据返回的信息，进行相应的操作，目前支持播放特定文件等操作。

(2) 实现流程

用户点击页面上的语音识别按钮，前端根据 html5 原生提供的 `getUserMedia` api 获取用户输入的音频。服务器调用谷歌的 API，根据返回的信息，进行进一步操作。主要流程如下：

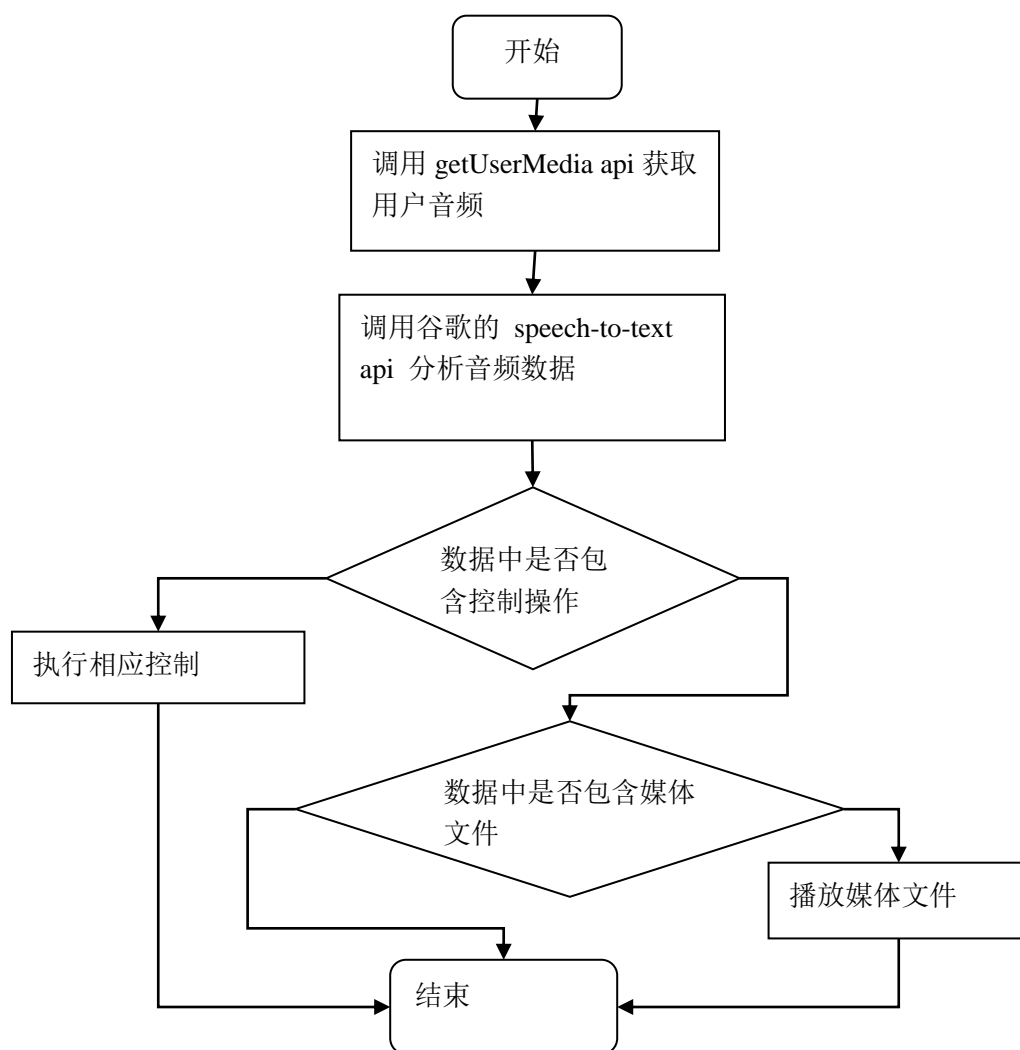


图 5.4 智能管理流程图

5.4 文件展示

(1) 功能描述

读取用户的媒体库，并在首页有序地展示出这些媒体文件。界面如下图：



图 5.5 在电脑端展示文件

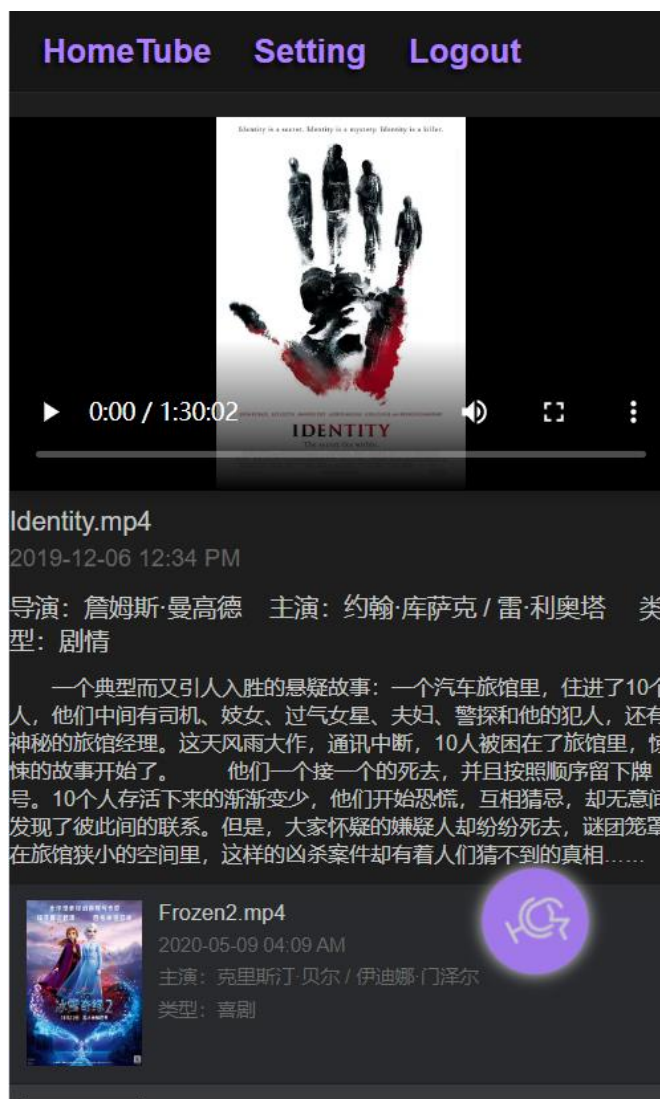


图 5.6 在移动设备上展示文件

(2) 实现流程

从数据库中读取每条媒体的信息，以数组的形式发送到前端，然后在前端对该数组进行遍历，取出每条媒体文件的信息。主要实现代码如下：

```
<div id="player">
  {{ if $playing.ID }}
    <video id="video" controls poster="/t/{{ $playing.ID }}"
src="/v/{{ $playing.ID }}.mp4"></video>
    <h1>{{ $playing.Title }}</h1>
    <h2>{{ $playing.Modified }}</h2>
    <p>{{ $playing.Description }}</p>
  {{ else }}
    <video id="video" controls></video>
  {{ end }}
</div>
<div id="playlist">
  {{ range $m := .Playlist }}
    {{ if eq $m.ID $playing.ID }}
      <a href="/v/{{ $m.ID }}" class="playing">
    {{ else }}
      <a href="/v/{{ $m.ID }}">
    {{ end }}
      
      <div>
        <h1>{{ $m.Title }}</h1>
        <h2>{{ $m.Modified }}</h2>
      </div>
    </a>
  {{ end }}
</div>
```

5.5 监听模块

(1) 功能描述

监听媒体库文件的改变，如果有增删等操作则及时通通知应用程序对问题库的更改做出反应。

(2) 实现流程

该功能主要定时读取媒体库中的文件信息，并于数据库中的媒体进行比对。如果发现异常，则通知系统做出修正。主要实现代码如下：

```

timer := time.NewTimer(debounceTimeout)
addEvents := make(map[string]struct{ })
removeEvents := make(map[string]struct{ })
for {
    select {
