实践报告

一、组员及分工

第 20 组

组长：小泽
分工：book 数据处理、用户接口、订单确认、买家付/退款发货、查询历史订单
组员：小萌
分工：买家接口、插入文本索引、搜索图书
组员：小郭
分工：卖家接口、订单状态、收货卖家收款

二、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

```
bookstore> db.store.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { _fts: 'text', _ftsx: 1 },
    name: 'book_info_text_index',
    weights: {
      'book_info.book_intro': 1,
      'book_info.content': 1,
      'book_info.tags': 1,
      'book_info.title': 1
    },
    default_language: 'english',
    language_override: 'language',
    textIndexVersion: 3
  }
]
bookstore> db.order.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { order_id: -1 }, name: 'order_id_-1' }
]
```

三、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

2. 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

3. 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

4. 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

5. 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

（2）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

2. 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

3. 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

（3）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

2. 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

四、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

2. 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

3. 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

4. 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

5. 插入文本索引搜索图书 search_book

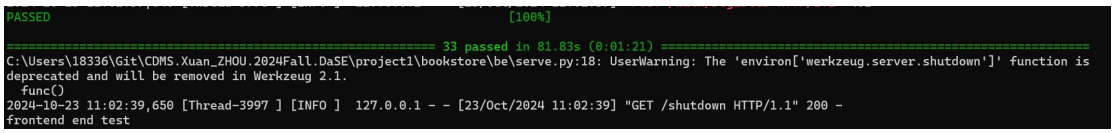
接口：URL= GET "http://127.0.0.1:5000//search_book/search_books/"
后端逻辑：实现图书搜索功能。接收查询关键字、搜索范围（可选，如 tags、title、book_intro、content、all）、商店 ID（可选）和每页显示条数（可选）作为参数。根据参数进行搜索，返回符合条件的图书列表以及总数和每页显示条数。
数据库操作：使用 MongoDB 的文本索引在 store 集合中进行搜索，根据不同的搜索范围和商店 ID 构建查询条件。如果查询成功，将结果转换为 JSON 格式并返回。如果查询失败，记录错误日志并返回空结果、总数为 0 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

6. 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

五、测试结果、测试覆盖率

基础功能更新后测试结果



```
PS C:\Users\18336\Git\CDMS.Xuan_ZHOU.2024Fall.DaSE\project1\bookstore> coverage report
Name                                 Stmts   Miss  Cover
--------------------------------------------------------
be\__init__.py                           0      0   100%
be\model\buyer.py                       91     23    75%
be\model\db_conn.py                     14      0   100%
be\model\error.py                       23      3    87%
be\model\seller.py                      48     13    73%
be\model\store.py                       31      4    87%
be\model\user.py                       116     26    78%
be\serve.py                             36      1    97%
be\view\auth.py                         42      0   100%
be\view\buyer.py                        34      0   100%
be\view\seller.py                       31      0   100%
fe\__init__.py                           0      0   100%
fe\access\__init__.py                    0      0   100%
fe\access\auth.py                       35      0   100%
fe\access\book.py                       70      6    91%
fe\access\buyer.py                      36      0   100%
fe\access\new_buyer.py                   8      0   100%
fe\access\new_seller.py                  8      0   100%
fe\access\seller.py                     31      0   100%
fe\bench\__init__.py                     0      0   100%
fe\bench\run.py                         13      0   100%
fe\bench\session.py                     47      0   100%
fe\bench\workload.py                   125      1    99%
fe\conf.py                              11      0   100%
fe\conftest.py                          19      0   100%
fe\test\gen_book_data.py                49      0   100%
fe\test\test_add_book.py                37      0   100%
fe\test\test_add_funds.py               21      0   100%
fe\test\test_add_stock_level.py         40      0   100%
fe\test\test_bench.py                    6      2    67%
fe\test\test_create_store.py            20      0   100%
fe\test\test_login.py                   28      0   100%
fe\test\test_new_order.py               40      0   100%
fe\test\test_password.py                33      0   100%
fe\test\test_payment.py                 60      1    98%
fe\test\test_register.py                31      0   100%
--------------------------------------------------------
TOTAL                                 1234     80    94%
```

