学校代码：12331

****

毕 业 设 计 说 明 书

支持移动设备的在线IDE的设计与实现

|  |  |
| --- | --- |
| **姓 名：** | 栾锟 |
| **学 号：** | 170507250117 |
| **指导教师：** | 张茂红（副教授） |
| **学 院：** | 数据科学与计算机学院 |
| **专 业：** | 计算机科学与技术（春季高考） 1班 |
| **完成日期：** | 2021年6月12日 |

山东女子学院毕业设计作者声明

本人声明：本人完成的毕业设计及呈交的毕业设计说明书是本人在导师指导下取得的成果。对前人及其他人员对毕业设计的启发和贡献已在毕业设计说明书中做出了明确的声明，并表示了谢意。除了特别加以标注和致谢的地方外，本人的毕业设计方案中不包含其他人和其他机构已经实现、发表或者撰写过的成果。

本人同意学校根据《中华人民共和国学位条例暂行实施办法》等有关规定保留本人的毕业设计说明书并向国家有关部门或资料库送交本说明书或其电子版，允许本说明书被查阅或借阅；本人授权山东女子学院可以将本毕业设计说明书的全部或者部分内容编入有关数据库进行检索，可以采用影印、缩印或者其它复制手段汇编本说明书。

作者签名： 日期： 年 月 日

指导教师签名： 日期： 年 月 日

# 摘要

本设计通过对互联网上现有的代码编辑工具进行调研分析，发现当前互联网上的代码编辑工具仍有缺陷。在离线编辑方面，大部分移动端的代码编辑工具，仅支持本地编译且占用大量本地空间，空间利用率低。而在在线编辑方面，编译工具仅支持PC端，对于移动端并没有合适的编译工具，导致移动端编程不方便。因此，设计并开发了一套基于支持移动设备的在线IDE系统。

本设计借助于Linux Shell的交互性，实现编译与运行工作，并得到相应的报错信息与输出显示。首先，前端根据用户信息定位用户路径，然后通过向服务器传递不同的请求，来使服务器执行不同的操作。服务器端在数据交互方面，使用subprocesss命令shell运行shell脚本进行相应的操作并通过Python IO文件流，将前端及与shell交互获取的数据写入到对应的文件中，再通过读取这些文件来获取shell的数据。经运行验证表明，本设计的思路具有可行性，达到了预期设计目标。

关键词：IDE；Linux Shell；Python; Subprocesss

# Abstract

This design investigates and analyzes the existing code editing tools on the Internet, and finds that the current code editing tools on the Internet still have defects. In terms of offline editing, most mobile code editing tools only support local compilation and take up a lot of local space, resulting in low space utilization. In terms of online editing, the compilation tool only supports the PC terminal, and there is no suitable compilation tool for the mobile terminal, which makes programming on the mobile terminal inconvenient. Therefore, a set of online IDE system based on supporting mobile devices was designed and developed.

This design uses the interactivity of Linux Shell to realize compiling and running work, and get the corresponding error message and output display. First, the front end locates the user path based on user information, and then sends different requests to the server to make the server perform different operations. In terms of data interaction on the server side, use the subprocesss command shell to run shell scripts for corresponding operations and through the Python IO file stream, write the data obtained by the front end and the shell interaction to the corresponding files, and then read these files to obtain the data of the shell.The operation verification shows that the idea of this design is feasible and achieves the expected design goal.

**Key words:** IDE; Linux Shell; Python; Subprocesss

目录

[1 绪论 1](#_Toc74119423)

[1.1 选题背景 1](#_Toc74119424)

[1.2 研究意义 1](#_Toc74119425)

[1.3 国内外研究现状 1](#_Toc74119426)

[1.4 论文组织结构 2](#_Toc74119427)

[2 相关开发技术及工具 3](#_Toc74119428)

[2.1 Flask 3](#_Toc74119429)

[2.2 Linux shell 3](#_Toc74119430)

[2.3 uni-app 3](#_Toc74119431)

[2.4 开发环境 4](#_Toc74119432)

[2.4.1 HbuilderX 4](#_Toc74119433)

[2.4.2 Nginx服务器 4](#_Toc74119434)

[2.4.3 Pycharm 4](#_Toc74119435)

[3 系统分析 5](#_Toc74119436)

[3.1 用户需求分析 5](#_Toc74119437)

[3.2 系统功能分析 6](#_Toc74119438)

[3.2.1云存储功能 6](#_Toc74119439)

[3.2.2代码编辑功能 7](#_Toc74119440)

[3.3 可行性分析 7](#_Toc74119441)

[3.3.1 经济可行性 8](#_Toc74119442)

[3.3.2 技术可行性 8](#_Toc74119443)

[4 系统设计 9](#_Toc74119444)

[4.1 总体设计 9](#_Toc74119445)

[4.2 类设计 10](#_Toc74119446)

[4.3 云存储设计 11](#_Toc74119447)

[4.3.1 云存储目录实现原理 11](#_Toc74119448)

[4.3.2 云存储设计 11](#_Toc74119449)

[4.4 具体功能设计 12](#_Toc74119450)

[4.4.1 后端设计 12](#_Toc74119451)

[4.4.2 登录功能设计 14](#_Toc74119452)

[4.4.3 注册功能设计 15](#_Toc74119453)

[4.4.4 目录管理功能设计 15](#_Toc74119454)

[4.4.4 代码编辑功能设计 16](#_Toc74119455)

[5 系统实现 18](#_Toc74119456)

[5.1 登录界面 18](#_Toc74119457)

[5.2 注册界面 18](#_Toc74119458)

[5.3 用户目录 19](#_Toc74119459)

[5.4 编译运行功能 21](#_Toc74119460)

[5.4.1 代码编辑页面 21](#_Toc74119461)

[5.4.2 报错页面 21](#_Toc74119462)

[5.4.3 控制台页面 22](#_Toc74119463)

[6 系统测试 23](#_Toc74119464)

[6.1 测试目的 23](#_Toc74119465)

[6.2 测试方法 23](#_Toc74119466)

[6.3 功能测试 23](#_Toc74119467)

[6.3.1 黑盒测试 23](#_Toc74119468)

[6.3.2 测试结果 27](#_Toc74119469)

[7 总结与展望 28](#_Toc74119470)

[7.1 总结 28](#_Toc74119471)

[7.2 展望 28](#_Toc74119472)

[参考文献 29](#_Toc74119473)

[附录A后端代码 31](#_Toc74119474)

[附录B shell脚本 41](#_Toc74119475)

[致谢 43](#_Toc74119476)

# 1 绪论

## 1.1 选题背景

随着网络技术的不断发展，特别是在云计算、云操作系统等相继问世之后。人们对于应用的需求发生了改变，不再像传统PC端一样，只追求应用的系统性，全面性，而是根据移动端、云端的优势开始追求方便性，轻量级，用更少的资源来完成具体的工作，追求更高的效率[1]。

另一方面，随着现代教育的普及，中小学开始普及少儿编程，而高校也开始将程序设计加入人才培养方案，促使越来越多的学生开始有着编写代码的需求，又因为现代社会对移动设备有巨大的依赖性，使得更多的学生选择使用移动设备编写代码，而非选择传统的PC编写环境[2]。

## 1.2 研究意义

目前市场现存的代码编写工具中，支持移动端的编写工具仍然较少，大多都是传统App，大量占用本地空间，内存分配与管理不合理，没有良好的跨平台性。因此，研究和设计一款在线代码编写工具都具有一定的现实意义[3]。

## 1.3 国内外研究现状

目前国内外云端编译环境已经初步成型，国外不乏一些优秀的在线IDE系统，Analyzing the learning process (in Programming) by using data collected from an online IDE就对于课程引入在线IDE系统做了深刻的讨论[4]。特别是国内外的几大算法竞赛网站，如Acwing，蓝桥云课，PTA，Leetcode几大在线评测网站逐步上线了在线调试功能，可以看出云编译环境市场巨大。特别是Acwing网站已经开发出了AC Editor这个PC端在线IDE系统工具，并且有相当高的完成度，支持云存储，编译，报错，智能提示等功能。

另一方面，随着移动端的发展，移动端化本来就是互联网发展的一大趋势，小程序市场还没有成型的在线编译小程序，因此市场巨大，有着无限发展前景。

## 1.4 论文组织结构

本设计基于轻量级Flask web框架与uni-app技术，运用Py+u-View+Linux-shell组合设计和开发的云端IDE开发软体，本软体的开发与设计采用的是目前使用最流行的浏览器/服务器（B/S）模式，同时使用简单易用，功能全面的Flask技术与Vue框架实现，并通过集成开发环境Pycharm平台提供的远程调试功能以及Mobaxterm与sublime，部署所有的项目，集代码运行、调试、测试于一体的所有开发功能。

下面详细介绍本文的组织结构：

1. 绪论。首先对选题背景与研究意义进行分析，其次对国内外的研究现状与市场产品进行调研，得出本系统的设计有着一定的重要性。

第二章 相关开发技术及开发工具。主要介绍了在线IDE系统设计与开发过程中用到的语言、相关开发技术、相关开发工具以及相应的集成开发环境。

第三章 系统分析。主要从用户需求分析，系统功能分析以及可行性分析三个方面着重剖析，明确了具体设计中需要完成的功能。

第四章 系统设计。根据在线IDE系统整体框架的详细需求分析结果进行总体设计，并进行详细的功能模块设计，运用结构图、流程图来体现各模块的具体；

第五章 系统实现。借助效果图以及文字描述介绍了本系统实现的各个功能。

第六章 系统测试。根据需求分析提出的功能编写合适的测试用例，对在线IDE系统进行黑盒测试。

第七章 总结与展望。本章节对在线IDE系统的开发与设计进行总结，整理在本系统具体开发过程中遇到的各类问题，目前存在的缺陷，以及对以后在线IDE系统的美好展望。

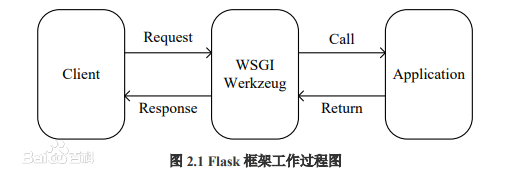
# 2 相关开发技术及工具

## 2.1 Flask

Flask是一个轻量级的python框架，使用Python语言编写，较其他同类型框架更为灵活、轻便、安全且容易上手[5]。

Flask是目前十分流行的web框架，采用Python编程语言来实现相关功能，被称为微框架(microframework)。

Flask运行机制如图1-1所示。



**图1-1 flask运行机制**

## 2.2 Linux shell

Linux 是支持多用户、多任务、支持多线程和多 CPU 的类Unix操作系统。

Shell终端是通过基于系统调用接口开发出的程序，用来让用户与系统进行对话，管理Linux系统。它是一个命令行工具，操作类似于Windows下的CMD工具[6]。