    case e := <-a.Watcher.Events:
        if e.Op&removeFlags != 0 {
            removeEvents[e.Name] = struct{ }{ }
        }
        if e.Op&addFlags != 0 {
            addEvents[e.Name] = struct{ }{ }
        }
        // reset timer
        timer.Reset(debounceTimeout)
    case <-timer.C:
        // handle remove events first
        if len(removeEvents) > 0 {
            for p := range removeEvents {
                a.Library.Remove(p)
            }
            // clear map
            removeEvents = make(map[string]struct{ })
        }
        // then handle add events
        if len(addEvents) > 0 {
            for p := range addEvents {
                _ = a.Library.Add(p)
            }
            addEvents = make(map[string]struct{ })
        }
        timer.Reset(debounceTimeout)
    }
}

```

5.6 设置

(1) 功能描述

设置是对用户使用系统功能进行设置的过程，用户在登录后需求有变化时，通过设置界面可以更改媒体库的路径、添加媒体库、更改代码等操作。界面如下图所示：

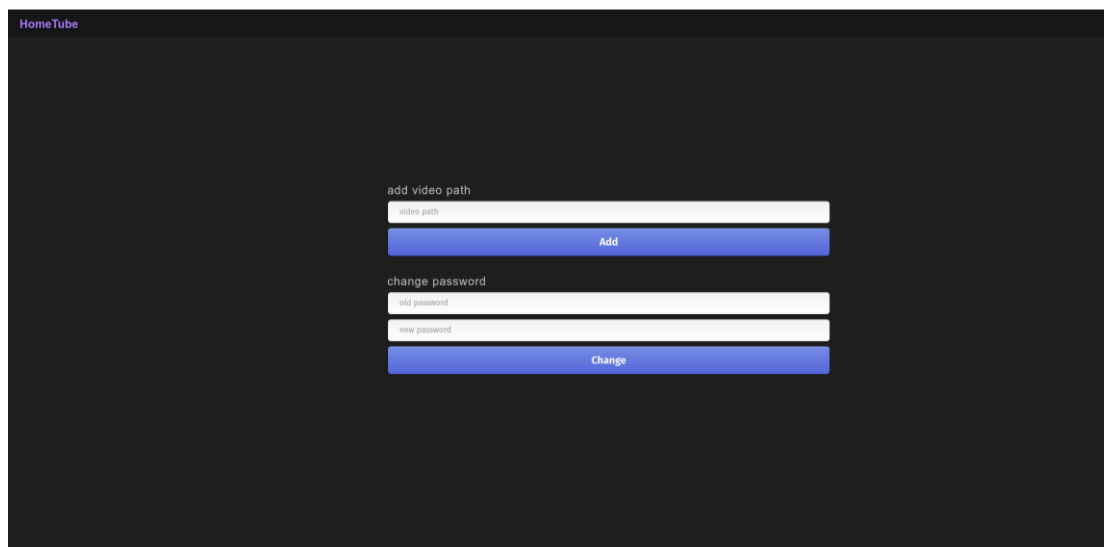


图 5.7 设置界面

(2) 实现流程

进入设置界面，用户可以更改媒体库的路径的设置新的密码，有两个不同的提交按钮分别绑定了不同的提交事件。在每个增加媒体库的事件中，媒体路径是必填项。如果用户填写的路径含有非法字符，或者经过校验路径不存在则会返回“路径不合法”和“该目录不存在”信息。以上操作完成之后，则完成系统设置。

在修改密码事件中，需要输入现有密码和新密码。如果现有密码为空则会提示“原密码不能为空”。同样的当新密码为空时，系统会提示“新密码不能为空”。当所有操作完成之后，系统首先调用 `validate()` 方法校验密码是否符合系统设定的规范。如果不符合，则会返回对应的提示。如当密码长度不足 6 位时，则会提示“密码长度至少为 6 位”。当新密码通过校验时，系统会校验输入的原密码是否和数据库中的一致。如果校验不通过，则会提示“原密码错误”；如果校验通过，则完成密码的修改。

