# 实践报告

#### 一、组员及分工

第20组

组长：10224602456高嘉泽

分工：book数据处理、用户接口、订单确认、买家付/退款发货

组员：10222140402吕欣萌

分工：买家接口、插入文本索引、搜索图书

组员：10223330402郭一心

分工：卖家接口、订单状态、收货卖家收款

#### 文档数据库设计：

数据库：bookstore

用户集合（user）

\_id ：标识

user\_id：用户ID

password：用户密码

balance：账户余额

token：缓存令牌

terminal：设备标识

订单集合（order）

\_id：标识

order\_id：订单ID

user\_id：用户ID

store\_id：店铺ID

book\_id\_list：所购图书ID集合

total\_price：订单总价

condition：订单状态

time：下单时间

新订单集合（new\_order）

\_id：标识

order\_id：订单ID

store\_id：店铺ID

user\_id：用户ID

订单详情集合（new\_order\_detail）

\_id：标识

order\_id：订单ID

book\_id：图书ID

count：图书数量

price：图书单价

店铺集合（store）

\_id：标识

store\_id：店铺ID

book\_id：图书ID

book\_info：图书信息

stock\_level：存货情况

商家店铺集合（user\_store）

\_id：标识

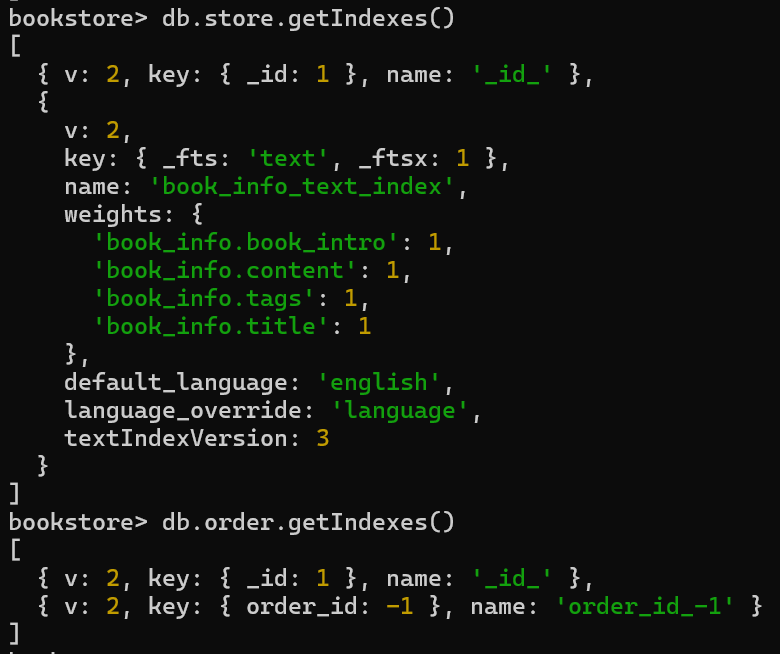
store\_id：店铺ID

user\_id：卖家ID

书籍信息集合（books）

同原有格式

创建索引情况



#### 基础功能展示

###### （1）用户接口

1. 注册register

**接口：**URL = POST "http://127.0.0.1:5000//auth/register/"

**后端逻辑：**实现用户注册功能，接收用户ID和密码作为参数，在数据库中检查用户ID是否存在，如果不存在则插入新用户记录。如果用户ID已存在或出现其他错误，返回相应的错误信息。

**数据库操作：**在数据库bookstore表user中查找是否存在use\_id为用户ID的数据；向数据库bookstore表user中插入新数据（user\_id, password, balance, token, terminal）。