## 2.3 uni-app

uni-app是一款使用 Vue.js 开发跨平台应用的前端框架。uni-app可将Vue.js编译到IOS、Android、微信小程序等多个平台[7]。具有良好的易用性，通过短暂的学习周期就可以开发多平台的应用。

## 2.4 开发环境

### 2.4.1 HbuilderX

HBuilderX是专用于VUE和uni-app开发的开发工具，具有轻巧、极速、vue开发强化、小程序支持等特点[8]。

HX的界面具有强大的语法提示、智能双击等，让文字处理的效率大幅提升，支持现代js开发中大量json结构的写法。

可以方便的调用各种命令行功能，并设置快捷键。

### 2.4.2 Nginx服务器

Nginx 是一款高性能的 HTTP 和反向代理服务器软件，第一个开源版本诞生于 2004 年，虽然诞生较晚但经过十多年的发展，已经成为非常流行的 Web 服务器软件[9]。

Nginx启动后，会产生一个master进程和多个worker进程。master进程主要用来管理worker进程，多个worker进程之间是对等且独立的。一个客户端请求，只可能在一个worker进程中处理，一个worker进程，不可能处理其它进程的请求。worker进程的个数是可以设置的。

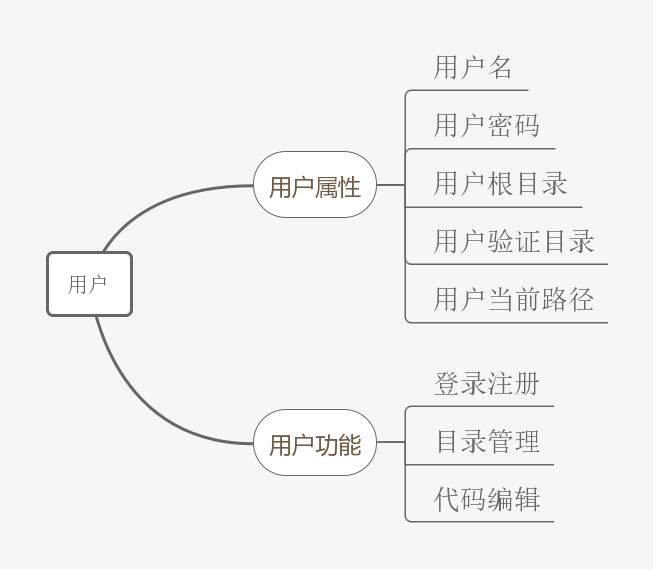
### 2.4.3 Pycharm

PyCharm是一种Python IDE，其带有一整套可以帮助用户在使用Python语言开发时提高其效率的工具[9]。此外，该IDE提供了一些高级功能，以用于支持Django框架下的专业Web开发，同时支持Google App Engine。

# 3 系统分析

## 3.1 用户需求分析

通过对用户的设计需求进行分析，用户的设计主要包含用户属性和用户功能两个方面，用户分析图如图3-1所示。



**图3-1 用户分析图**

（1）用户属性

用户属性是用于实现用户操作所需的所有数据。包括用户名，用户密码，用户根目录，用户验证目录和用户当前路径。

根据用户需求，将用户名作为用户属性的主关键字段，将用户密码作为用户属性的副关键字段。

根据云存储的需求，用户需要根目录，验证目录，当前路径这样三个字段来实现用户目录管理的功能。根目录作为定位和限制用户访问的关键字段，验证目录用于存储用户的关键数据信息，当前路径用于记录用户操作时所在的目录。

（2）用户功能

用户功能是用户可以操作的所有功能。包括登录注册、目录管理、代码编辑。

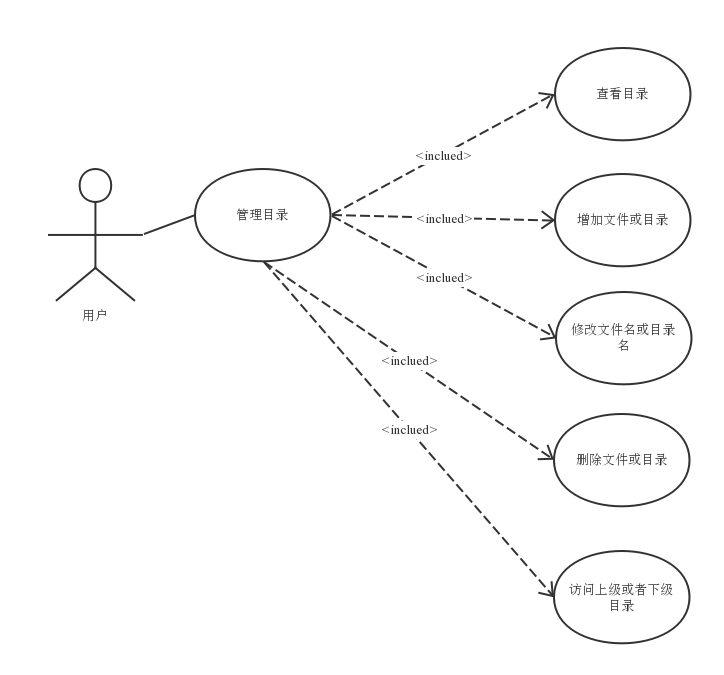
根据用户的基本需求，本系统需要实现登录注册的功能。

根据云存储及代码编辑的需求，本系统需要实现目录管理和代码编辑的功能。

## 3.2 系统功能分析

### 3.2.1云存储功能

本小节主要分析云存储模块实现所需的功能，如图3-2云存储用例图所示。



**图3-2 云存储用例图**

（1）查看目录：满足用户目录可视化的需求，方便用户进行目录的管理。

（2）增加文件或目录：满足用户增加文件和目录的需求，方便用户进一步编写自己的代码。

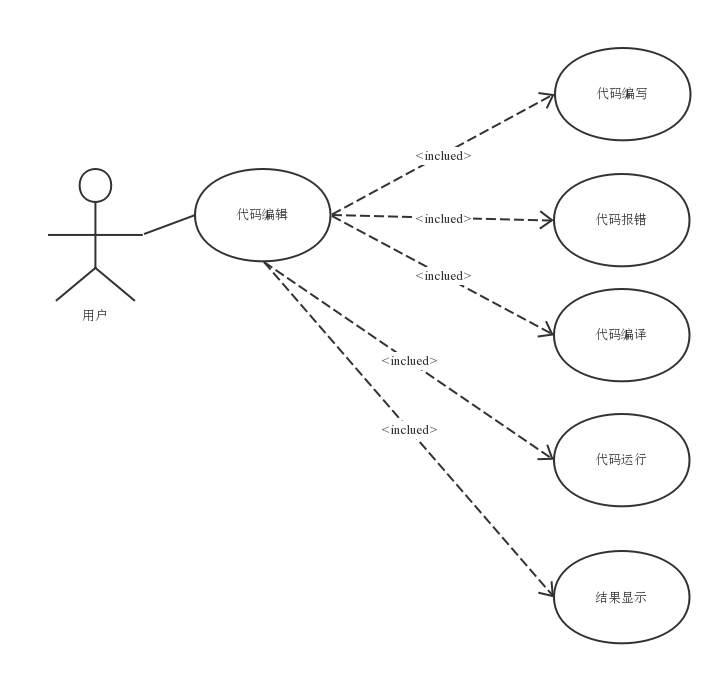
（3）修改文件名或目录：为了满足用户目录的管理，因此需要提供文件名的修改功能。

（4）删除文件或目录：提供用户删除用户空间数据的功能。

（5）访问上下级目录：提供用户可以实现用户空间的多级访问，形成多级目录。

### 3.2.2代码编辑功能

代码编辑功能能够给用户提供良好的代码编写环境、即时的编译报错功能以及程序的运行功能。包括代码编写、代码编译、代码报错、代码运行、结果显示等功能。如图3-3所示。

****

**图3-3代码编辑用例图**

（1）代码编写：满足用户最主要的需求，方便用户进行代码的编写。

（2）代码报错：满足用户debug的需求，方便用户对错误进行修改。

（3）代码编译：满足用户最基本的需求，实现用户代码的编译功能。

（4）代码运行：满足用户运行的需求，将代码运行起来。

（5）结果显示：满足用户最基本的需求，能将结果显示出来，方便用户测试。

## 3.3 可行性分析

在线IDE系统能够保证程序的编译环境不受硬件需求与地理条件的影响，仅需良好的网络通信支持就可以在不同的地点进行代码编写与运行。下面从开发和设计在线IDE系统的技术可行性和经济可行性两个方面进行可行性分析。

### 3.3.1 经济可行性

本系统采用目前最流行的浏览器/服务器（B/S）模式开发，以Linux-CentOS作为云端服务器的系统，使用轻量级的Flask技术+uni-app框架进行前后端开发，并通过开发工具Pycharm平台和Mobaxterm工具部署后端项目到服务器进行运行和调试代码。本系统主要的成本在于服务器的维护。所以，通过以上分析得出本系统的设计与实现都具有一定的经济可行性。

### 3.3.2 技术可行性

在线IDE系统在设计和开发的全部过程中，具备一定的专业技术，在软件开发过程中，本系统需要用到的开发语言与技术包括Python、Typescript、flask开发技术和vue编程语言，开发环境包括Python3.6、G++、Linux-CentOS操作系统等等，开发工具包括Pycharm集成开发平台、Hbuilder以及Mobaxterm工具等等，满足在线IDE系统在整个开发过程中最基本的技术要求，所以本系统在开发过程实际操作中具有一定的技术可行性。