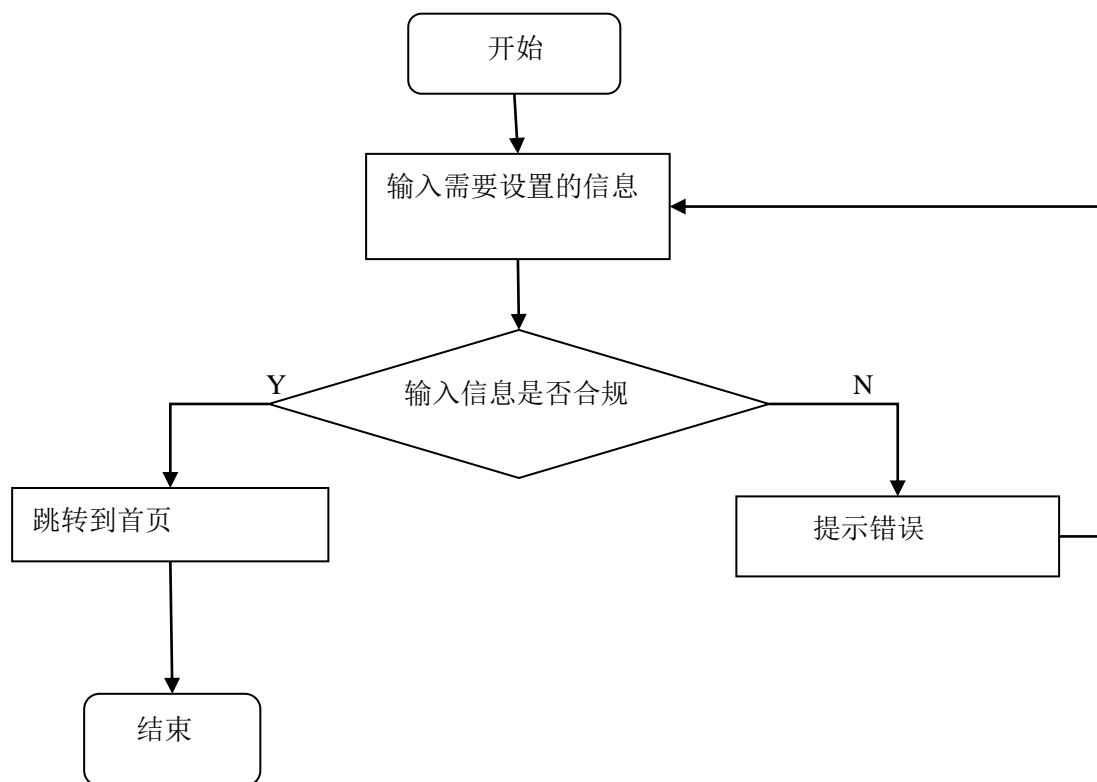


图 5.8 设置流程图

5.7 交叉编译

Go 语言能够简单地在一种平台上编译出能运行在体系结构不同的另一种平台上的程序，比如在 PC 平台（X86 CPU）上编译出能运行在以 ARM 为内核的 CPU 平台上的程序。只需要修改 \$GOARCH 变量指定目标平台处理器架构；\$GOOS 指定目标平台的操作系统，在任一平台即可编译出任何平台的二进制文件。

为了满足多平台的需求，编写了一个 python 脚本达到一次编译所有平台分发软件的目的。脚本代码如下：

```
# Build Go project for different platforms and create zip archive of project
# Muggle April 10, 2020
import os
import zipfile

def build(pkg, bin, env):
    src = 'github.com/muggle/HomeTube'
    x = os.system('{env} go build -o bin/{bin} {src}'.format(
        env=env,
        bin=bin,
```

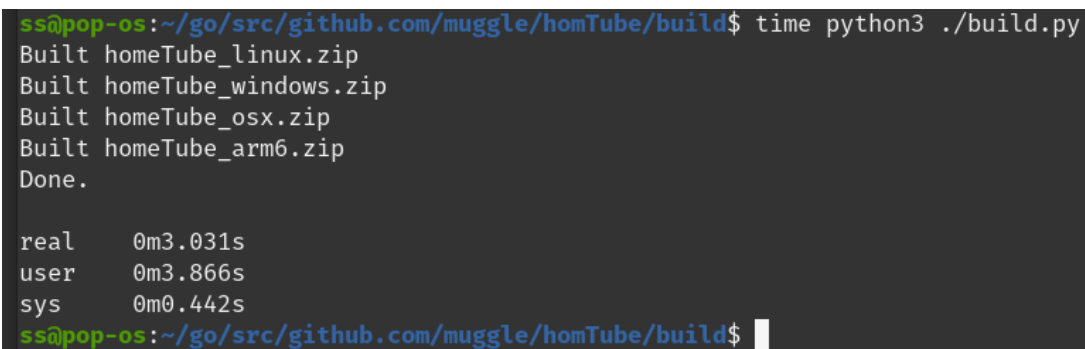
```
        src=src,
    ))
    if x != 0:
        print('Error building ' + pkg)
        return
    z = zipfile.ZipFile('bin/' + pkg, mode='w')
    z.write('bin/' + bin, arcname=bin)
    z.write('./config.json', arcname='config.json')
    z.write('./README.md', 'README.md')
    z.write('./videos/README.md', 'videos/README.md')
    for filename in os.listdir('./static'):
        z.write('./static/' + filename, 'static/' + filename)
    for filename in os.listdir('./templates'):
        z.write('./templates/' + filename, 'templates/' + filename)
    # cleanup executable
    os.remove('bin/' + bin)
    print('Built ' + pkg)

build(
    pkg="homeTube_linux.zip",
    bin="homeTube",
    env='GOOS=linux GOARCH=amd64',
)

build(
    pkg="homeTube_windows.zip",
    bin="homeTube.exe",
    env='GOOS=windows GOARCH=amd64',
)

build(
    pkg="homeTube_osx.zip",
    bin="homeTube",
    env='GOOS=darwin GOARCH=amd64',
)
```

```
build(  
    pkg='homeTube_arm6.zip',  
    bin='homeTube',  
    env='GOOS=linux GOARCH=arm GOARM=6',  
)  
print('Done.')
```



```
ss@pop-os:~/go/src/github.com/muggle/homTube/build$ time python3 ./build.py  
Built homeTube_linux.zip  
Built homeTube_windows.zip  
Built homeTube_osx.zip  
Built homeTube_arm6.zip  
Done.  
  
real    0m3.031s  
user    0m3.866s  
sys     0m0.442s  
ss@pop-os:~/go/src/github.com/muggle/homTube/build$
```

图 5.9 交叉编译

如图 5.9 所示，只需要极短的时间即可完成四个目标平台的编译。

5.8 本章小结

本章内容主要对系统的关键功能进行叙述，并列出每个功能的实现过程以及部分代码和流程图。

第六章 系统测试

6.1. 系统测试综述

系统测试已成为系统开发的重要组成部分，也受到了越来越多的关注。随着系统开发规模的不断扩大和复杂性的稳步提高，系统中出现的越来越多的漏洞使发现和修改这些软件的弱点变得更加困难，整个项目的测试任务也变得更加繁琐。因此，测试对于发现程序中日益增多的错误变得尤为重要。

从项目生命周期的角度来看，测试通常是指对整个生命周期中的程序进行测试。其关键是存在明确的测试对象，但是一旦在测试过程中发生错误，就很难保证测试的质量。目前，在项目的修复过程中成本相对交大。因此，根据软件的不同阶段进行严格检查形成的结果是软件开发过程的软件测试最理想的方法。

