# 《Python程序设计基础》程序设计作品说明书




题目： Web应用程序

学院： 21计科03

姓名： 芮志远

学号： B20210405129

指导教师： 周景

起止日期：2023.11.10–2023.12.10

# 摘要

随着互联网的蓬勃发展，Web应用程序在提供更便捷的用户体验方面发挥着日益重要的作用。这一潮流不仅使用户能够更轻松地与数据进行互动，而且也为创新的学习和信息管理提供了广阔的可能性。

在这个背景下，我们推出了名为“学习笔记”的Web应用程序，旨在为用户打造一个便捷而强大的学习笔记管理平台。这个项目的核心功能是让用户能够记录他们感兴趣的主题。通过添加日志条目，用户能够系统地整理和追踪自己在学习每个主题时的进展和思考。这不仅有助于提高学习效率，还为用户提供了一个方便的方式来创建、管理和回顾学习材料。

在“学习笔记”中，用户可以注册、登录和登出，实现对个人学习主题和条目的全面管理。包括创建新主题、添加新条目、以及浏览和阅读已存在的条目和主题。这种综合性的用户管理功能旨在让学习过程更加个性化、灵活和高效。

关键词：Web应用程序、Django、模型、映射URL

# 第1章 需求分析

## 1.1 项目概述

本实验所设计的基于Django框架的“学习笔记”管理平台，能够帮助用户有效的管理学习笔记（主题与主题下的条目），直观的看到每个主题下的条目，方便思考与总结。

整个系统的界面应设计得友好美观，便于用户的操作、能够快速上手，还应具有比较完备的功能。主要设计了以下功能：

- 用户登录、登出
- 用户注册
- 主题显示
- 主题内条目查看
- 条目编辑

## 1.2 运行环境

### 1.2.1 软件环境

PyCharm，python3.9，windows 10操作系统。

### 1.2.2 硬件环境

搭载win10系统的PC机。

### 1.2.2 应用环境搭建

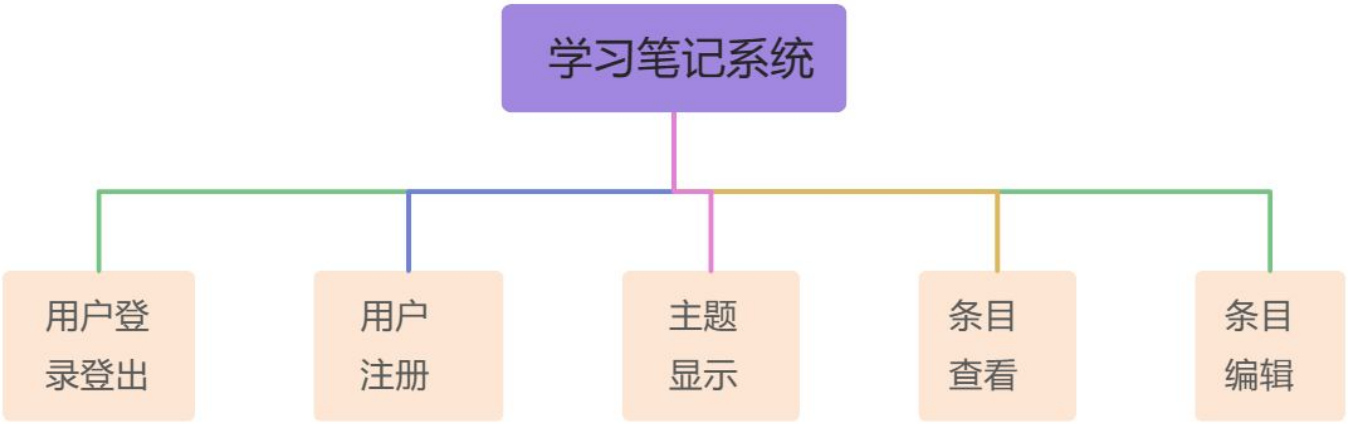
- 建立虚拟环境：python –m venv ll_env
- 激活虚拟环境：ll_env\Scripts\activate
- 建立虚拟环境：python –m venv ll_env
- 安装Django：

```
pip install --upgrade pip
pip install django
```