# 4 系统设计

本章对在线IDE系统的开发与设计进行总体概述。在本系统开发与设计的概述阶段中，将根据本系统各个具体模块功能实现的不同，对系统模块的所有功能进行必要的模块划分操作，并针对这些划分后的模块功能进行详细的设计操作。

## 4.1 总体设计

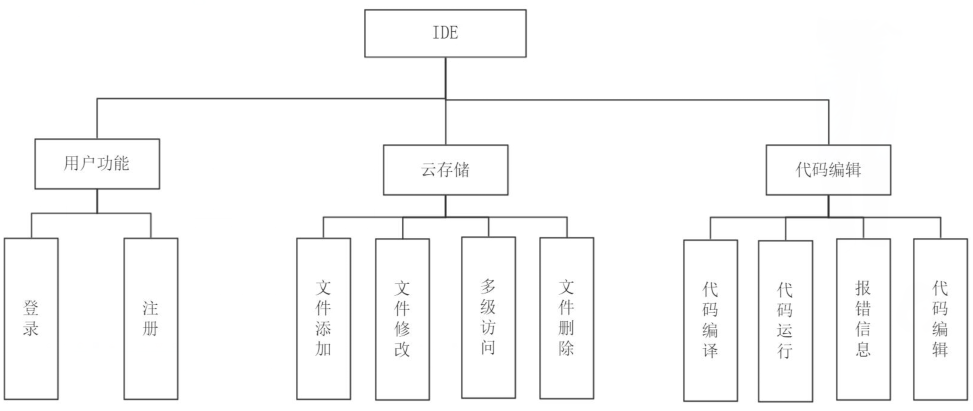
根据上一章在线IDE系统的需求分析，将对本系统进行详细的总体设计。本系统的总体设计主要包含：

（1）类设计

（2）云存储设计

（3）具体功能设计

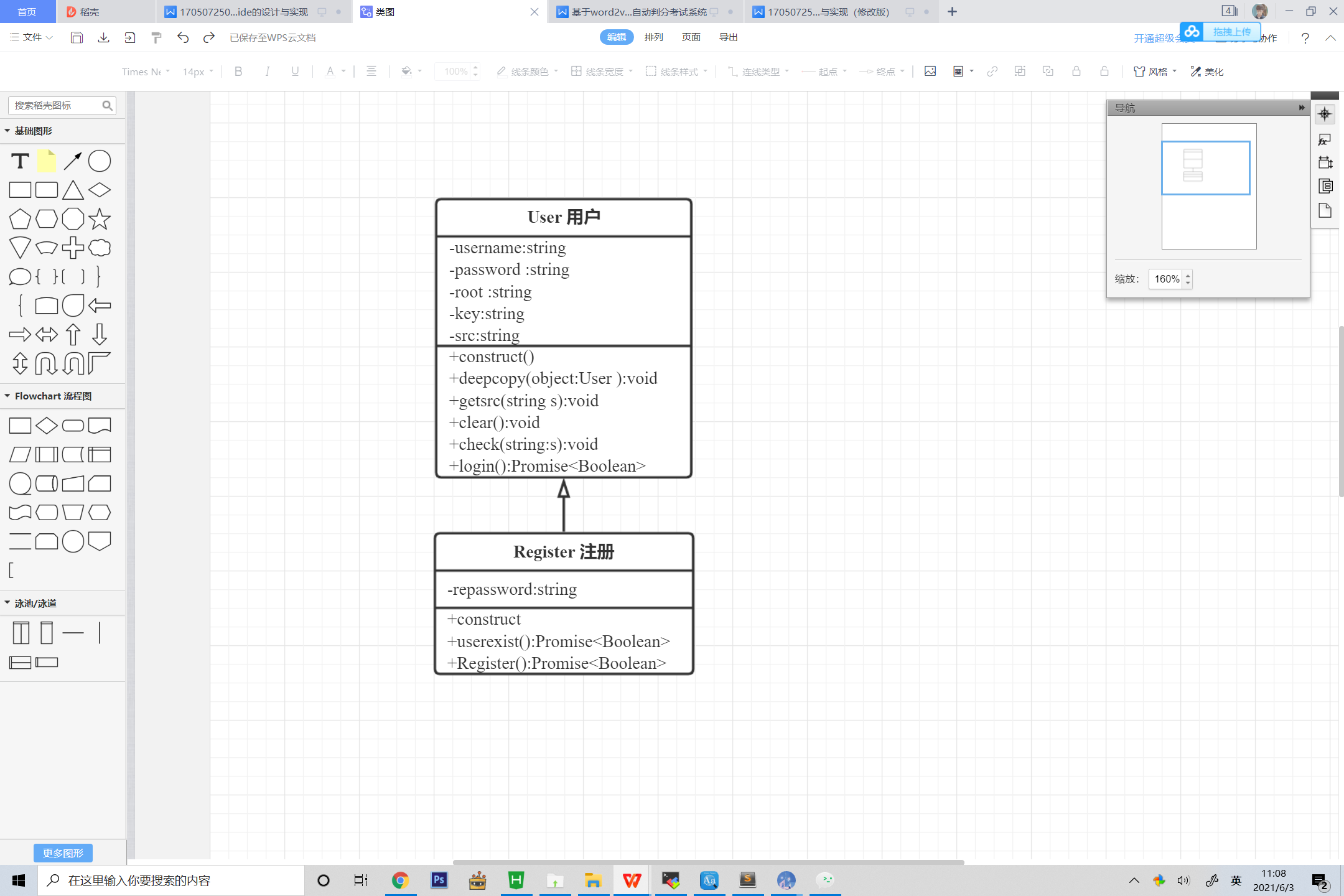
如图4-1总体设计结构图所示。



**图4-1 总体设计结构图**

## 4.2 类设计

类的设计包括User类和它的子类Register的设计，用户类包含登录以及目录管理和后端代码编辑运行所需的关键数据与信息，如用户名、用户密码、用户根目录、用户当前目录、用户验证路径等。考虑到注册需求，本设计使用Register类继承User类的所有属性与方法，并单独封装注册的相关方法。如图4-2所示。



**图4-2 用户****类图**

为了实现系统的功能，设计了User类和子类Login，一个用于存储用户信息，一个用于进行注册。

User类用于保存用户的数据信息，用户登录后，将所有为了实现用户功能所需要的数据都存储在用户类中。

用户字段：用户所需的全部数据存储在类中。

①用户名：用户的账号长度由5~16位字符组成，首字符字母或下划线。

②用户密码：用户的登录密码长度由5~16位字符组成，首字符字母或下划线。

③用户根目录：用户根目录由用户名每四位做一个划分形成的路径组成。用户仅能管理用户根目录中space的文件与目录。

④用户验证目录：用户验证目录存在于用户根目录中，文件名为当前用户的密码。用于登录验证时，验证文件名与用户输入密码是否一致。同时，shell层交互所获取的数据以及程序语言的目标文件都存储在这里。

⑤用户当前路径：用户当前所在的路径，根据当前路径，用户可在当前路径下进行目录的管理。

Register类继承于User类，用于实现用户的注册。

## 4.3 云存储设计

### 4.3.1 云存储目录实现原理

字典树，也叫做Trie树，是基于哈希树的变种，对于复杂的查询来说，字典树就如同字典一样，由于字符串前缀是相同的，因此通过记忆化的方法来合并字符串的前缀，从而省略大量无意义的重复计算，例如“abc”，“abee”，“abdbs”三个字符串，公共前缀子串为“ab”，这里则声明子串“ab”为这三个字符串的祖先。与二叉树的规律相似，字典树每个节点是由一个字符组成的，对于深度为h的一个节点，即是所有前缀为从起点到当前字符的所有字符串第h个字符为当前字符。字典树能简化字符串的查询，提高查询效率[10]。

### 4.3.2 云存储设计

为了解决为每一个用户分配一个唯一地址的问题，本设计通过在用户功能设计时要求用户名字段唯一的方式，使每一个用户都有一个唯一标识。在指定空间建立对应与用户名相同的文件夹，就可以分配一个唯一的地址空间。

考虑到用户数量问题，本设计规定用户名长度为5~15，用户数量约为1EB的数量单位，完全满足用户申请的需求。

对于有多个相同公共前缀的字符串集合，当公共前缀长度达到一定大小时，在集合中查找目标字符串的时间损耗呈几何级增长，时间损耗巨大。本设计采用字典树的思想，假设字符串长度为N，字典树节点最大长度为K，传统的搜索时间约为42N，采用字典树算法，时间效率为42K \* N / K，通过比值运算计算敛散性，得出算式42N-K \* K / N，发现随字典树节点长度K的缩减，时间效率的优化也是呈几何倍数的,本设计中基于平衡性问题的考虑，设定字典树的节点最大长度为4，因此实际应用中，最坏查询仅为424\*15/4<1e(2\*4+1)=1e9，单次查询只需要1s就可以完成原本复杂的查询。

在具体实现中，为避免用户的违法访问，也为了实现用户验证的需求，本设计在用户名在字典树中路径的终点处放置用户目录和用户验证两个文件夹，用于定位用户的目录以及做用户验证这样的操作，字典树节点长度均不长于4个字符，用户名和密码长度均大于5，将两个文件夹设置与用户名密码一致，避免字典树与用户信息的访问冲突性。