在测试系统之前，需要检查一些重要的资料，首先应当列出项目中需要测试的任务，然后再为测试做准备。通过查看相关的测试数据，了解系统测试。其中，白盒测试主要是测试系统的结构，它只测试软件的内部结构；也就是说，该测试方法在进行测试时无需像白盒测试那样考虑内部原理，而是从用户的角度来看待软件产品，以这种方式进行测试能够发现许多潜在的问题。因此，本项目选择了黑盒测试作为测试方法。

6.2. 测试用例

6.2.1 用户登录模块

表 6-1 用户注册测试

功能测试					
概述					
测试编号			001		
功能描述			用户注册		
功能 URL			/register		
用例目的			验证是否符合功能要求		
前提条件			进入注册界面		
测试操作					
编号	输入/动作	期望得到的响应	实际情况	是否 正确	错误 编号
1	不输入用户名	提示输入用户名	提示输入用户名	是	
2	用户名含有非法字符	提示用户名无效	提示用户名无效	是	

3	不输入密码	提示输入密码	提示输入密码	是	
4	输入密码少于 6 位	提示密码不安全	提示密码不安全	是	
5	输入已经注册过的用户名	提示用户已存在	提示用户已存在	是	
6	正确输入所有选项，点击注册按钮	注册成功，跳转到主页并登录成功	注册成功，跳转到主页并登录成功	是	

表 6-2 用户登录测试

功能测试					
概述					
测试编号			002		
功能描述			用户登录		
功能 URL			/login		
用例目的			验证是否符合功能要求		
前提条件			进入登录界面		
测试操作					
编号	输入/动作	期望得到的响应	实际情况	是否正确	错误编号
1	输入不存在的用户名	提示用户名不存在	提示输入用户名不存在	是	
2	输入正确的用户名和错误的密码	提示密码错误	提示密码错误	是	
3	输入正确的用户名以及密码	登录成功，跳转到首页	登录成功，跳转到首页	是	
4	不填写用户名	提示用户名不能为空	提示用户名不能为空	是	
5	填写用户名但不填写密码	不填写用密码	不填写用密码	是	

6.2.2 文件扫描模块

表 6-3 用户注册测试

功能测试	
概述	
测试编号	003
功能描述	文件扫描
功能 URL	
用例目的	验证是否符合功能要求
前提条件	用户经过初始化
测试操作	

编号	输入/动作	期望得到的响应	实际情况	是否正确	错误编号
1	通过设置添加一个含有媒体文件的路径	首页展示出了该路径下的媒体文件并可正常播放。	首页展示出了该路径下的媒体文件并可正常播放。	是	
2	通过设置添加一个不含媒体文件的路径	首页没有新加入的视频	首页没有新加入的视频	是	
3	点击首页上的任意一个视频	视频正常播放	视频正常播放	是	

6.2.3 视频智能化管理

表 6-4 视频智能化管理测试

功能测试					
概述					
测试编号			004		
功能描述			智能化管理		
功能 URL					
用例目的			验证是否符合功能要求		
前提条件			用户已经登录		
测试操作					
编号	输入/动作	期望得到的响应	实际情况	是否正确	错误编号
1	媒体库中有相同视频	显示有重复视频并提示用户删除	显示有重复问题，同时提示具体路径以及删除按钮	是	
2	扫描一个媒体库	显示出视频的分类，简介等具体信息	媒体信息显示正常	是	
3	往媒体库中添加一个 vp9 编码格式的 MP4 视频	浏览器正常播放	视频被转码为 h264 格式，浏览器正常播放	是	

视频转码中 vp9 格式视频最终被转为 h264 格式视频。


```
libavformat 58. 29.100 / 58. 29.100
libavdevice 58. 8.100 / 58. 8.100
libavfilter 7. 57.100 / 7. 57.100
libavresample 4. 0. 0 / 4. 0. 0
libswscale 5. 5.100 / 5. 5.100
libswresample 3. 5.100 / 3. 5.100
libpostproc 55. 5.100 / 55. 5.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from './Frozen 2.1080P-LanYu.mp4':
Metadata:
  major_brand      : isom
  minor_version    : 512
  compatible_brands: isomiso2avc1mp41
  encoder          : Lavf58.29.100
Duration: 01:44:04.25, start: 0.000000, bitrate: 3493 kb/s
Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 1920x1080 [SAR 1:1 DAR 16:9], 3360 kb/s, 23.98 fps, 23.98 tbr, 11988 tbn, 47.95 tbc (default)
Metadata:
  handler_name     : VideoHandler
Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 128 kb/s (default)
Metadata:
  handler_name     : SoundHandler
At least one output file must be specified
```

图 6.1 视频成功转码

6.2.4 视频播放模块

表 6-5 视频播放测试

功能测试					
概述					
测试编号			005		
功能描述			文件扫描		
功能 URL					
用例目的			验证是否符合功能要求		
前提条件			用户已经登录		
测试操作					
编号	输入/动作	期望得到的响应	实际情况	是否正确	错误编号
1	点击一个视频文件。	视频文件正常播放，且可以全屏	视频文件正常播放，且可以全屏	是	
2	在移动设备上点击一个视频	视频文件正常播放，且可以全屏	视频文件正常播放，且可以全屏	是	