**测试用例：**1.注册用户 2.已存在ID注册用户

1. 注销unregister

**接口：**URL = POST "http://127.0.0.1:5000//auth/unregister/"

**后端逻辑：**实现用户注销功能，接收用户ID和密码作为参数，检查用户ID是否存在，密码是否正确，如果均正确则在数据库中删除此ID的数据，如果用户不存在或密码错误或出现其他错误，返回相应的错误信息。

**数据库操作：**在数据库bookstore表user中查找是否存在user\_id为用户ID的数据；在数据库bookstore表user中删除user\_id为用户ID的数据。

**测试用例：**1.注销用户 2.错误id、错误密码注销用户

1. 登录login

**接口：**URL = POST "http://127.0.0.1:5000//auth/login/"

**后端逻辑：**实现用户登录功能，接收用户ID、密码和terminal作为参数，检查用户ID是否存在，密码是否正确，如果均正确则生成新的token，并在数据库中更新此ID的数据。如果用户不存在或密码错误或出现其他错误，返回相应的错误信息。

**数据库操作：**在数据库bookstore表user中查找是否存在user\_id为用户ID的数据；在数据库bookstore表user中更新user\_id为用户ID的数据（token，terminal）。

**测试用例：**1.登录用户 2.错误ID登录用户 3.错误密码登录用户

1. 修改密码password

**接口：**URL = POST "http://127.0.0.1:5000//auth/password/"

**后端逻辑：**实现用户修改密码功能，接收用户ID、旧密码和新密码作为参数，检查用户ID是否存在，旧密码是否正确，如果均正确则生成新的token和terminal，并在数据库中更新此ID的数据。如果用户不存在或密码错误或出现其他错误，返回相应的错误信息。

**数据库操作：**在数据库bookstore表user中查找是否存在user\_id为用户ID的数据；在数据库bookstore表user中更新user\_id为用户ID的数据（password，token，terminal）。

**测试用例：**1.修改密码 2.错误旧密码修改密码 3.错误ID修改密码

1. 登出logout

**接口：**URL = POST "http://127.0.0.1:5000//auth/logout/"

**后端逻辑：**实现用户登出功能，接收用户ID、token作为参数，检查用户ID是否存在，token是否有效，如果均正确则生成新的terminal，并在数据库中更新此ID的数据。如果用户不存在或token无效或出现其他错误，返回相应的错误信息。

**数据库操作：**在数据库bookstore表user中查找是否存在user\_id为用户ID的数据；在数据库bookstore表user中更新user\_id为用户ID的数据（token，terminal）。

**测试用例：**1.用户登出 2.错误ID登出 3.错误token登出

###### 买家接口

1. 买家下单new\_order

**接口：**URL = POST"http://127.0.0.1:5000//buyer/new\_order/"

**后端逻辑：**实现买家线上下单功能。接收商店 ID 和书籍 ID 与数量的列表作为参数，创建一个新的订单。构建一个包含用户ID、商店ID和书籍列表的详细信息。若失败则新订单建立失败，返回相关信息。

**数据库操作：**根据传入的user\_id和book\_id以及book\_id\_and\_count的参数，将订单信息插入到new\_order表中（用户ID、商店ID、订单状态以及相关信息），对于订单中的每本书更新store表中的库存数量，最后在new\_order\_detail表中记录订单和书籍之间的关系。

**测试用例：**1.包含不存在书籍时订单建立失败 2.库存水平低时订单建立失败 3.正常情况下订单建立成功 4.不存在的用户ID创建订单失败 5.使用不存在店铺ID创建订单失败

1. 付款payment

**接口：**URL = POST "http://127.0.0.1:5000//buyer/payment/"

**后端逻辑：**实现付款功能。接收用户ID、密码和订单ID作为参数，用于验证用户身份和确定要支付的订单。1.使用用户ID和密码进行身份验证，确保用户有权支付该订单。2.根据订单ID在数据库中查找对应的订单，确认订单的存在以及状态。3.在前两点通过的情况下，更新订单状态为已支付，更新库存，以及更新用户的账户余额。

**数据库操作：**根据传入的user\_id和password在user表中进行查询验明用户身份，根据order\_id在new\_order表中查询订单存在性以及可支付状态，支付成功后更新store表。

**测试用例：**1.正常情况下（账户余额充足）支付成功 2.密码错误支付失败 3.账户余额不足情况下支付失败4.重付支付同一订单时支付失败 5.错误用户id支付失败

1. 用户充值add\_funds

**接口：**URL = POST "http://127.0.0.1:5000//buyer/add\_funds/"

**后端逻辑**：实现用户充值功能。接收用户ID、密码和充值的金额作为参数，验证用户身份通过后，将传入的充值金额加到相应用户的账户余额上。若用户授权错误或充值过程出现错误则返回相应信息。

**数据库操作：**根据传入的user\_id和password在user表中进行查询，以验明用户身份。验证通过后，在user表中找到对应用户信息，并更新用户的资金余额balance，将传入的add\_value累加到现有资金余额balance上。