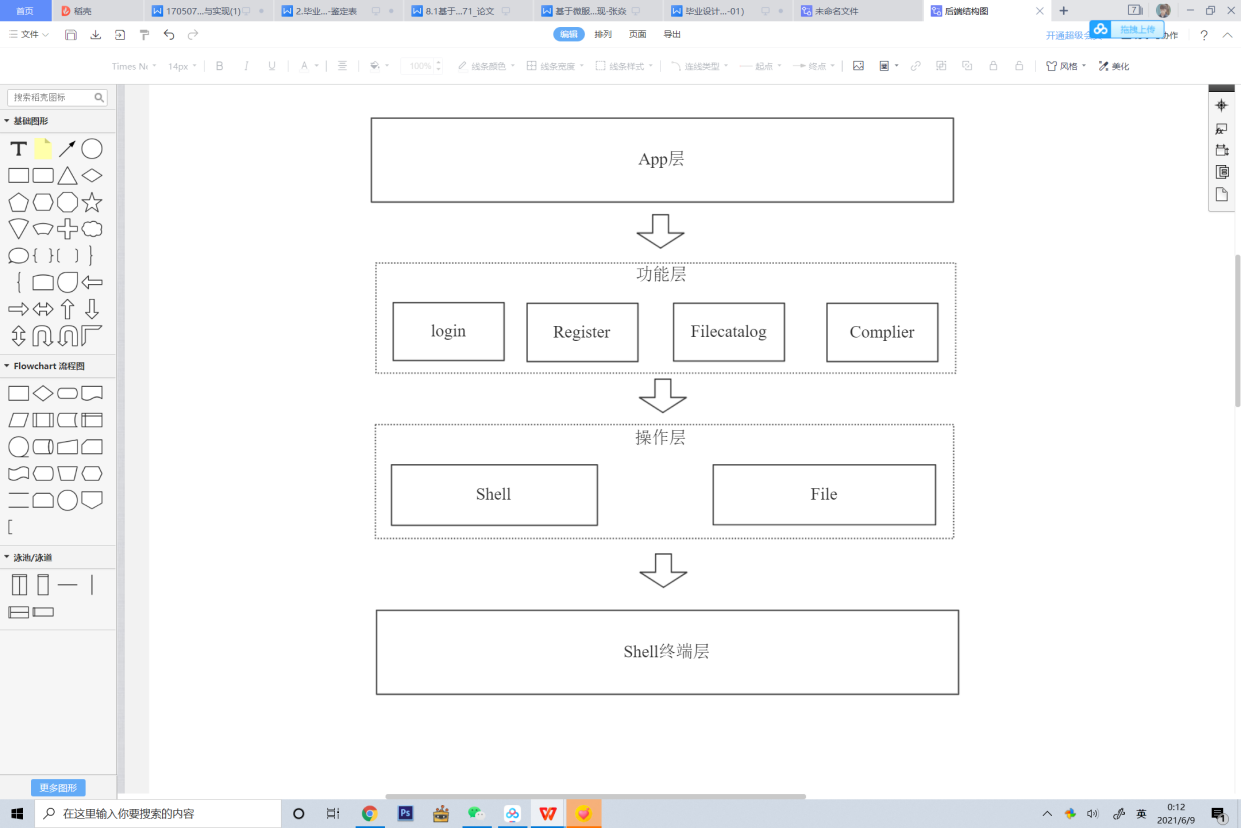
（1）用户空间：在用户路径的终点目录中，声明一个名称为“space”大小为5的文件夹作为用户根目录，当用户注册时检测字典树路径终点space已存在说明用户已经存在则返回注册失败信息。

（2）用户验证目录：在字典树的终点目录处，声明一个名称与用户密码一致的文件夹，用于用户登录时密码的验证，如果输入密码与其不一致则返回密码错误信息。同时，为了提高空间的利用率，本设计将与shell层交互的数据以及程序语言的目标文件存储在验证目录中。

## 4.4 具体功能设计

### 4.4.1 后端设计

后端服务器的设计是主要分为四个层级，App层、功能层、操作层、Shell终端层。如图4-3结构图所示。



**图4-3 后端结构图**

（1）App层：接受前端请求，并根据不同的请求调用功能层的不同功能模块。

（2）功能层：根据不同的功能请求，调用操作层的文件操作或shell操作实现具体操作，如登录、注册、目录管理，编译等，当前层包括四个模块，Login模块，Register模块，Filecatalog模块，Complier模块。

①Login模块：负责实现登录功能，Login模块会调用file层操作来进行用户验证。

②Register模块：负责实现注册功能，Regiser模块会先调用file层验证用户是否存在，如果不存在则，调用相应方法建立对应的文件。

③Filecatalog模块：负责实现目录管理功能，是实现云存储的关键模块，通过调用file层的不同方法实现文件管理。

④Compllier模块：负责实现代码的编译，运行以及报错功能。通过调用shell层的与编译运行相关的方法来实现相关的功能。

1. 操作层：操作层是具体的操作实现层，包括文件操作和文件编译等。该层包含Shell模块和file模块。该层的操作是整个后端的核心功能实现，通过Python的subprocess模块向shell终端发出请求实现对应的操作，对于shell的数据获取通过subprocess与PythonIO文件流结合的办法来实现，subprocess将数据通过IO进行输入输出。

①Shell模块：实现与shell的全部交互以及信息获取

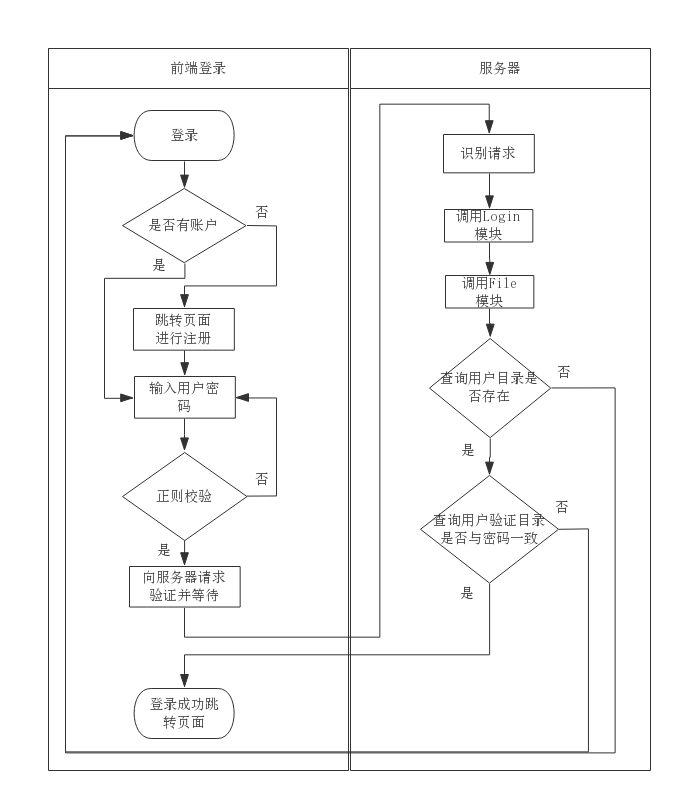
②File模块：通过调用Shell模块中封装好的方法，来封装服务器中的文件操作。

1. Shell终端层：shell终端层主要依托于shell终端和shell脚本来调用相应指令完成对应操作。
2. 后端实现核心思想：后端实现的核心是基于subprocess+PythonIO文件流+shell脚本的组织结构，如图4-3后端结构图所示，在操作层的Shell模块和file模块通过subprocess和文件流将底层的Shell指令封装成简单易用的python接口，方便上层调用。

Shell模块通过subprocess与shell交互调用对应的shell脚本，同时通过将subprocess的输入、输出和报错设置为文件流的形式，将原本复杂的二进制流转换成方便的文件流，简化了交互的难度。

### 4.4.2 登录功能设计

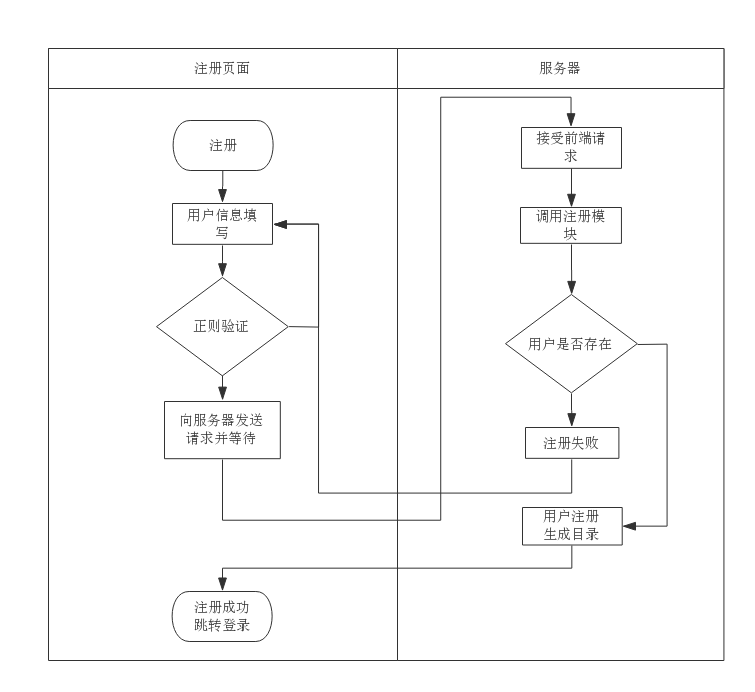
前端提供登录和注册跳转功能，如果用户没有账号则跳转到注册页面进行相应的注册，前端提供了正则校验功能，在注册完成后，用户在登录页面填写好信息，前端会对用户填写的用户名密码进行正则校验，当验证成功后，前端会请求服务器进行验证，服务器接收到信息后，调用Login模块进行登录工作，Login模块使用File模块的方法，检测当前用户名的用户空间是否存在，如果不存在则说明用户也不存在，之后验证用户路径的验证目录是否与用户名一致，最后返回前端对应数据[11]。登录功能的设计如图4-4所示。



**图4-4 登录功能****流程图**

### 4.4.3 注册功能设计

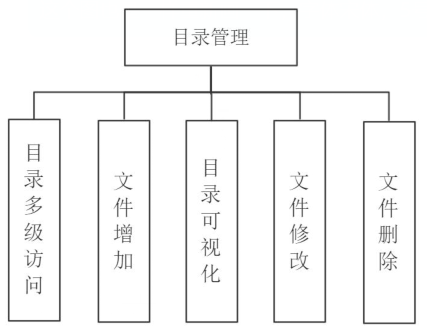
前端通过表单界面，使用户可以进行注册信息填写，填写结束后，表单会自动进行正则验证，不满足页面则弹出提示信息，填写完成后，用户点击注册按钮，前端将数据发送到服务器，请求注册用户，服务器接收前端请求，调用注册模块检测用户是否存在，如果不存在则创建相应目录[12]。注册功能的设计如图4-5所示。