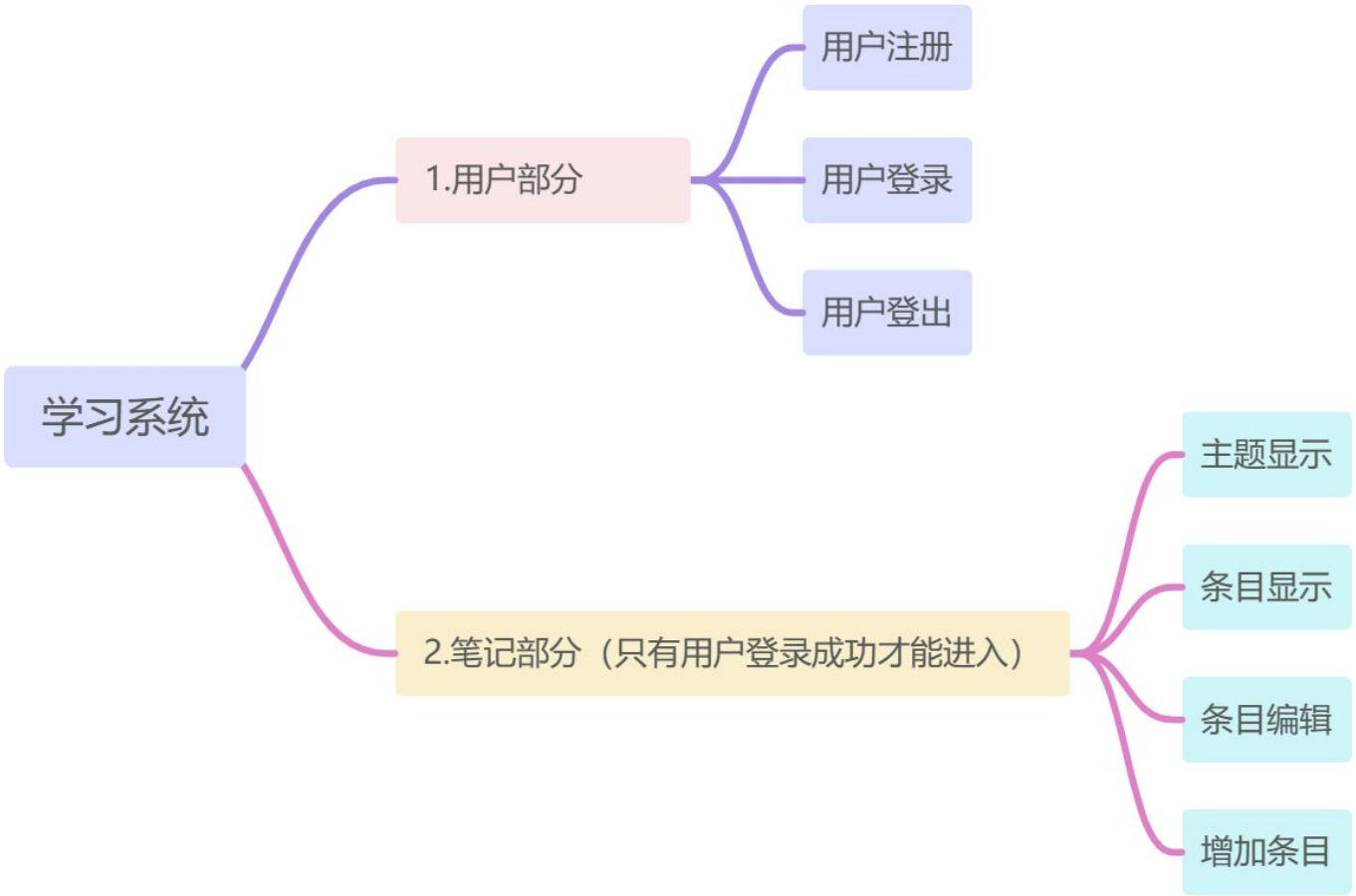
- 在Django中创建项目：django-admin startproject ll_project .
- 在Django中创建项目：django-admin startproject ll_project .
- 创建数据库：python manage.py migrate

# 第2章 分析与设计

## 2.1 系统架构与模块




## 2.2 系统流程




## 2.3 数据库模型

### 2.3.1 主题模型

模型就是一个类，它告诉Django如何处理应用程序中存储的数据。包含属性和方法，Topic类包含text（文本）、date_added（添加日期）、owner（所属用户）三个属性。这里编写__str__()方法，它的返回值为text值。