视频播放模块的测试流程如下：

- ① 点击一个视频，视频自动播放，且能够自由控制。

图 6.2 视频播放



图 6.3 笔记本全屏播放

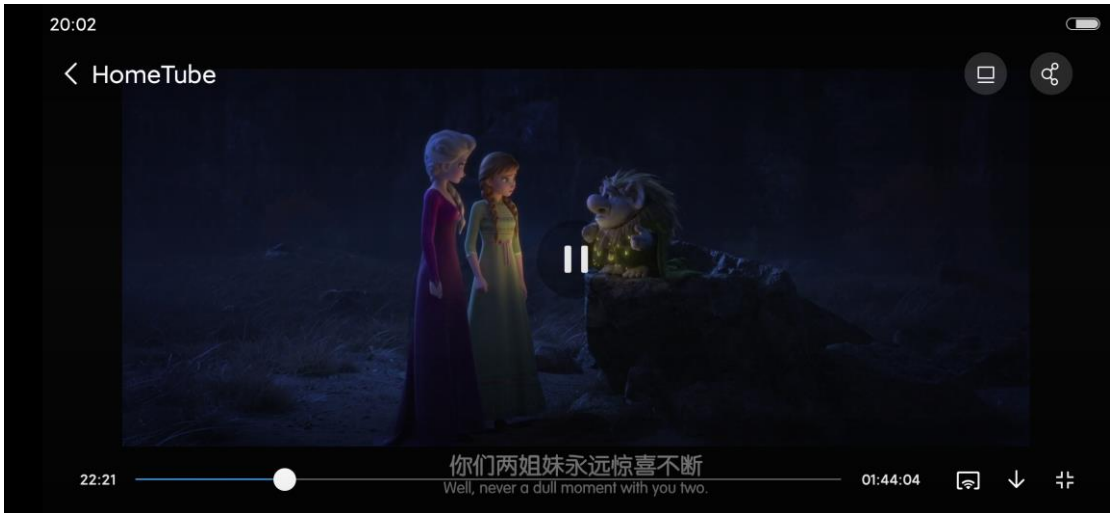


图 6.4 移动端视频播放

6.2.5 语音控制测试

表 6-6 文件监听测试

性能测试					
概述					
测试编号			006		
功能描述			语音控制测试		
功能 URL					
用例目的			验证语音控制模块是否符合要求		
前提条件			用户已经登录		
测试操作					
编号	输入/动作	期望得到的响应	实际情况	是否符合要求	错误编号
1	点击语音识别按钮，喊出 冰雪奇缘	播放冰雪奇缘	等待一段时间后播放冰雪奇缘	是	
2	点击语音识别按钮，喊出删除冰雪奇缘	删除冰雪奇缘源文件	等待一段时间后媒体文件被删除	否	001

由于 HTML5 的 getUserMedia api 采样率问题，所以识别的准确率并不高。

6.2.6 文件监听模块

表 6-7 文件监听测试

功能测试					
概述					
测试编号			007		
功能描述			监听模块		
功能 URL					
用例目的			验证是否符合功能要求		
前提条件			用户经过初始化		
测试操作					
编号	输入/动作	期望得到的响应	实际情况	是否 正确	错误 编号
1	删除已经识别的任意一个媒体文件	首页上没有出现该视频的信息	首页上没有出现该视频的信息	是	
2	往已经识别的目录下添加一个媒体文件	首页上出现该文件信息	刷新后，首页出现该媒体文件信息	是	

6.2.7 性能测试

表 6-8 系统性能测试

性能测试					
概述					
测试编号			008		
功能描述			性能测试		
功能 URL					
用例目的			验证性能是否符合要求		
前提条件					
测试操作					
编号	输入/动作	期望得到的响应	实际情况	是否符合要求	错误编号
1	分析软件大小并与同科类软件比较	该软件空间占用小于同类软件	软件空间占用仅为同类软件的 14%	是	
2	测试软件启动速度并与同类软件比较	启动速度快于同类软件	启动速度明显快于同类软件	是	

性能测试平台为 ubuntu 20.04 AMD64、CPU 为 i5-6300HQ，运行内存 12GB，系统盘为 100G NVMe SSD。选用的对比软件为 Plex 版本 1.19.3.2764、Emby 版本 4.4.2.0。测试流程如下：

(1) 对比三个软件安装包空间占用，结果如下：

表 6-9 软件占用空间

软件	占用空间 (MB)
Plex	91.1
Emby	69.5
HomeTube	20.2

可见，在应用软件分发体积上，HomeTube 有绝对的优势

(2) 利用 Linux 自带的 Time 函数测试软件从启动到启动完成所需要的时间（如图 6.4 所示）。由于三款软件都不会自动退出，因此在软件启动完成后用快捷键 ctrl D 退出，由此可能会产生较大的误差。得到的结果如下：

表 6-10 软件启动时间对比

软件	启动时间 (S)
Emby	5.137
Plex	3.152
HomeTube	0.353

```
ss@pop-os:~/go/src/github.com/muggle/homTube/build/bin$ time ./homeTube
2020/05/09 17:23:37 Local server: http://0.0.0.0:34249
2020/05/09 17:23:37 /home/ss/Videos
2020/05/09 17:23:37 Added: /home/ss/Videos/Frozen 2.1080P-LanYu.mp4
2020/05/09 17:23:37 Added: /home/ss/Videos/致命ID.Identity.2003.1080p.BluRay.x264.英语中字-RARBT.mp4
^C
real    0m0.353s
user    0m0.008s
sys     0m0.007s
ss@pop-os:~/go/src/github.com/muggle/homTube/build/bin$
```

图 6.5 time 命令测试软件启动时间

可见在软件启动时间上，HomeTube 也有较大的优势。

6.3. 本章小结

本章重点介绍了选择的测试方法和使用该方法的原因，以及用于测试目的的测试用例。由于软件开发周期过短，以及缺乏软件开发方面的经验，软件设计的

并不完善。为了发现软件中存在的问题及软件功能的不足,需要不断地进行测试。并针对测试发现的问题一步一步着手完善。

第七章 总结

在这个系统的整个开发过程中，我遇到了很多技术层面的问题。首先，在开发语言方面，我一直在 Java 和 Go 语言之间犹豫不决。尽管 Go 语言的语法习惯还是他编译的属性都深得我意，但是毕竟之前做过的项目都是用的 Java。我没有用过 Go 语言开发过大型的项目，因此对自己是否有能力独自完成一个项目抱着很大的质疑。但是考虑到我这个项目的目标平台是多系统和低性能设备的。虽然二者都有非常不错的跨平台性能，但是 Java 作为一个由虚拟机运行的语言，多多少少会有一些性能的损失。而 Go 语言则直接将代码编译成可执行的二进制文件，对低性能设备十分友好。并且正值疫情居家区间，我有大量的空闲时间。因此我想利用这些时间重新认识认识这门由谷歌公司推出的新生语言。于是这个基于 Go 语言的项目就诞生了。