**测试用例：**1.正常情况下传入金额充值成功 2.在错误用户ID下添加资金失败 3.在错误的用户密码下为买家添加资金失败

###### 卖家接口

1. 添加图书 add\_book

**接口：**URL = POST "http://127.0.0.1:5000//seller/add\_book/"

**后端逻辑：**实现卖家对店铺内书籍信息及描述的增加。接收卖家ID，店铺ID，书籍ID，书籍描述和库存为参数。1、验证卖家、店铺、书籍的ID是否存在。2、尝试在数据库中插入需要增加的书籍信息及描述。若添加过程中出现如卖家、商店ID不存在、书籍ID已存在或插入失败等，则返回相应的错误信息。

**数据库操作：**在store表中，插入新添加的书籍信息和对应的商店信息：在商店和书籍ID列更新对应的ID信息，在’book\_info’列更新书籍描述，在’stock\_level’列更新库存数目。

**测试用例：**1.正常情况下添加书籍信息成功。2.店铺不存在时添加失败。3.已添加书籍重复添加失败。4.卖家不存在时添加失败

1. 增加库存 add\_stock\_level

**接口：**URL = POST "http://127.0.0.1:5000//seller/add\_stock\_level/"

**后端逻辑：**实现商家对店铺已有书籍的库存增加。接收卖家ID，店铺ID，书籍ID，新增库存数目为参数。1、验证卖家、店铺、书籍的ID是否存在。2、更新数据库中的库存信息。若更新过程中出现如卖家、商店、书籍的ID不存在或更新失败等，则返回相应的错误信息。

**数据库操作：**在store表中，根据’store\_id’和’book\_id’进行查验，找到对应的库存信息，并加上新增的数目进行更新。

**测试用例：**1.卖家不存在时更新失败。2.商店不存在时更新失败。3.书籍不存在时更新失败。4.正常情况下更新成功

3.创建店铺 create\_store

**接口：**URL = POST "http://127.0.0.1:5000//seller/create\_store/"

**后端逻辑：**实现卖家创建新店铺。接收卖家ID，店铺ID为参数。1、验证卖家、店铺的ID是否存在。2、在数据库中插入新建的店铺。若在创建过程中出现如卖家ID不存在，店铺ID已存在或插入失败等，则返回相应的错误信息。

**数据库操作：**在user\_store表中，插入新建店铺的store\_id和user\_id。

**测试用例：**1.正常情况下，创建成功2.店铺ID已存在时创建失败

#### 附加功能展示

说明：

· 附加功能1-4是基于new\_order实现下单后的功能，payment\_buyer和payment\_seller替代了原有的payment函数功能（并新增了功能）。

· 附加功能5搜索图书是独立的功能。

· 附加功能6查询历史订单为方便实现不同状态下查询订单，将测试函数安排在1-4功能的测试文件中。

1. 订单确认order\_confirm

**接口：**URL = POST "http://127.0.0.1:5000//buyer/order\_confirm/"

**后端逻辑：**实现买家订单确认功能。接收订单ID、时间和意愿作为参数，验证订单是否存在，验证操作时间是否在15分钟内，如果超时，自动取消订单，更新图书库存；在15分钟内，买家可以选择取消订单或确认订单，数据库将更新此订单ID的状态信息，如果取消订单，则会更新图书库存。如果出现其他错误，返回相应错误信息。

**数据库操作：**在数据库bookstore表new\_order中查询订单ID是否存在；根据参数更新order\_id为此订单ID的订单状态condition，超时为overtime-close，取消为cancel，确认为confirm。超时和取消的状态下，更新store中的图书库存stock\_level。

**测试用例：**1.15分钟内确认订单 2.15分钟内取消订单 3.超时确认订单 4.意愿输入字符非法 5.错误订单ID确认订单失败

1. 订单状态order\_condition

**接口：**URL = POST "http://127.0.0.1:5000//buyer/order\_condition/"

**后端逻辑：**实现用户付款后更改订单状况的后续操作。接受订单ID，用户意愿为参数。1、验证用户意愿。2、在数据库中查找对应的订单，并根据用户意愿更新订单状态，如果取消订单，则会更新图书库存。若更新过程中出现如用户意愿输入无效等错误，则返回相应的错误信息。

**数据库操作：**在new\_order表中，根据订单ID查找对应的订单，并根据用户意愿参数，更新订单状态condition，取消订单为cancel，确认收货为receive。取消状态下，更新store中的图书库存stock\_level。