Topic（主题模型） Python

```python
class Topic(models.Model):
    # """用户学习主题"""
    # 属性1存储少量文本
    text = models.CharField(max_length=200)
    # 属性2记录日期和时间
    date_added = models.DateTimeField(auto_now_add=True)
    owner = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        # """返回模型字符串表示"""
        return self.text
```

### 2.3.2 条目模型

Topic类包含topic（所属主题）、text（条目内容）、date_added（添加日期）三个属性。这里编写__str__()方法，它的返回值为返回一个条目的简单字符串（前50个字符，方便展示）。

Entry（条目模型） Python

```python
class Entry(models.Model):
    # """学到的有关某个主题的具体知识"""
    topic = models.ForeignKey(Topic, on_delete=models.CASCADE)
    text = models.TextField()
    date_added = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name_plural = 'entries'

    def __str__(self):
        # """返回一个表示条目的简单字符串"""
        return f"{self.text[:50]}..."
```

## 2.4 关键的实现

### 2.4.1 映射URL

用户通过在浏览器中输入URL和单机连接来请求网页，因此需要确定项目需要哪些URL，如下：

urls.py Python

```python
urlpatterns = [
    path("admin/", admin.site.urls),
    path("accounts/", include("accounts.urls")),
    path("", include("learning_logs.urls")),
]
```

learning_logs/urls.py Python

```python
app_name = "learning_logs"
urlpatterns = [
    # 主页
    path("", views.index, name="index"),

    # 显示所有主题的页面
    path("topics/", views.topics, name="topics"),

    # 特定主题的详细页面
    path("topics/<int:topic_id>/", views.topic, name="topic"),

    # 用于添加新主题的网页
    path("new_topic/", views.new_topic, name="new_topic"),

    # 用于添加条目
    path("new_entry/<int:topic_id>/", views.new_entry, name="new_entry"),

    # 用于编辑条目的页面
    path("edit_entry/<int:entry_id>/", views.edit_entry, name="edit_entry"
),
]
```

### 2.4.2 编写视图

视图函数接受请求中的信息，准备好生成网页所需的数据，再将这些数据发送给浏览器。

view.py Python

```python
# 为主页编写视图的代码
def index(request):
    # 学习笔记的主页
    return render(request, "learning_logs/index.html")

# 主题视图代码
@login_required
def topics(request):
    # 显示所有主题，显示该用户所有的主题
    # topics = Topic.objects.order_by("date_added")
    topics = Topic.objects.filter(owner=request.user).order_by("date_added")
    context = {'topics':topics}
    return render(request, "learning_logs/topics.html", context)

# 显示主题条目
@login_required
def topic(request,topic_id):
    topic = Topic.objects.get(id=topic_id)

    # 确认请求的主题属于当前用户
    if topic.owner != request.user:
        raise Http404

    entries = topic.entry_set.order_by('-date_added')
    context = {'topic':topic, 'entries':entries}
    return render(request, 'learning_logs/topic.html', context)

# 添加新主题
@login_required
def new_topic(request):
    if request.method != 'POST':
        # 未提交数据，创建一个新的表单
        form = TopicForm()
    else:
        # POST提交数据，对数据进行处理
        form = TopicForm(data=request.POST)
        if form.is_valid():
            new_topic = form.save(commit=False)
            new_topic.owner = request.user
            new_topic.save()
            # form.save()
            # 将用户浏览器重新定向到页面topics
            return redirect('learning_logs:topics')

```

```
    # 显示空表单或指出表单数据无效
    context = {'form':form}
    return render(request, 'learning_logs/new_topic.html', context)

# 添加新条目
@login_required
def new_entry(request, topic_id):
    # 在特定主题中添加新条目
    topic = Topic.objects.get(id=topic_id)
    if request.method != 'POST':
        # 未提交数据，创建一个新的表单
        form = TopicForm()
    else:
        # POST提交数据，对数据进行处理
        form = EntryForm(data=request.POST)
        if form.is_valid():
            new_entry = form.save(commit=False)
            new_entry.topic = topic
            new_entry.save()
            return redirect('learning_logs:topic', topic_id=topic_id)

    # 显示空表单或指出表单数据无效
    context = {'topic': topic, 'form': form}
    return render(request, 'learning_logs/new_entry.html', context)

# 编辑条目
@login_required
def edit_entry(request,entry_id):
    # 编辑既有的条目
    entry = Entry.objects.get(id=entry_id)
    topic = entry.topic
    if topic.owner != request.user:
        raise Http404

    if request.method != 'POST':
        # 初次请求，使用当前条目填充表格
        form = EntryForm(instance=entry)
    else:
        # POST提交数据，对数据进行处理
        form = EntryForm(instance=entry, data=request.POST)
        if form.is_valid():
            form.save()
            return redirect('learning_logs:topic', topic_id=topic.id)

    context = {'entry': entry, 'topic': topic, 'form': form}
    return render(request, 'learning_logs/edit_entry.html', context)
```