在代码编写之前，我充分调查了一个媒体管理软件需要具备的基本功能。在疫情居家区间，我也成为了一个迫切需要一款好的媒体管理软件的用户；并且切身使用了市面上的几款家庭媒体管理软件。作为用户我切身体会到了这个类型软件需要解决的痛点以及市面上软件的不足。视频管理中最重要的是不是华丽的界面和繁多的功能，它的核心是视频共享。只有做好了多设备，多平台的共享，一个家庭媒体管理软件才有存在的意义。普通家庭中，并没有多少人会去购买专业的 NAS 设备，他们所拥有的设备可能仅仅是一个五年前的台式、一台十年前的笔记本、更甚至是一台淘汰的安卓手机。而现在网络上流行的媒体管理软件往往对性能要求颇高，往往不能很好地在以上设备中运行。因此努力提高软件性能成了我工作的重点。考虑各个，在系统的整体设计中采用了 B/S 结构，前端主要由 Go 语言自带的模板类驱动 HTML 实现。并致力于优化代码，减少不必要的性能消耗。在软件的编写过程中，由于需要调用 Go 语言调用 C 语言的 ffmpeg 库。库中的各项依赖在 Windows 系统中解决起来特别繁琐，因此在软件的开发中期不得不将开发环境移动到了 Ubuntu 下。

软件最终实现了大部分预期的功能，拥有着不错的性能和较为智能的媒体管理能力，能够做到清理重复文件等简单工作，软件能够较为实时地获取网络信息，对媒体文件进行精准分类和信息补充。但遗憾的是，本软件并没有开发出客户端程序，目前只能依靠并不可靠的浏览器运行。二是没有充分利用 ffmpeg，没有使用 ffmpeg 进行推流。导致播放视频一些高画质视频时显得比较吃力。如果能够改善这一点软件的实用性将会有很大提高。

参考文献

- [1] it 之家. 网易网盘关闭部分入口, 国内网盘行业 “三足鼎立” [EB/OL]. <https://www.ithome.com/0/446/817.htm>, 2019-9-23/2019-12-27.
- [2] 什么值得买. 群晖中国区 CEO 陈予建访谈: NAS 市场仍在快速增长, 中小企业用户成为增量主力[EB/OL]. <https://post.smzdm.com/p/az59p475/>, 2018-10-16/2019-12-27.
- [3] 维基百科. 服务器消息块[EB/OL]. <https://zh.wikipedia.org/wiki/伺服器訊息區塊/>, 2018-8-30/2019-12-29
- [4] nurdletech. Securing FTP using SSH[EB/OL]. <https://nurdletech.com/linux-notes/ftp/ssh.html>
- [5] 什么值得买. 原创 篇三: 家庭多媒体中心软件 Emby 介绍[EB/OL]. <https://post.smzdm.com/p/735222/>, [2018-7-26]/2019-12-29
- [6] 高素春. UML 在面向对象软件开发中的应用[J/OL]. 河南科技:1-6[2019-12-28]. <http://kns.cnki.net/kcms/detail/41.1081.T.20191213.1548.002.html>.
- [7] 刘秋香, 刘振伟. 浅析几款主流的 UML 建模工具[J]. 电脑知识与技术, 2018, 14(32):245+253.
- [8] Andrawos M, Helmich M. Cloud Native Programming with Golang: Develop microservice-based high performance web apps for the cloud with Go[M]. Packt Publishing Ltd, 2017.
- [9] James Whitney, Chandler Gifford, Maria Pantoja. Distributed execution of communicating sequential process-style concurrency: Golang case study[J]. , 2019, 75(3).
- [10] Togashi N, Klyuev V. Concurrency in Go and Java: performance analysis[C]//2014 4th IEEE International Conference on Information Science and Technology. IEEE, 2014: 213-216.
- [11] 麦冬, 陈涛, 梁宗湾. 轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版), 2017(07):58-59.
- [12] Hong P. Practical Web Design: Learn the fundamentals of web design with HTML5, CSS3, Bootstrap, jQuery, and Vue.js[M]. Packt Publishing Ltd, 2018.
- [13] 万玛宁, 关永, 韩相军. 嵌入式数据库典型技术 SQLite 和 BerkeleyDB 的研究[J]. 微计算机信息, 2006 (01Z): 91-93.
- [14] 王珊, 萨师煊. 数据库系统概论(第 5 版)[M]. 高等教育出版社, 2014-09
- [15] medium. 5 Tips To Speed Up Golang Development With IntelliJ Or Goland. [EB\OL]. <https://medium.com/@kepererry/5-tips-to-speed-up-golang-development-with-intellij-or-goland-6646110e9c5e>, 2019-5-2/2019-9-27
- [16] 邓正良. 基于 FFmpeg 和 SDL 的视频流播放存储研究综述[J]. 现代计算机, 2019(22):47-50.

致 谢

自始至终都没有料到，我最后的学生时光大部分居然是在家中度过的。这篇论文从题目的拟定，到框架结构，数据收集，创作整理，以及最后的反复修改都是在家里完成的。居家生活总是使人变得懒散，在这个过程中主要的动力来源都靠老师的鞭挞。在写作工程中我遇到了很多始料未及的困难，让我困惑不安，但这些问题都在老师详尽的指导下解决。我也希望能够尽自己最大的努力，写出一篇有用的论文，做出一个有用的作品。在一波三折下，论文最终完成。在不断地完善和修改的过程中，让我深深体会到了耕耘和收获之间密不可分的因果关系。

除了要衷心地感谢我的指导老师以外，还要感谢所有在大学期间传授我知识的老师，每一位老师的教导都是我完成这篇论文的基础。同时还要感谢网络上无私奉献的大神们，学校并没有开设 Go 语言的课程，我是在他们一篇篇文章的基础上才对 Go 语言从生疏到熟练。也需要感谢 ffmpeg 的开发团队，他们开发的高效软件是这篇论文的必要组成成分，也是在我们生活中最常用到的软件之一。也许你随手打开一个视频，都有它在后台默默贡献着。在大学生活中，朝夕相处的同学是最难忘的回忆，我们一起欢笑，一起悲伤，互相帮助，共同进步。我要感谢同学们给我的关心和留给我美好的回忆，没有他们我的生活将会失去很多色彩。

四年的磨砺让我学到了很多，四个月的居家生活让我思考了很多。2020 让我见识了很多从来没有见识过的事情，也许未来并不太平，但是我会继续努力，不懈前行。毕业绝对不是终点，而是通向未来的起点。

基于 Golang 的家庭媒体管理软件的设计与实现

刘得根¹

(1. 江西理工大学, 软件工程学院, 江西 南昌 330013)

摘 要: 高清电视设备逐渐大众化。用户对影视资源的质量有了更高的追求。但市场上却缺少一个用于管理这些杂乱视频的软件。系统使用 B/S 架构, 后台由谷歌的 Go 语驱动, 前端是由 Vue.js 构建的响应式界面。主要专注于视频的跨平台播放与管理。在初步扫描文件时, 通过辨别文件的 MD5 码来识别重复的文件。使用了网络爬虫技术在网络的公开数据中采集媒体等信息。在不打扰用户的前提下, 使用了 ffmpeg 对不适合浏览器播放的视频进行转码, 使软件有舒适的体验。

关键词: Golang; 媒体管理; B/S 架构; ffmpeg

Abstract: High definition TV equipment is becoming more and more popular. Users have a higher pursuit for the quality of film and television resources. But there is a lack of software on the market to manage the clutter. The system uses B / S architecture, the background is driven by Google's go language, and the front end is driven by Vue.js Build a responsive interface. It mainly focuses on the cross platform broadcast and management of video. During the initial scanning of documents, identify the duplicate documents by identifying the MD5 code of the documents. We use the technology of web crawler to collect media and other information in the open data of the network. On the premise of not disturbing the user, we use ffmpeg to transcode the video which is not suitable for the browser, so that the software has a comfortable experience.