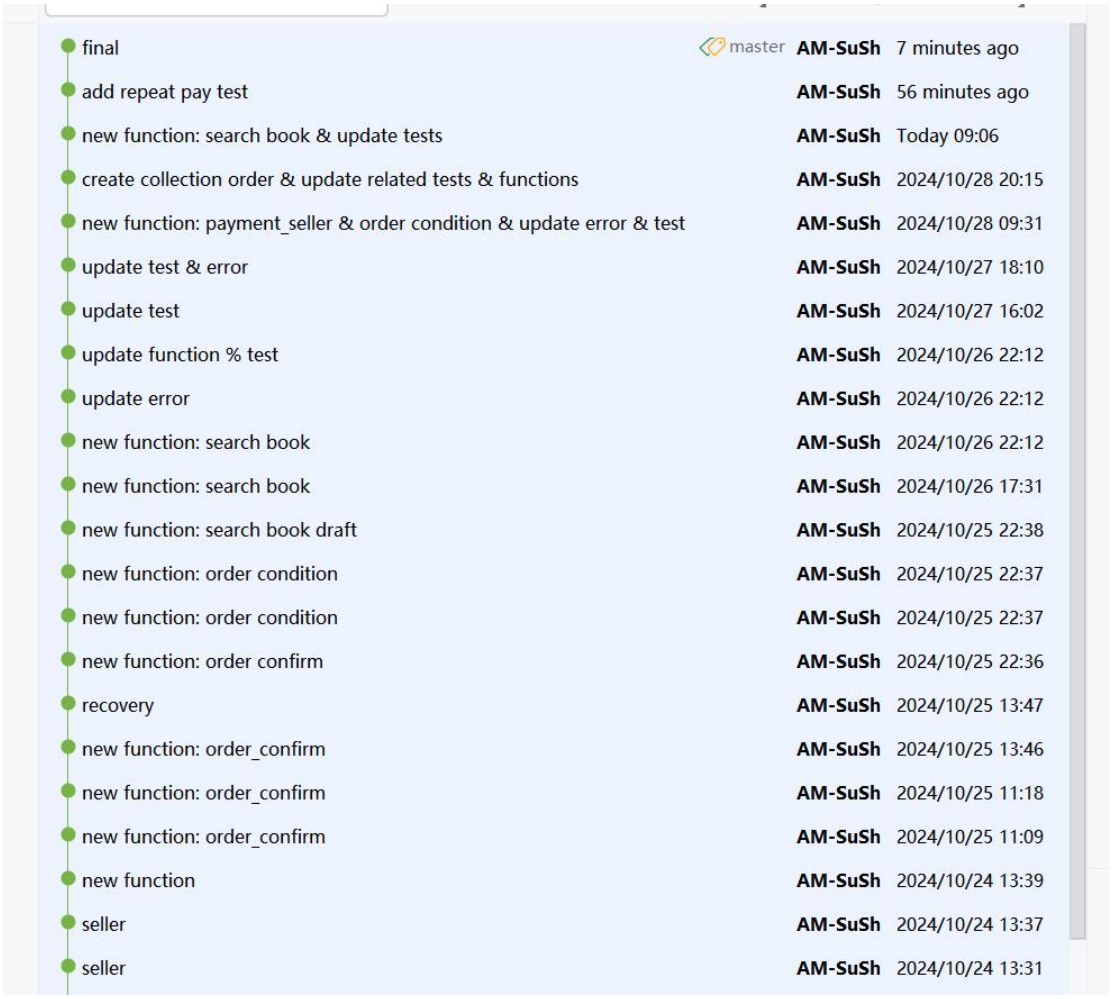
附加功能更新后测试结果

```
fe/test/test_search_book.py::TestBookSearch::test_response_time PASSED                                                [100%]

============================================== 71 passed in 89.37s (0:01:29) ==============================================
C:\Users\18336\Git\CDMS.Xuan_ZHOU.2024Fall.DaSE\project1\bookstore\be\serve.py:18: UserWarning: The 'environ['werkzeug.server.shutdown']' function is
deprecated and will be removed in Werkzeug 2.1.
  func()
2024-10-30 11:05:45,492 [Thread-4277 ] [INFO ]  127.0.0.1 - - [30/Oct/2024 11:05:45] "GET /shutdown HTTP/1.1" 200 -
frontend end test
PS C:\Users\18336\Git\CDMS.Xuan_ZHOU.2024Fall.DaSE\project1\bookstore> coverage report
Name                                 Stmts   Miss  Cover
--------------------------------------------------------
be\__init__.py                           0      0   100%
be\model\__init__.py                     0      0   100%
be\model\buyer.py                      211     43    80%
```

```
be\model\__init__.py                  0      0   100%
be\model\buyer.py                   211     43    80%
be\model\db_conn.py                  14      0   100%
be\model\error.py                    31      3    90%
be\model\search_book.py              46      5    89%
be\model\seller.py                   48     13    73%
be\model\store.py                    27      2    93%
be\model\user.py                    115     26    77%
be\serve.py                          36      1    97%
be\view\auth.py                      42      0   100%
be\view\buyer.py                     70      0   100%
be\view\seller.py                    31      0   100%
fe\__init__.py                        0      0   100%
fe\access\__init__.py                 0      0   100%
fe\access\auth.py                    35      0   100%
fe\access\book.py                    59      0   100%
fe\access\buyer.py                   67      0   100%
fe\access\new_buyer.py                8      0   100%
fe\access\new_seller.py               8      0   100%
fe\access\seller.py                  31      0   100%
fe\bench\__init__.py                  0      0   100%
fe\bench\run.py                      13      0   100%
fe\bench\session.py                  47      0   100%
fe\bench\workload.py                125      1    99%
fe\conf.py                           11      0   100%
fe\conftest.py                       19      0   100%
fe\test\gen_book_data.py             49      0   100%
fe\test\test_add_book.py             37      0   100%
fe\test\test_add_funds.py            23      0   100%
fe\test\test_add_stock_level.py      40      0   100%
fe\test\test_bench.py                 6      2    67%
fe\test\test_create_store.py         20      0   100%
fe\test\test_login.py                28      0   100%
fe\test\test_new_order.py            47      0   100%
fe\test\test_order_condition.py      66      0   100%
fe\test\test_order_confirm.py        54      0   100%
fe\test\test_password.py             33      0   100%
fe\test\test_payment.py              64      0   100%
fe\test\test_payment_buyer.py        78      0   100%
fe\test\test_payment_seller.py       67      0   100%
fe\test\test_register.py             31      0   100%
fe\test\test_search_book.py         109      0   100%
-------------------------------------------------------
TOTAL                              1846     96    95%
PS C:\Users\18336\Git\CDMS.Xuan_ZHOU.2024Fall.DaSE\project1\bookstore>
```