**测试用例：**1.用户意愿为cancel，更新订单状态为取消订单。2、用户意愿为receive，更新订单状态为确认收货3.用户意愿输入无效时更新失败。4.错误订单ID收货失败

1. 买家付/退款、订单发货payment\_buyer

**接口：**URL = POST "http://127.0.0.1:5000//buyer/payment\_buyer/"

**后端逻辑：**实现买家付/退款、订单发货功能。接收用户ID、密码、订单ID作为参数，验证订单是否存在、用户是否存在，密码是否正确、用户当前账户余额是否足够支付。检查订单状态，订单状态为确认订单（confirm）时从买家用户中扣除订单款项，将订单状态改为发货；订单状态为取消订单（cancel）时将钱款返还买家账户。如果出现验证错误或其他错误，返回相应的错误信息。

**数据库操作：**在数据库bookstore表order中查询订单ID是否存在，表user中查询用户ID是否存在，表order中查询order\_id下的订单状态、订单金额；确认订单则更新表user中买家的balance，更新表order中的condition为delivery；取消订单则更新表user中买家的balance，更新表order中的condition为cancel。

**测试用例：**1.确认订单买家付款、订单发货 2.错误用户ID付款失败 3.错误密码付款失败 4.错误订单ID付款失败 5.取消订单退款买家 6.账户余额不足付款失败 7.重复付款失败

1. 订单收货、卖家收款 payment\_seller

**接口：**URL = POST "http://127.0.0.1:5000//buyer/payment\_seller/"

**后端逻辑：**实现用户确认订单后卖家收款。接受卖家ID，订单ID为参数。1、验证订单是否存在。2、根据订单ID查找订单信息，订单金额，记录订单状态。3、订单状态为收货时更新卖家存款。4、卖家收款后，更新订单状态为close，即订单完成。若收款过程中出现如订单、卖家不存在、卖家未收款等，则返回相应的错误信息。

**数据库操作：**在order表中查找对应的订单，并获取订单金额、状态，在user表中查找卖家并在balance更新其存款，订单完成后，将order的condition更新订单状态为close。

**测试用例：**1.订单ID不存在时收款失败。2.卖家不存在时收款失败。3.订单未确认收货时收款失败。4.正常情况下卖家成功收款

1. 插入文本索引搜索图书search\_book

**接口：**URL= GET "http://127.0.0.1:5000//search\_book/search\_books/"

**后端逻辑：**实现图书搜索功能。接收查询关键字、搜索范围（可选，如 tags、title、book\_intro、content、all）、商店 ID（可选）和每页显示条数（可选）作为参数。根据参数进行搜索，返回符合条件的图书列表以及总数和每页显示条数。

**数据库操作：**使用 MongoDB 的文本索引在 store 集合中进行搜索，根据不同的搜索范围和商店 ID 构建查询条件。如果查询成功，将结果转换为 JSON 格式并返回。如果查询失败，记录错误日志并返回空结果、总数为 0 和每页显示条数为默认值（10）。

**测试用例：**1.正常情况下搜索全站图书成功 2.测试输入空字符时，显示结果数量为零 3.测试输入特殊字符时输出结果为零 4.测试索引下搜索标签有结果输出 5.测试索引下搜索目录有结果输出 6.输入罕见关键词时无结果输出 7.传入不同商店ID，测试结果是否正确过滤了商店 8.输入不存在商店ID，显示结果数量为零 9.测试分页功能正常执行 10.测试输入不同每页条数的情况 11.测试响应时间小于1s

1. 查询历史订单search\_order

**接口：**URL = POST "http://127.0.0.1:5000//buyer/search\_order/"

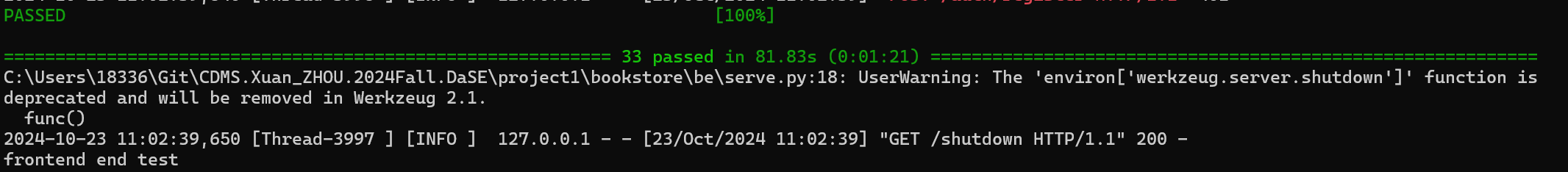
**后端逻辑：**实现用户查询历史订单功能。接受用户id为参数，检查该用户id是否存在，如果存在，则在order表里查询user\_id为该用户id的所有历史订单。