Key words: Golang; Media Management Software;ffmpeg;B/S

0 前言

现阶段用户对视频清晰度的要求越来越高, 再加上国内各大视频网站版权竞争所产生的乱象, 以及 NAS 的兴起。越来越多的用户选择通过某些渠道获得高质量的媒体文件储存在家庭 NAS 中, 用户将视频、音频等媒体收回自己手里之后, 市场上并没有一个适合国人使用的软件去更好的管理这些文件。

传统的管理方法当中如通过 SMB、FTP 等传输协议进行跨平台访问时体验并不友好。用户面对的只是一个个冰冷的二进制文件, 并不能直观便捷地管理文件。如果将这些文件以精致的界面汇集起来, 再通过网络

爬虫采集一些海报、简介等基本的信息。用户的使用体验将会有质的飞跃。

媒体管理软件采用谷歌公司提供的 Go 语言构建, Go 语言是一门精简完善的编程语言具有精简, 易学等特点。为快速构建软件提供了可能性。为了实现不同编码格式的兼容性, 使用了 ffmpeg 在 CPU 闲暇时对媒体文件编码进行转换。

1 相关技术

1.1 架构概述

媒体管理系统的开发应用了 Golang 语言, 它是一种开发的一种静态强类型、编译型、并发型, 并具有垃圾回收功能的编程语言。Go 语言程序编译为系统可执行文件运

行，不依赖其他库、无需运行时环境。极大方便了软件的分发和用户的部署。针对 WEB 开发，Go 语言内置了丰富的 net 库，不依赖 Nginx、Tomcat 等 WEB 服务器便运行 WEB 程序。

在媒体软件的网页开发上采用了 B/S 结构，B/S 是 Browser/Server 的缩写（浏览器/服务器结构）。在这种结构中，媒体软件不需要开发任何用户界面，只需要通过 Web 浏览器向服务器发送请求，由 Web 服务器进行处理，简化了系统的开发、维护以及使用。

1.2 Golang

Go 语言是 Google 开发的一种静态强类型、编译型、并发型，并具有垃圾回收功能的编程语言。与 C++ 相比，Go 语言不包含诸如枚举，异常处理，继承，泛型，保证，虚拟函数等功能，但添加了诸如切片类型，构造函数，管道，垃圾收集函数，接口等语言级别的支持。于 2009 年 11 月发布之后，在 Google 产品中得到了广泛使用，并得到了广泛的反响，越来越完善的生态为软件提供了良好的支持。

1.3 网络爬虫

网络爬虫可以自动化浏览网络中的信息，当然浏览信息的时候需要按照我们制定的规则进行，这些规则我们称之为网络爬虫算法。使用 Go 语言可以很方便地编写出爬虫程序，从网络中检索出媒体软件所需的信息。

1.4 ffmpeg

ffmpeg 是一个免费的开源软件，可以记录，录制视频和音频，转换音频和流音频，支持各种音频和视频的解码器。在世界上许

多与视频有关的项目中，基本上都有 ffmpeg 或者其组件在背后默默支持。

2 系统设计

2.1 系统模块

系统中的模块包括：初始化模块、用户登陆模块、爬虫模块、文件扫描模块、文件展示模块、语音识别模块。

2.2 数据库设计

系统主要涉及到以下四个数据表：

- ① 用户表。用户基本信息表，用于储存用户基本信息，有用户名、密码、用户 ID、初始化信息四项数据。
- ② 路径表。用于储存媒体库的路径信息，有路径 ID、路径、前缀、用户名四项数据。
- ③ Session 表。用户储存用户登录的浏览器 session 信息，有 session_id、用户名、过期时间（TTL）三个数据项。
- ④ Video 表。用于与储蓄视频的详细信息。

3 系统关键功能实现

3.1 初始化

初始化是对整个系统进行初始化的过程，在初次打开软件时供用户注册和输入媒体库所用。

3.1.1 系统初始化

在初始化过程中，输入用户名、密码和媒体库的路径，系统将用户名保存在数据库中作为本系统的登录管理用户。媒体库作为系统的资源库也被保存在数据库中。初始化界面如下：

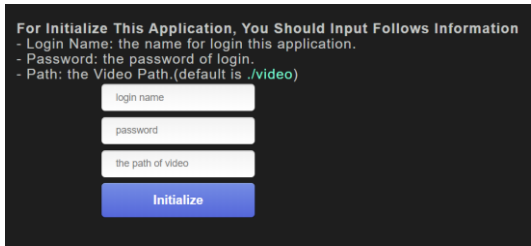


图 1 系统初始化界面

3.1.2 用户登录

系统初始化完成之后，将会跳转到用户登录界面，以验证唯一的用户名和密码。登录后跳转到文件展示页面。主要界面如下：

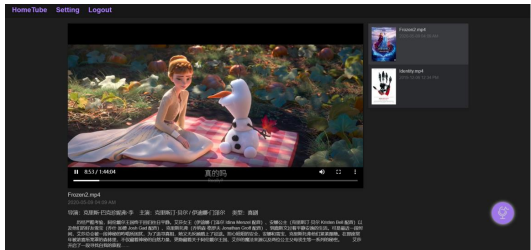


图 2 登录后的文件展示界面

用户登录时，点击输入用户的用户名和密码，当用户输入不合法的字符或者某项为空时，前端会对输入变量做一个基础的判断，并提示用户输入的数据不合法。在输入玩成并点击登录按钮后，登录按钮所绑定的方法会将所填写的数据与数据库中的信息进行核验，如果数据用于一致的对比结果，就会提示登录成功，否则提示登录失败。