**图4-5 注册功能流程图**

### 4.4.4 目录管理功能设计

目录管理功能主要为用户提供云存储的管理，如图4-6所示，主要包括五个子功能，目录可视化、目录多级访问、文件增加、文件修改以及文件删除。

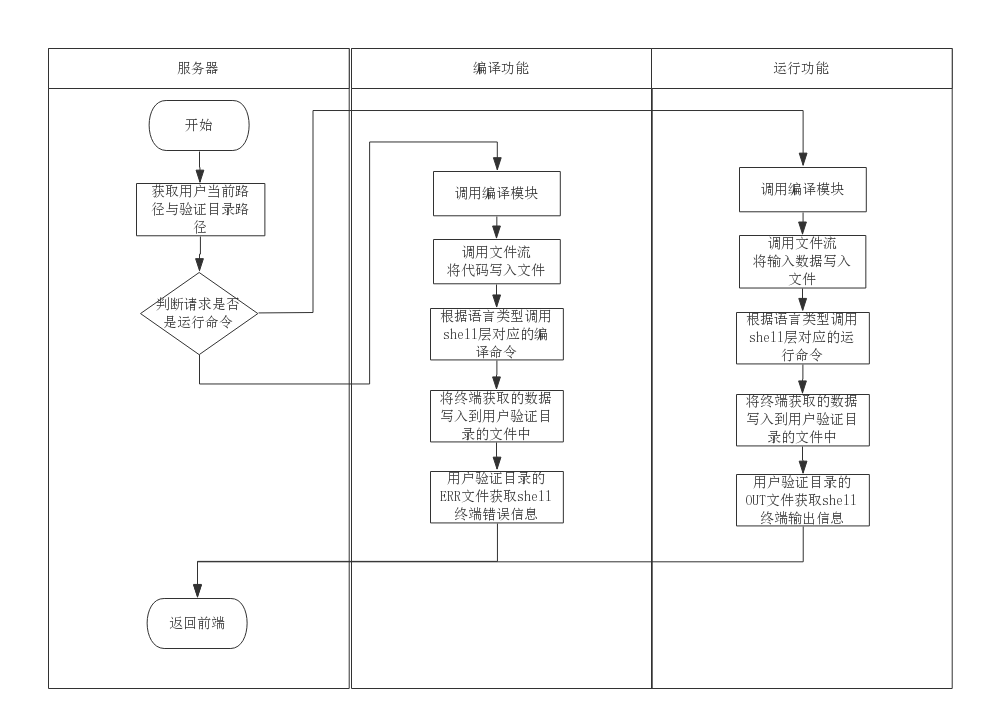


**图4-6 目录管理功能**

1. 目录多级访问：提供用户访问上级目录和下级目录的功能，当用户点击返回上级目录按钮或点击下级目录时，向服务器请求新的目录表，服务器返回请求后，前端进行目录更新。
2. 目录可视化：前端通过请求目录表，获取目录信息，通过滑动控件显示出来。方便用户进行目录管理。
3. 文件增加：当用户在界面内填写完信息后，前端把请求发送到服务器，服务器调用filecatalog模块touch或mkdir添加文件或文件夹。
4. 文件修改：前端将请求发送到服务器，服务器根据请求调用filecatelog模块的rename方法进行文件名的修改。
5. 文件删除：前端将请求发送到服务器，服务器根据请求调用filecatelog模块的delfile方法删除对应文件。

### 4.4.4 代码编辑功能设计

代码编辑功能为用户提供良好的代码编译与运行环境，包括编译功能和运行功能。如图4-7所示。



**图4-7 代码编辑功能流程图**

（1）代码编译功能：用户点击编译按钮，前端将代码传送至后端进行编译。后端根据代码类型调用具体的编译命令进行编译，并将目标文件和错误提示写入到用户验证目录中。后端读取错误文件并将数据返回到前端。

（2）代码运行功能：用户填写好测试数据，点击运行按钮，前端将输入数据传送至后端运行。后端将输入数据写入到用户验证目录的输入文件中，根据代码类型调用具体的运行命令运行用户验证目录中的目标文件，并将输入流重定向到之前写好的输入文件，输出流重定向到用户验证目录的输出文件。后端读取输出并将数据返回到前端。

# 5 系统实现

## 5.1 登录界面

在用户的登录界面，用户输入自己的账号信息，前端会进行正则验证，用户点击登录按钮，客户端向服务器发出请求，并获取相应的信息，如图5-1所示。



**图5-1 登录界面**

## 5.2 注册界面

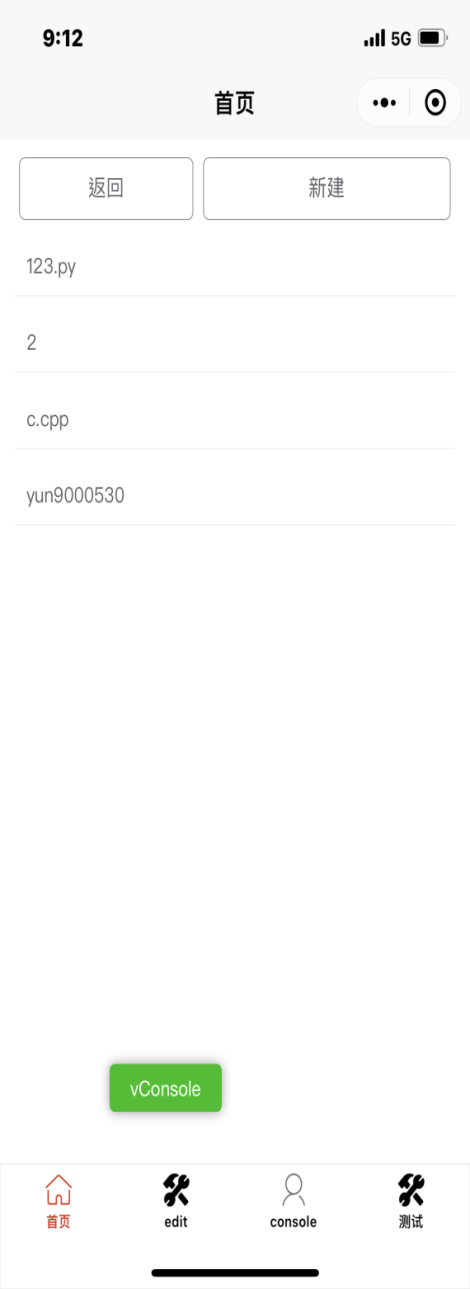
在注册页面中用户进行账号注册，与登录界面一致，前端会进行正则验证，验证成功后，进行服务器验证，前端会显示对应提示。如图5-2所示。