六、git 版本管理


final master AM-SuSh 7 minutes ago
add repeat pay test AM-SuSh 56 minutes ago
new function: search book & update tests AM-SuSh Today 09:06
create collection order & update related tests & functions AM-SuSh 2024/10/28 20:15
new function: payment_seller & order condition & update error & test AM-SuSh 2024/10/28 09:31
update test & error AM-SuSh 2024/10/27 18:10
update test AM-SuSh 2024/10/27 16:02
update function % test AM-SuSh 2024/10/26 22:12
update error AM-SuSh 2024/10/26 22:12
new function: search book AM-SuSh 2024/10/26 22:12
new function: search book AM-SuSh 2024/10/26 17:31
new function: search book draft AM-SuSh 2024/10/25 22:38
new function: order condition AM-SuSh 2024/10/25 22:37
new function: order condition AM-SuSh 2024/10/25 22:37
new function: order confirm AM-SuSh 2024/10/25 22:36
recovery AM-SuSh 2024/10/25 13:47
new function: order_confirm AM-SuSh 2024/10/25 13:46
new function: order_confirm AM-SuSh 2024/10/25 11:18
new function: order_confirm AM-SuSh 2024/10/25 11:09
new function AM-SuSh 2024/10/24 13:39
seller AM-SuSh 2024/10/24 13:37
seller AM-SuSh 2024/10/24 13:31