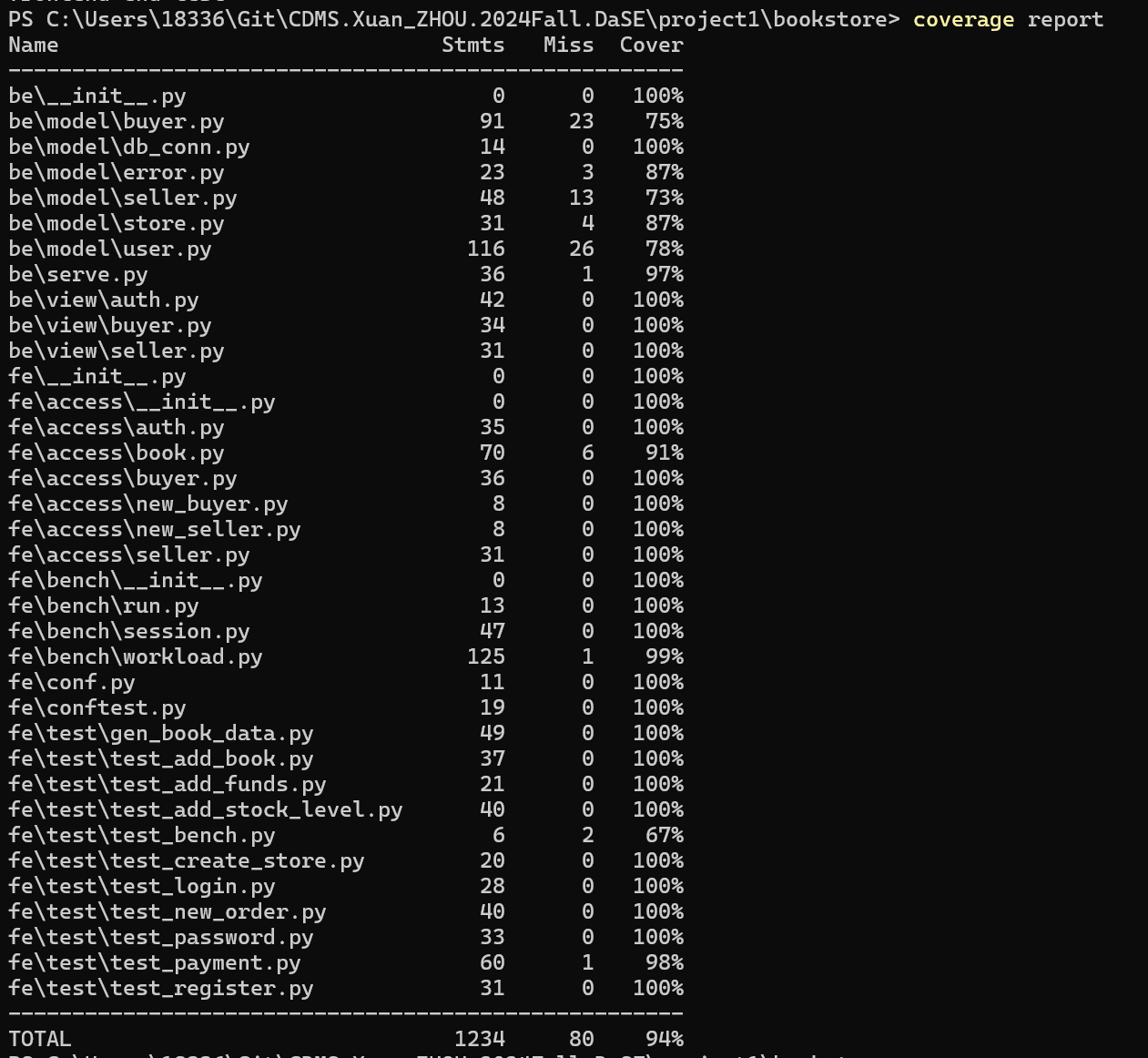
**数据库操作：**在表user中检查user\_id是否存在，在表order中查询用户id为user\_id的所有订单。