**图5-2 注册界面**

## 5.3 用户目录

用户目录中，用户进行目录管理，包括目录可视化、目录的多级访问、文件的增加、文件的删除、文件的修改。如图5-3所示。



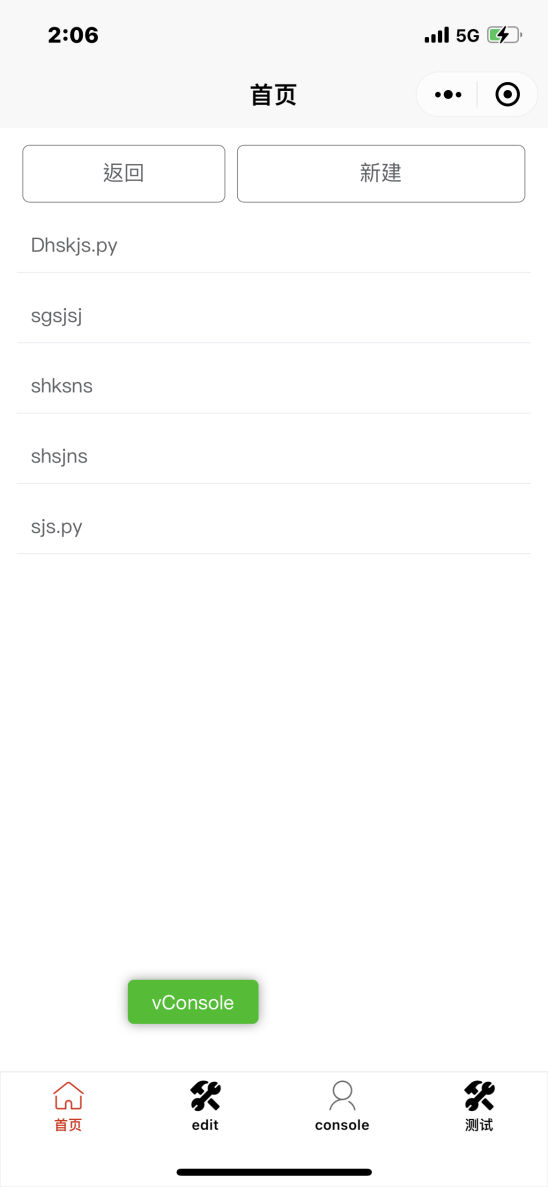
**图5-3 用户目录**

（1）文件管理：在用户目录，用户对文件左滑可以对具体文件进行重命名或删除操作。如图5-4所示。



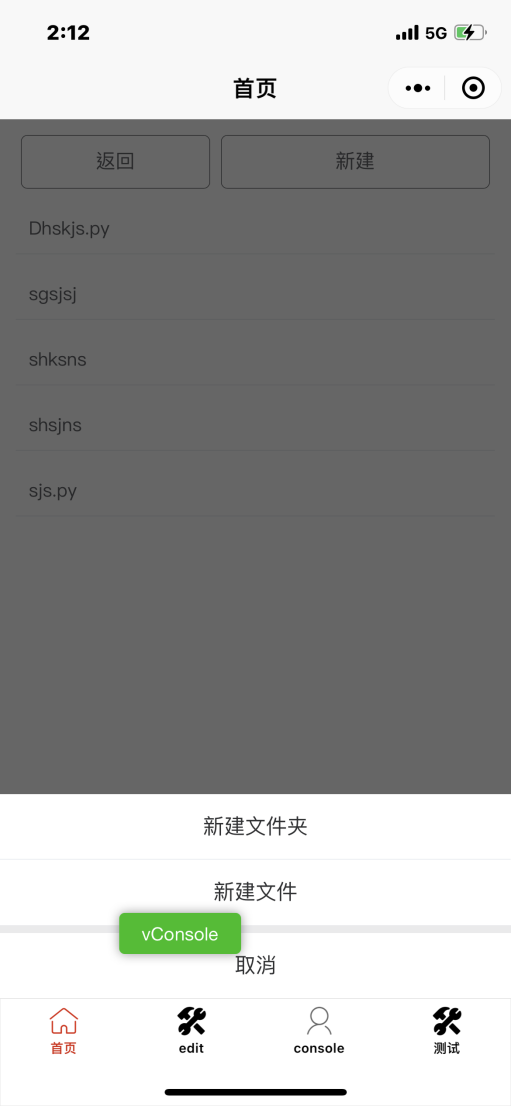
**图5-4 文件管理**

（2）目录多级访问：在目录中，用户点击文件可以访问下级目录。点击返回按钮，可以返回到上一级目录。如图5-5所示。



**图5-5目录多级访问**

（3）文件添加：在用户目录，用户点击新建按钮创建文件，用户可以根据自己需求新建不同语言的文件。如图5-6所示。

****

****

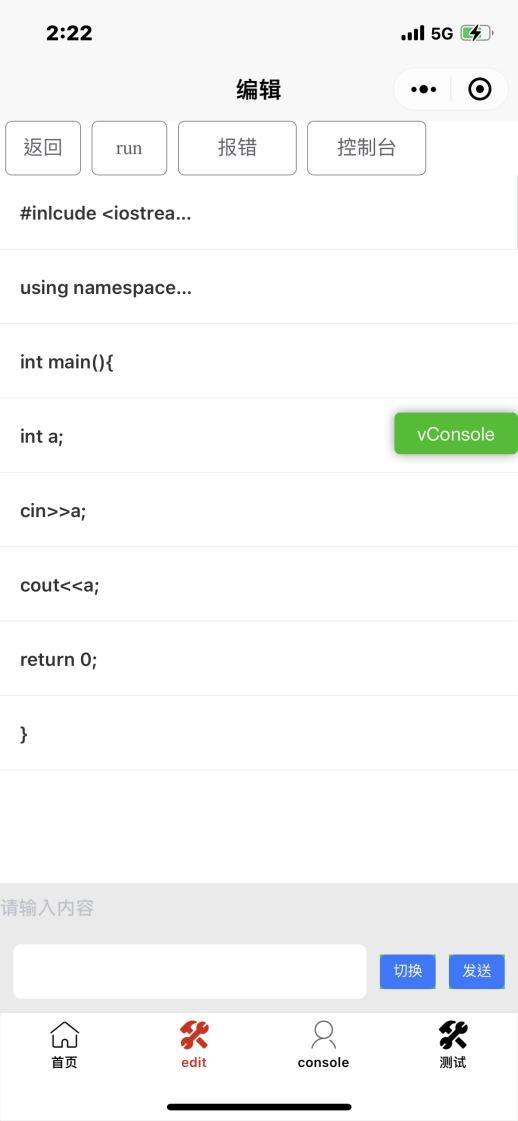
****

**图5-6 文件添加**

## 5.4 编译运行功能

### 5.4.1 代码编辑页面

在代码编辑页面，用户可以进行代码的编写，并可以点击run按钮进行编译，并通过报错和控制台页面进行界面跳转。如图5-7所示。



**图5-7 代码编辑页面**

### 5.4.2 报错页面

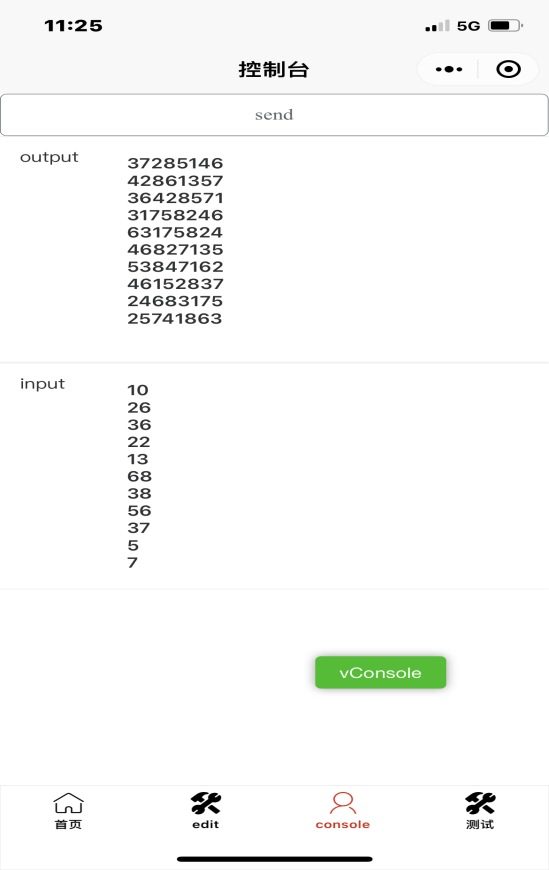
在报错页面，用户可以查看编译错误，并可返回编辑页面进行代码的修改。如图5-8所示。

****

**图5-8报错页面**

### 5.4.3 控制台页面

在控制台页面，可以进行代码的运行，可以让用户输入测试用例，用户点击运行按钮，服务器将输出结果显示在页面上。如图5-9控制台页面所示。



**图5-9控制台页面**

6 系统测试

测试是系统开发必不可少的一环，尽可能的对开发的系统进行全面测试，可以有效防止系统在上线后发生不可预估的错误，本次测试主要针对用户注册登录，目录管理，代码编译运行等几个核心功能进行测试[13]。

## 6.1 测试目的

（1）测试已执行项目是否符合设计要求。具体包括：各功能是否实现，各功能流程是否正确。

（2）为项目提供业务和运行稳定性。

（3）缺陷数量和缺陷率应控制在可接受的范围内。

## 6.2 测试方法

本次系统测试采用了黑盒系统测试的系统测试方式。通过黑盒测试测试业务上的逻辑是否正确，功能是否能够运行成功[14]。

依照在线IDE系统的需求分析，编写IDE的测试用例，确保测试点完全覆盖到本平台的各个模块，并保证每个测试用例在整个测试过程中至少被执行一次。测试用例设计主要采用等价类划分法[15]。

## 6.3 功能测试

### 6.3.1 黑盒测试

用户界面测试内容可以分为以下两类。

（1）界面整体测试。主要评价用户界面的规范化、合理性和一致性。

（2）界面元素测试。主要是对菜单、控件等界面中的元素进行测试。

界面测试中会有很多种检查表，不同的软件企业、不同的软件产品会有不同的检查表内容。表6-1至表6-3给出了几种主要的、较为通用的界面测试检查表，可以在实际工作中予以参考和适当补充。

表6-1窗体界面的测试

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试内容 | | |
| 1 | 窗体大小合适 | 不过于密集或空旷 | |
| 2 | 内部控件布局合理 | 不过于密集或空旷 | |
| 3 | 移动窗体 | 窗体本身刷新正确，背景刷新正确 | |
| 4 | 缩放窗体，窗体上的控件也应该随着窗体而缩放 | | |
| 5 | 不同的显示分辨率下，窗体内容正确 | | |
| 6 | 随操作不同，状态栏的内容能正确的变化 | | |
| 7 | 单击工具栏图标后能正确执行相应操作 | | |
| 8 | 工具栏显示的图标和菜单中的图标一致，能直观的代表要完成的操作 | | |
| 9 | 错误信息的内容 | 内容正确、语义清晰、无错别字 | |
| 10 | 父窗体的中心位置在屏幕对角线焦点附近 | | |
| 11 | 主窗体的中心位置在屏幕对角线焦点附近 | | |
| 12 | 子窗体位置在主窗体的左上角或正中 | | |
| 13 | 多个子窗体弹出时应该依次向右下方偏移，以显示出窗体标题为宜 | | |
| 14 | 重要的、使用较频繁的按钮要放在界面上醒目的位置 | | |
| 15 | 界面长宽接近黄金点比例，不要长宽比例失调 | | |
| 16 | 按钮大小基本接近 | | |
| 17 | 不用太长的名称 | | |
| 18 | 按钮的大小与界面的大小和空间协调 | | |
| 19 | 字体的大小与界面的大小比例协调 | | 通常使用宋体，字号为9-12 |
| 20 | 前景色与背景色搭配合理协调，使用柔和颜色，杜绝刺目的颜色 | | |
| 2 | 界面风格要保持一致 | | 字体、字号、颜色相同等 |

表6-2 菜单的测试

|  |  |
| --- | --- |
| 编号 | 测试内容 |
| 1 | 菜单能正常工作，菜单标题与实际执行内容一致，无错别字 |
| 2 | 快捷键和热键无重复 |
| 3 | 快捷键和热键正常工作，与实际执行内容一致 |
| 4 | 菜单的字体、字号一致，无中英文混合使用 |
| 5 | 菜单和语境相关，对于不同用户或用户执行不同的功能时显示菜单不同 |
| 6 | 与当前进行的操作无关的菜单应该被置为灰色 |
| 7 | 鼠标右键菜单操作，测试内容同以上1-6 |
| 8 | 菜单采用“常用-主要-次要-工具-帮助”的顺序排列，符合Windows风格 |
| 9 | 下拉菜单根据菜单的含义进行分组，并按照一定的规则排列，用横线隔开 |
| 10 | 菜单深度一般要求最多控制在3层以内 |
| 11 | 菜单前的图标大小适合，与字高保持一致 |
| 12 | 主菜单数目合适，应为单排布置 |

**表6-3 控件的测试**

|  |  |
| --- | --- |
| 编号 | 测试内容 |
| 1 | 界面控件风格一致，符合Windows风格 |
| 2 | 控件摆放对齐、间隔一致 |
| 3 | 控件没有重叠区域 |
| 4 | 无错别字、无中英文混合，文字无全角和半角混合使用 |
| 5 | 控件的字体一致，大小适宜 |
| 6 | 控件显示完整，不被裁切，不被重叠 |