3.2 媒体智能管理

在用户初始化界面或当用户添加新的媒体库路径时，扫描所给的目录的所有媒体文件，并标记重复文件。加入数据。并调用 ffmpeg 解码模块解析视频格式，尺寸等信息。同时启动爬虫，爬取视频的海报，描述等

3.2.1 媒体文件的智能管理

在用户初始化界面或当用户添加新的媒体库路径时，扫描所给的目录的所有媒体

文件，并标记重复文件。加入数据。并调用 ffmpeg 解码模块解析视频格式，尺寸等信息。同时启动爬虫。

3.2.2 网络爬虫

通过由 Go 语言所编写的网络爬虫，采集网络上公开的信息。如视频海报、视频分类，视频简介等。网络爬虫主要使用了 Go 语言原生的 Http 工具包获取网页的 html 数据，然后再通过正则表达式与第三方的 Goquery 包提取所需的数据。所获取的数据保存成 json 文件，储存在媒体库中与媒体文件同名的文件夹中。

3.2.3 ffmpeg 转码

在存在视频编码并不能支持浏览器播放时，系统会调用视频工具 ffmpeg 对视频的编码格式进行转换。

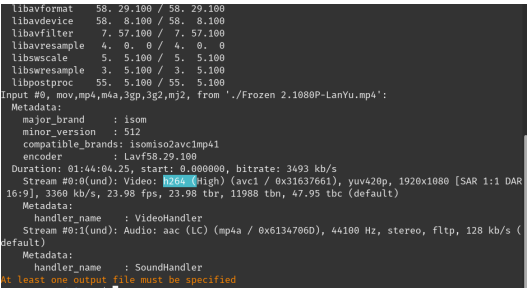


图 3 视频编码转为浏览器支持的 h264

3.3 交叉编译

Go 语言能够简单地在一种平台上编译出能运行在体系结构不同的另一种平台上的程序，比如在 PC 平台（X86 CPU）上编译出能运行在以 ARM 为内核的 CPU 平台上的程序。只需要修改 \$GOARCH 变量指定目标平台处理器架构；\$GOOS 指定目标平台的操作系统，在任一平台即可编译出任何平台的二进制文件。

4 总结与展望

软件最终实现了大部分预期的功能，拥有着不错的性能和较为智能的媒体管理能力，能够做到清理重复文件等简单工作，软件能够较为实时地获取网络信息，对媒体文件进行精准分类和信息补充。但遗憾的是，本软件并没有开发出客户端程序，目前只能

依靠并不可靠的浏览器运行。二是没有充分利用 ffmpeg，没有使用 ffmpeg 进行推流。导致播放视频一些高画质视频时显得比较吃力。如果能够改善这一点软件的实用性将会有很大提升。

参考文献

- [1] it 之家. 网易网盘关闭部分入口，国内网盘行业“三足鼎立”[EB/OL]. <https://www.ithome.com/0/446/817.htm>, 2019-9-23/2019-12-27.
- [2] 什么值得买. 群晖中国区 CEO 陈予建访谈：NAS 市场仍在快速增长，中小企业用户成为增量主力[EB/OL]. <https://post.smzdm.com/p/az59p475/>, 2018-10-16/2019-12-27.
- [3] 维基百科. 服务器消息块[EB/OL]. <https://zh.wikipedia.org/wiki/伺服器訊息區塊>, 2018-8-30/2019-12-29
- [4] nurdletech. Securing FTP using SSH[EB/OL]. <https://nurdletech.com/linux-notes/ftp/ssh.html>
- [5] 什么值得买. 原创 篇三：家庭多媒体中心软件 Emby 介绍[EB/OL]. <https://post.smzdm.com/p/735222/>, [2018-7-26]/2019-12-29
- [6] 高素春. UML 在面向对象软件开发中的应用[J/OL]. 河南科技:1-6[2019-12-28]. <http://kns.cnki.net/kcms/detail/41.1081.T.20191213.1548.002.html>.
- [7] 刘秋香, 刘振伟. 浅析几款主流的 UML 建模工具[J]. 电脑知识与技术, 2018, 14(32):245+253.
- [8] Andrawos M, Helmich M. Cloud Native Programming with Golang: Develop microservice-based high performance web apps for the cloud with Go[M]. Packt Publishing Ltd, 2017.
- [9] James Whitney, Chandler Gifford, Maria Pantoja. Distributed execution of communicating sequential process-style concurrency: Golang case study[J]. , 2019, 75(3).
- [10] Togashi N, Klyuev V. Concurrency in Go and Java: performance analysis[C]//2014 4th IEEE International Conference on Information Science and Technology. IEEE, 2014: 213-216.
- [11] 麦冬, 陈涛, 梁宗湾. 轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版), 2017(07):58-59.
- [12] Hong P. Practical Web Design: Learn the fundamentals of web design with HTML5, CSS3, Bootstrap, jQuery, and Vue.js[M]. Packt Publishing Ltd, 2018.
- [13] 万玛宁, 关永, 韩相军. 嵌入式数据库典型技术 SQLite 和 BerkeleyDB 的研究[J]. 微计算机信息, 2006(01Z): 91-93.
- [14] 王珊, 萨师煊. 数据库系统概论(第 5 版)[M]. 高等教育出版社, 2014-09
- [15] medium. 5 Tips To Speed Up Golang Development With IntelliJ Or Goland.[EB/OL]. <https://medium.com/@kepererry/5-tips-to-speed-up-golang-development-with-intellij-or-goland-6646110e9c5e>, 2019-5-2/2019-9-27
- [16] 邓正良. 基于 FFmpeg 和 SDL 的视频流播放存储研究综述[J]. 现代计算机, 2019(22):47-50.

独创性声明

本人郑重声明，所呈交的设计（论文）是我本人在指导教师指导下进行的研究工作并取得的研究成果。尽我所知，除了中文特别加以标注和致谢的地方外，设计（论文）中没有抄袭、剽窃其他人已经发表或撰写的研究成果，也不存在为获得江西理工大学或其他教育机构的学位或证书所使用过的材料。为本设计（论文）的完成给予过的帮助、做出过的贡献均已在设计（论文）中做了明确的说明并表示了谢意。

签名：

日期：

关于设计（论文）使用授权的说明

本人完全了解江西理工大学有关保留、使用毕业设计（论文）的规定，即：学校有权保留送交设计（论文）的原件，允许设计（论文）被查阅和借阅；学校可以公布设计（论文）的全部或部分内容，可以影印、缩印或其他复制手段保存设计（论文）。即使是保密的设计（论文）解密后也应遵守项规定。

签名：

日期：