**测试用例：**1.新建订单后查询订单 2.确认订单后查询订单 3.付款成功（发货）后查询订单4.收货后查询订单 5.取消订单后查询订单 6.订单完成后查询订单

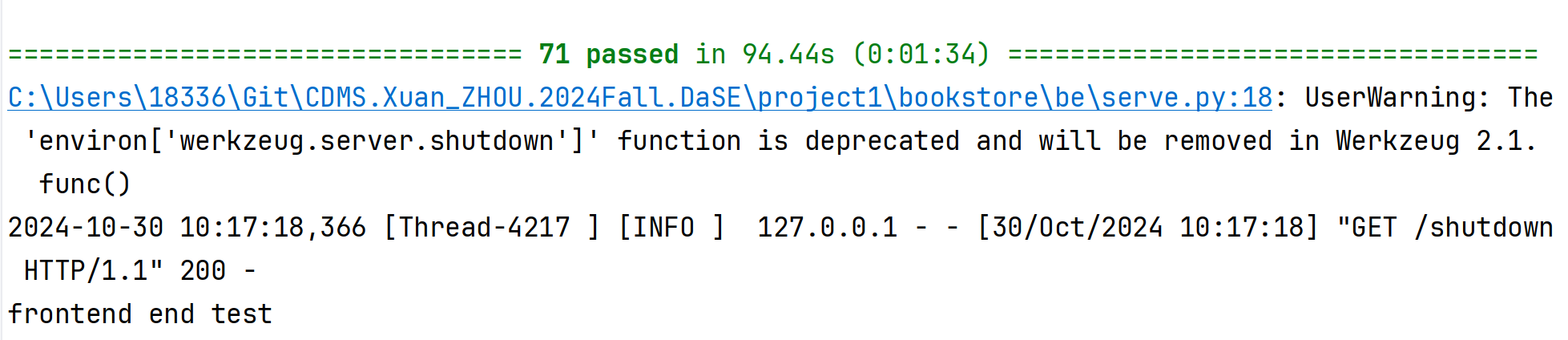