注册页面主要对用户名，密码的输入格式验证以及注册请求的测试。如表6-2所示。

表6-2 注册页面测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 用户名称 | 密码 | 测试说明 | 预期结果 | 测试结果 |
| 1 | Jenny | 123456 | 密码首字符字母或下划线 | 注册失败 | 注册失败 |
| 2 | Jenny | i123 | 密码长度不小于5位 | 注册失败 | 注册失败 |
| 3 | Jenny | A1b2c3d4e5f6g7h8 | 密码长度不大于15位 | 注册失败 | 注册失败 |
| 4 | Sone | a12345 | 用户名长度不小于5 | 注册失败 | 注册失败 |
| 5 | A1b2c3d4e5f6g7h8 | A12345 | 用户名长度不大于15位 | 注册失败 | 注册失败 |
| 6 | Sone |  | 密码为空 | 注册失败 | 注册失败 |
| 7 |  | A12345 | 用户名为空 | 注册失败 | 注册失败 |
| 8 | Idetest | idetest | 注册成功 | 注册成功 | 注册成功 |
| 9 | Sdwujk | Ccc123456 | 注册成功 | 注册成功 | 注册成功 |
| 10 | Idetest | idetest | 用户已注册 | 注册失败 | 注册失败 |

用户登录测试时对于用户名、密码格式以及用户密码验证的测试。如表6-3所示。

表6-3 用户登录测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 用户名 | 密码 | 测试说明 | 预期结果 | 测试结果 |
| 1 | admin | 123456 | 密码首字符字母或下划线 | 登录失败 | 登录失败 |
| 2 | 3ystem | t23456 | 用户名首字符字母或下划线 | 登录失败 | 登录失败 |
| 3 | Idetest | T123456 | 密码不正确 | 登录失败 | 登录失败 |
| 4 | Idetest | Idetest | 验证成功 | 登录成功 | 登陆成功 |
| 5 | Jenny | i123 | 密码长度不小于5位 | 注册失败 | 注册失败 |
| 6 | Jenny | A1b2c3d4e5f6g7h8 | 密码长度不大于15位 | 注册失败 | 注册失败 |
| 7 | Sone | a12345 | 用户名长度不小于5 | 注册失败 | 注册失败 |
| 8 | A1b2c3d4e5f6g7h8 | A12345 | 用户名长度不大于15位 | 注册失败 | 注册失败 |
| 9 | Sone |  | 密码为空 | 注册失败 | 注册失败 |
| 10 |  | A12345 | 用户名为空 | 注册失败 | 注册失败 |

目录测试主要对于目录增加，目录修改、目录删除三个功能做功能性测试，如表6-4所示。

表6-4 目录测试用例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 路径 | 文件类型 | 文件名 | 操作类型 | 测试说明 | 预期结果 | 测试结果 |
| 1 | /space/ | 文件夹 | Test | 添加 | 在space根目录下创建文件夹test | 新增成功 | 新增成功 |
| 2 | /space/ | cpp | C1 | 添加 | 在space根目录下创建文件C1.cpp | 新增成功 | 新增成功 |
| 3 | /space/Test/ | java | abc | 添加 | 在space/Test目录下创建文件abc.java | 新增成功 | 新增成功 |
| 4 | /space/Test | 文件夹 | Py | 添加 | 在space/Test下创建文件夹Py | 新增成功 | 新增成功 |
| 5 | /space/ | python | New | 添加 | 在space/Test下创建文件New.py | 新增成功 | 新增成功 |
| 6 | /space/Test | 文件夹 | Py | 修改文件名 | 修改/space/Test/目录下的Py文件名为python | 修改成功 | 修改成功 |
| 7 | /space/ | Cpp | C1 | 修改文件名 | 修改/space/Test目录下的C1.Cpp文件名为app | 修改成功 | 修改成功 |
| 8 | /space/ | Cpp | app | 删除文件 | 删除/space/目录下的app.cpp文件 | 删除成功 | 删除成功 |
| 9 | /space/Test/ | 文件夹 | python | 删除文件夹 | 删除/space/Test目录下的python文件夹 | 删除成功 | 删除成功 |
| 10 | /space/ | 文件夹 | Test | 删除文件夹 | 删除/space/目录下的Test文件夹 | 删除成功 | 删除成功 |

编译运行主要测试能否对特定代码进行正确的编译，以及输入数据能否返回正确数据，如表6-5所示。

表6-5 编译运行测试

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 测试文件 | 文件类型 | 编译结果 | 预期结果 | 运行结果与预期一致性 | 测试说明 | 测试结果 |
| 1 | A.cpp | Cpp | 编译报错 | 编译报错 | 一致 | 语句错误 | 测试成功 |
| 2 | B.cpp | Cpp | 编译成功 | 编译成功 | 一致 | 运行成功 | 测试成功 |
| 3 | C.py | Py | 编译成功 | 编译成功 | 一致 | 运行成功 | 测试成功 |
| 4 | D.py | Py | 编译成功 | 编译成功 | 一致 | 运行成功 | 测试成功 |
| 5 | E.java | Java | 编译成功 | 编译成功 | 一致 | 运行成功 | 运行成功 |

### 6.3.2 测试结果

通过对各模块的测试，发现各模块功能运行无误，基本满足系统测试需求。如表6-7所示。

表6-7 测试结果表

|  |  |
| --- | --- |
| 程序 | 正常启动 |
| 用户功能 | 正常工作 |
| 云存储功能 | 正常 |
| 目录管理功能 | 正常 |
| 编译功能 | 正常 |
| 运行功能 | 正常 |
| 报错功能 | 正常 |

7 总结与展望

## 7.1 总结

本设计围绕在线IDE系统的设计和开发，通过了解在线IDE系统的研究背景，以及在线IDE系统目前在国内外的研究现状，得出在线IDE系统的设计与开发的可行性和必要性。

在登录模块中，本设计采用了字典树的思想，将用户名每四个字符做一个划分，将字典树节点长度限制为0~4，用长度为5的“space”文件夹来标识用户空间，同时使用密码长度为6~13作为验证文件夹名，并将shell输入输出错误都存储在该文件夹中。

在目录模块中，本设计通过 dirname，ls，touch，mkdir，cd等shell命令为提供了云存储空间的增删改的功能。

在编译模块中，本设计通过shell层，诸如javac，python3，g++等linux服务器所提供的编译环境，实现用户代码的编译，并提供完整的报错信息。

在控制台模块中，本设计使用post将前端的输入数据和后端的输出结果进行交互，实现了代码的验证功能。

在开发设计的过程中，由于uni-app采取的是异步的，导致前端无法实时获取后端返回的数据。因此，在实际开发中采取async+await+promise的方法将后端请求封装到promise内部，将异步的请求同步化。

## 7.2 展望

经过测试，在线IDE系统基本上已达到了预期的需求。但是，由于时间原因，系统设计和开发方面也存在不足之处，诸如界面不够完美，系统安全性不高，并且在线IDE系统还未有完全良好的移动端特性。后期，将对该系统进一步设计和完善。

参考文献

1. Salihbegovic A , Eterovic T , Kaljic E , et al. Design of a domain specific language and IDE for Internet of things applications[C]// Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on. IEEE, 2016.35-48.
2. Hudak D E, Ludban N, Krishnamurthy A, et al. A computational science IDE for HPC systems: design and applications[J]. International journal of parallel programming, 2016, 37(1): 91-105.
3. Diep C K, Tran Q N, Tran M T. Online model-driven IDE to design GUIs for cross-platform mobile applications[C]//Proceedings of the Fourth Symposium on Information and Communication Technology. 2016: 294-300.
4. Espana-Boquera S,D Guerrero-Lopez,Hermi Da -Perez A , et al. Analyzing the learning process (in Programming) by using data collected from an online IDE[C]// International Conference on Information Technology Based Higher Education & Training. IEEE, 2017. 224-25.

[5] Grinberg M . Flask web development : developing web applications with Python[J]. O'Reilly Media, Inc.2018.77-89.

[6] Wright C S . Starting to Write Your Own Linux Shellcode[J]. Social Science Electronic Publishing, 2018.44-61.

[7] Qi Zhang,Shulin Yang,Ruoyu Ren. Research on Uni-app Based Cross-platform Digital Textbook System[A]. International Association of Applied Science and Engineering.Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering (CCSE 2020)[C].International Association of Applied Science and Engineering:成都青恒景逸会务服务有限公司,2020:6.

[8] 许溜溜. 基于HBuilder快速开发移动端APP的设计与实现[J]. 电脑知识与技术, 2020, v.16(10):80-81.

[9] Chittibabu P . PyCharm IDE – A Handbook of Python Programming[M]. 2020.

[10] Liu Q, Wang Q . Dictionary with tree structure for matching pursuit video coding[J]. Electronics Letters, 2000, 36(15):1266-1268.

[11] Wu L, Liang G, Kui S, et al. CEclipse: An online IDE for programing in the cloud[C]//2011 IEEE World Congress on Services. IEEE, 2011: 45-52.

[12] 熊俊漉. 基于React的Web前端组件化研究与应用[D]. 重庆邮电大学, 2018.

[13] 邱程,邬小鲁,何聪,范志容,李凯. 基于黑盒测试的车载影音娱乐系统测试用例设计方法[A]. 中国汽车工程学会.2016中国汽车工程学会年会论文集[C].中国汽车工程学会:中国汽车工程学会,2016:4.

[14] Khan M E, Khan F. A comparative study of white box, black box and grey box testing techniques[J]. Int. J. Adv. Comput. Sci. Appl, 2012, 3(6).

[15] Nidhra S. Black Box and White Box Testing Techniques - A Literature Review[J]. International Journal of Embedded Systems & Applications, 2012, 2(2):29-50.