### 2.4.3 编写模板

模板定义网页的外观，而每当网页被请求时，Django都将填入相关数据。模板让我们能够访问视图提供的任何数据。

# 第3章 软件测试

本项目利用Django shell来测试项目和排除故障，输入python manage.py shell进入Django shell测试环境。

## 3.1 单元测试用例

查询主题单元测试代码 Python

```
from learning_logs.models import Topic
topics = Topic.objects.all()
for topic in topics:
    print(topic.id, topic)

输出结果:
1 Chess
2 Rock Clibming
```

查询该主题所有属性 Python

```
from learning_logs.models import Topic
t = Topic.objects.get(id=1)
t.text
t.date_added

输出结果:
'Chess'
datetime.datetime(2023, 12, 8, 3, 33, 36, 928795, tzinfo=datetime.timezone.utc)
```

查询该主题下所有条目 Python

```
from learning_logs.models import Topic
t = Topic.objects.get(id=1)
t.entry_set.all()
```

### 3.2 测试用例执行报告

| # | 测试目标 | 输入 | 预期结果 | 测试结果 |
|---|---|---|---|---|
| 1 | 查询主题 | 无 | 查询到所有主题和序号 | 查询到了所有主题和序号 |
| 2 | 查询主题或条目属性 | 主题或条目的id | 查询到相应属性值 | 查询到了text、date_added等属性值 |
| 3 | 获取特定主题下所有条目 | 主题id值 | 查询到该主题下所有条目 | 查询到该id下条目的内容 |

## 结论

1、该项目基础功能都实现了，其中包括用户管理、主题管理、条目管理、简单的界面设计。以提供用户一个便捷而强大的学习笔记管理平台。

- 用户管理： 实现了用户注册、登录和登出功能，确保用户能够轻松地创建和管理他们的学习笔记。
- 主题管理： 提供主题显示功能，用户能够创建新主题，使学习过程更加个性化和灵活。
- 条目管理： 用户可以添加新条目，查看主题内的条目，以及编辑已存在的条目。这有助于系统地整理和追踪在学习每个主题时的进展和思考。
- 界面设计： 设计了友好美观的界面，以提高用户体验，确保用户能够快速上手并轻松地使用平台的各项功能。

2、尽管项目取得了一些显著的成果，但也存在一些不足之处：

- 功能完备性： 需要进一步完善功能，例如可能添加搜索功能、标签管理等，以提供更全面、丰富的学习体验。
- 界面优化： 考虑到用户体验的重要性，界面设计可能需要一些优化，以确保更直观、简洁的操作。
- 安全性： 在用户管理方面，需要确保用户数据的安全性，可能需要进一步加强身份验证和数据保护措施。

# 参考文献

[1] 张乐。 案例教学法在Python语言程序设计教学中的应用[J]. 计算机时代，2021,（04）： 72–75.

[2] 王琦伟。 计算思维在中职计算机专业教学中的培养––以Python教学活动为例[J]. 中国新通信，2021,23（02）： 216–218.

[3] 谢红霞，孟学多。 "Python数据分析基础"线上线下混合教学设计与实施[J]. 计算机时代，2021,（04）： 89–91+94.

[4] 柳青。 Python程序设计课程教与学的线上设计与实践[J]. 电脑知识与技术，2021,17（09）： 15–17.

[5] 王照。 Python语言编程特点及应用[J]. 电脑编程技巧与维护，2021,（03）： 19–20+44.

[6] 夏玲玲，戴文，韩旭，钱怡吉。 基于Python的微信公众号信息采集系统设计与实现[J]. 电子制作，2021,（06）： 58–59+64.