#### 测试结果、测试覆盖率

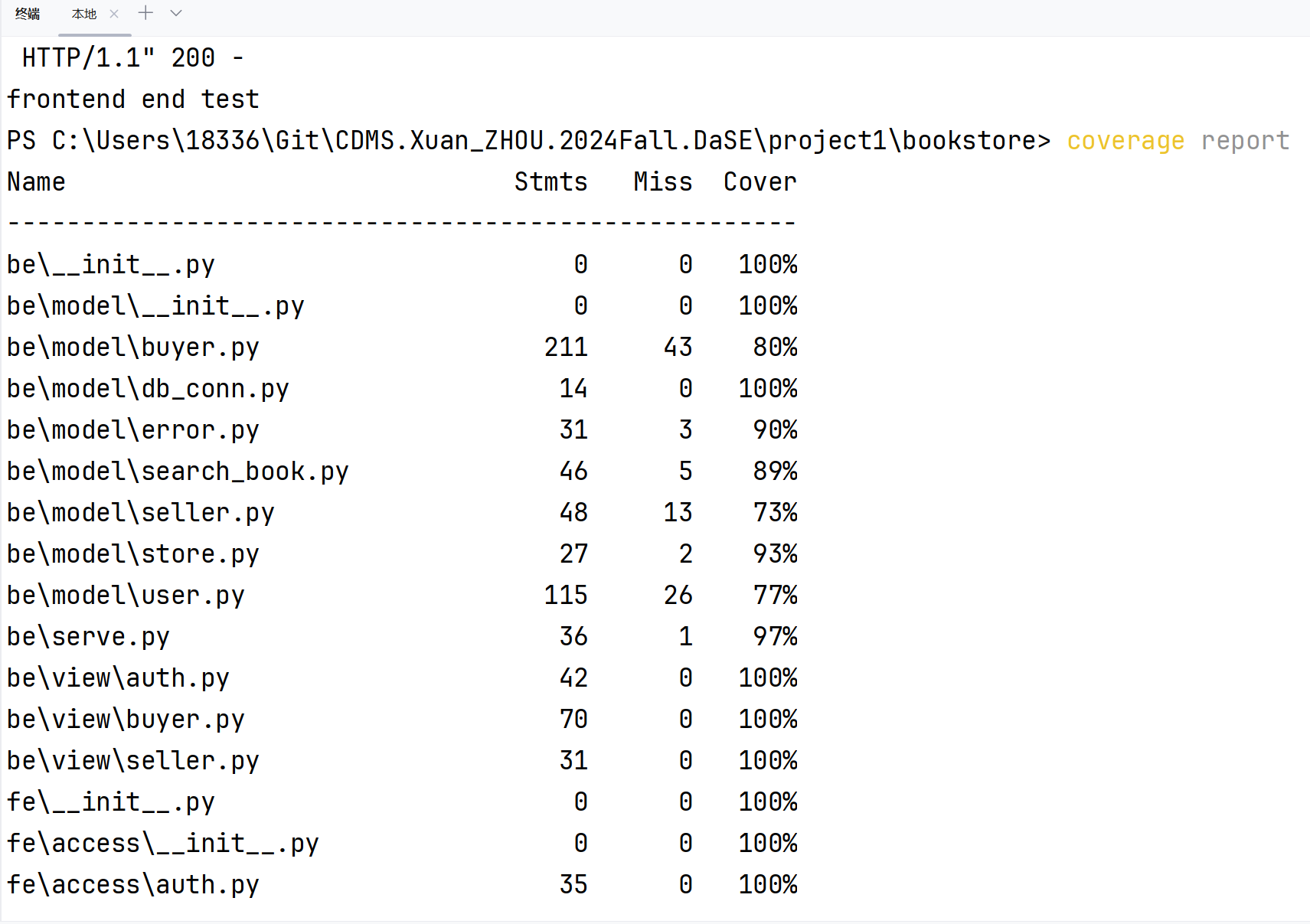
基础功能更新后测试结果