附录A后端代码

应用层代码如下：

app.py

# -\*- coding: utf-8 -\*-

# antuor:dxiaod

from flask import Flask, request

from flask\_cors import \*

app = Flask(\_\_name\_\_)

import file

import complier

import login

import register

import filecatalog

@app.route('/register',methods=['POST'])

@cross\_origin()

def Register():

print("App层接收注册请求")

return register.Register(request.json)

@app.route('/login',methods=['POST'])

@cross\_origin()

def Login():

print("App层接收登录请求")

return login.Login(request.json)

@app.route('/complier',methods=['POST'])

@cross\_origin()

def com():

print("App层接收编译请求")

return complier.Complier(request.json)

@app.route('/getfile',methods=['POST'])

@cross\_origin()

def getfile():

print("App层接收目录管理请求")

return filecatalog.Catalog(request.json)

app.run(host="0.0.0.0",port= "8080",debug=True)

编译层代码如下：

complier.py

import shell

user = False

req = False

def Complier(req):

global user

print("Complier层接收编译请求")

user = req.get("user")

request = req.get("request")

print("语言类型为","language")

if request =="complier":

print("请求类型为编译")

return complier(req.get("filename"),req.get("code"),req.get("language"))

elif request == "run":

print("请求类型为运行")

return run(req.get("filename"),req.get("language"),req.get("in"))

elif request == "getcode":

return getcode(req.get("filename"))

return "1"

def run(filename,language,Int):

global user

shell.init(user)

print("写入输入数据")

File = open(user["key"]+"In","w")

File.write(Int)

File.close()

print("写入成功")

return shell.run(filename,language)

def getcode(filename):

global user

print("读取代码")

File = open(user["src"]+filename,"r")

code = File.readlines()

File.close()

print("读取成功")

return code

def complier(filename,code,language):

global user

shell.init(user)

writecode(filename,code)

return shell.complier(filename,language)

def writecode(filename,code):

global user

print("写入代码")

print(user["src"]+filename)

File = open(user["src"]+filename,"w")

File.write(code)

File.close()

print("读取成功")

文件层代码如下：

File.py

import subprocess

import shell

user = False

def init(User):

global user

user = User

shell.init(user)

def fileexist(src):

global user

print("调用file层filexist判断",src,"是否存在")

shell.init(user)

return shell.exist(src)

def mkdir(src):

global user

shell.init(user)

print("调用File层mkdir创建",src)

shell.mkdir(src)

def touch(src):

shell.init(user)

print("调用File层touch创建",src)

shell.touch(src)

def delfile(src):

global user

shell.init(user)

print("调用File层defile删除",src)

return shell.delfile(src)

def filelist():

global user

shell.init(user)

print("调用File层filelist获取目录表")

return shell.getlist()

#重命名

def rename(src,newname):

global user

shell.init(user)

print("调用File层rename重命名",src,"为",newname)

return shell.rename(src,newname)

#返回

def back():

global user

print("调用File层back返回上一级目录")

if user["root"]+"space/"==user["src"]:

print("已在用户根目录无法返回上级")

return user["src"]

else:

shell.init(user)

return shell.back()

return user["src"]

Filecatalog.py

#import shell

import file

user = False

req = False

def get():

global user

file.init(user)

s = file.filelist()

return s

def add(addname,isfile):

global user

file.init(user)

print(addname,isfile)

src = user["src"]

src+=addname

if isfile==True:

print(src)

file.touch(src)

else:

file.mkdir(src)

return "true"

def delfile(name):

global user

file.init(user)

file.delfile(user["src"]+name)

return "true"

def back():

global user

file.init(user)

return file.back()

def rename(src,name):

global user

file.init(user)

file.rename(src,name)

return "True"

def Catalog(Req):

global user

global req

req = Req

user = Req.get("user")

request = Req.get("request")

print(user["username"],"用户请求",end="")

if request == "get":

print("获取",user["src"],"目录表")

return get()#usersrc

elif request == "add":

print("在",user["src"],"添加文件",req.get("filename"))

return add(req.get("filename"),req.get("isfile"))

elif request == "del":

print("删除",user["src"],"的文件",req.get("filename"))

return delfile(req.get("filename"))

elif request == "back":

print("返回上一级目录")

return back()

elif request == "rename":

print("重命名","文件",req.get("filename"),"为",req.get("newname"))

src = user["src"]+req.get("filename")

return rename(src,user["src"]+req.get("newname"))

登录模块代码如下：

Login.py

import file

def Login(Req):

user = Req.get("user")

#print("login: ",user["password"])

return userexist(user)

def userexist(user):#检测用户是否存在

file.init(user)

If file.fileexist(user["src"])and file.fileexist(user["root"]+user["password"]):

return 'True'

return "False"

注册模块代码如下：

Register.py

import login

import file

user = False

def userexist():#检测用户是否存在 用户空间是space 密码文件时password

global user

print("当前用户路径为",user["src"])

file.init(user)

if file.fileexist(user["src"]) :

return True

return False

def Register(json):

global user

user = json.get("user")

print("Register层开始检测用户是否存在")

if userexist(): #检查用户是否存在

print("用户已存在，注册失败")

return "True"

else:

print("用户不存在，开始注册")

register() #不存在进行注册

print("注册成功")

return "False"

return "True"

def register():

global user

file.mkdir(user["src"])

print("建立用户空间成功")

file.mkdir(user["root"]+user["password"]+"/")

print("建立用户验证目录成功")

file.mkdir(user["root"]+user["password"]+"/Python")

file.mkdir(user["root"]+user["password"]+"/C++")

#file.touch(user["key"]+"In")

Shell模块代码如下：

Shell.py

import subprocess

user = False

def init(User):

global user

user = User

def Shell(commmand,write,time,istimeout):

global user

if write:

In = open(user["key"]+"In","r")

else:

In = open(user["key"]+"In","w")

Out = open(user["key"]+"Out","w")

Err = open(user["key"]+"Err","w")

result = subprocess.Popen(commmand,stdin= In,stdout=Out,stderr=Err,shell = True,encoding="utf-8")

'''

if istimeout:

result.wait(time)

else:

'''

result.wait()

In.close()

Out.close()

Err.close()

#print(get(S),"no")

#shell

def shellin(commmand):

return Shell(commmand,True,0,False)

def shell(commmand):

return Shell(commmand,False,0,False)

#Get

def getmessage(src,Type):

global user

return get(src+Type)

def get(src):

global user

#print(src,"src",type(src))

file = open(src,"r")

s =file.readlines()

s = "".join(s)

file.close()

return s

def getout():

global user

return getmessage(user["key"],"Out")

def getoutList():

a = getout()

print(a)

return a.split("\n")

def geterr():

global user

return getmessage(user["key"],"Err")

#fileop

def back(src):

print("调用shell层back方法获取上一级目录")

shell("bash shell/back.sh "+src)

return getoutList()[0]

def exist(src):

global user

shell("bash shell/exist.sh "+src)

if getoutList()[0]=="True":

print(src,"已存在")

else:

print(src,"不存在")

return getoutList()[0]=="True"

##空间操作

#目录获取

def getlist():

global user

print("调用shell层getlist方法获取",user["src"],"目录表")

shell("bash shell/getlist.sh "+user["src"])

s = getout()

print(s)

return s

#新建文件

def mkdir(src):

shell("bash shell/mkdir.sh "+src)

def touch(src):

shell("bash shell/touch.sh "+src)

#删除文件

def delfile(src):

shell("bash shell/delfile.sh "+src)

#修改文件名

def rename(src,newname):

shell("bash shell/rename.sh "+src+" "+newname)

#返回目录

def back():

global user

shell ("bash shell/back.sh "+user["src"])

return getoutList()[0]+"/"

def complier(filename,language):

global user

src = user["src"]

print("调用shell层complier方法编译",language,"语言")

if language == "cpp":

key = user["key"]+"C++/"

shell("bash shell/cpp.sh "+src+filename+" "+key+"app")

elif language == "py":

key = user["key"]+"Python/"

rename(src+filename,key+"app.py")

shell("python3 "+key+"app.py")

else:

key = user["key"]+"Java/"

delfile(key)

mkdir(key)

shell("bash shell/java.sh "+src+filename+" "+key)

s = geterr()

print("报错信息为",s)

return s

def run(filename,language):

global user

print("调用shell层complier方法运行",language,"语言")

if language == "cpp":

shellin(user["key"]+"C++/app")

elif language == "Java":

shellin("Java "+user["key"]+"Java/"+filename[:-5])

else:

shellin("python3 "+user["key"]+"Python/app.py")

return getout()

附录B shell脚本

Back.sh 获取上级路径

#! /bin/bash

src=$1

dirname $src

Cpp.sh 编译c++

#! /bin/bash

g++ $1 -w -Wall -O3 -o $2

Delfile.sh 删除文件

#! /bin/bash

rm -rf $1

Exist.sh 判断文件是否存在

#! /bin/bash

src=$1

if test -e $src

then

echo True

else

echo False

fi

Getlist.sh 获取当前目录表

#! /bin/bash

ls $1

Java.sh 编译java

#! /bin/bash

javac $1 -g -d $2

Mkdir.sh 创建目录

#! /bin/bash

src=$1

mkdir -p "$src"

Rename.sh 文件重命名

#! /bin/bash

mv $1 $2

Touch.sh 创建文件

#! /bin/bash

touch $1

致谢

春夏秋冬倏忽而过，不知不觉大学生活已经来到了尾声。在整个毕业设计的过程中，非常感谢帮助我的老师，同学还有关心我的朋友们，在我遇到问题的时候，他们给予我最大的帮助，让我克服困难。

毕业设计的选题开始、任务书、开题报告到中期检查以及答辩PPT的制作，张老师每一步都认认真真的给我做指导，提建议。在我设计过程中，徐震学长同我积极讨论问题，帮助我解决服务器方面的配置的难题。同时，在我心情不好打退堂鼓时，半糖给予无私的关怀与鼓励；在系统测试与系统设计方面，我也感谢可新和铸子提出的宝贵意见。

我还要感谢母校给我们提供了宝贵的学习与实践的机会，如果没有学校多方面的教导，也不会让我们养成良好的学习习惯，与独立的动手能力，编写强大的代码，修改bug能力。

最后，我由衷的感谢我的导师张老师以及学校对我的悉心教导。