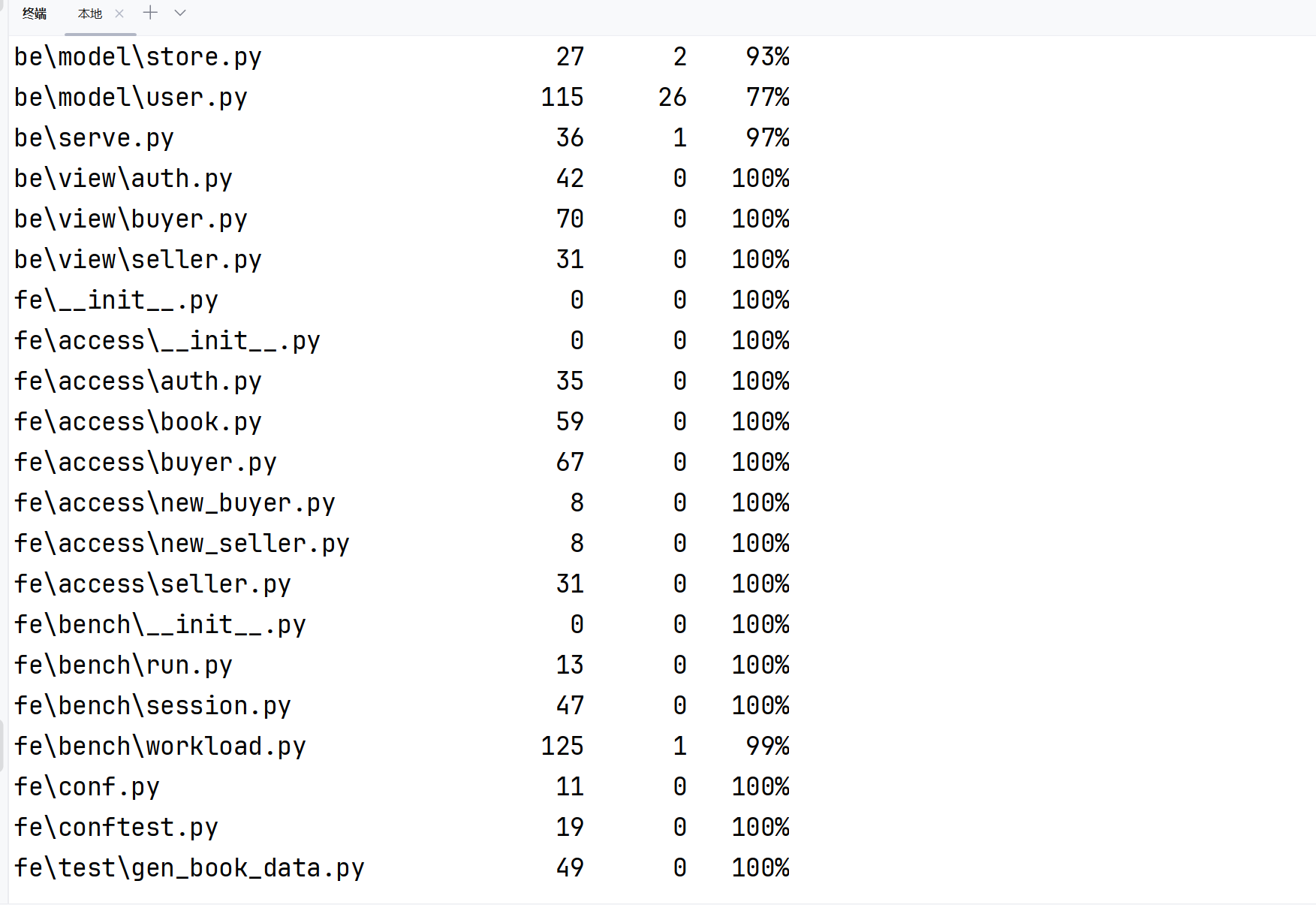


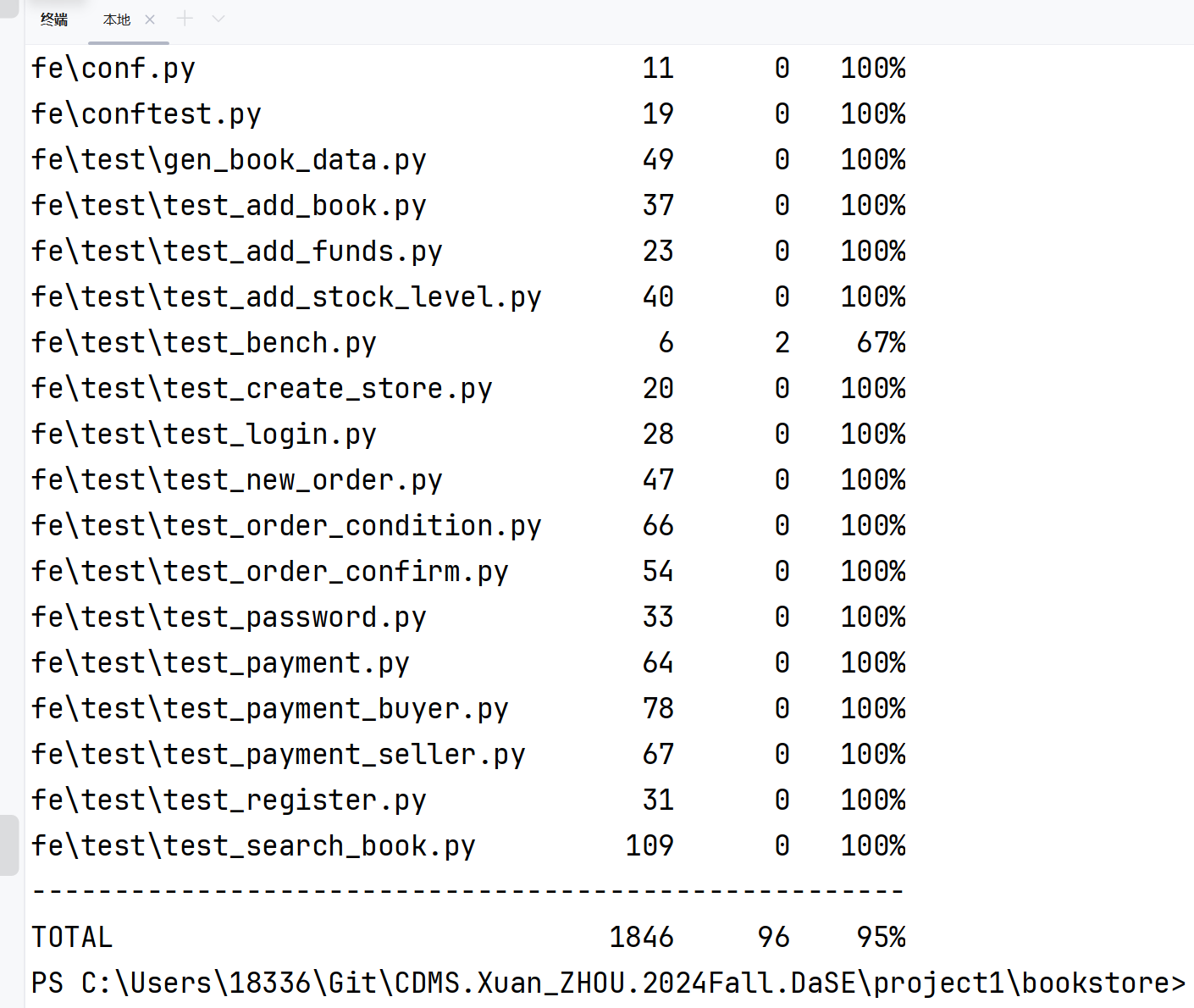


附加功能更新后测试结果









#### git版本管理

