

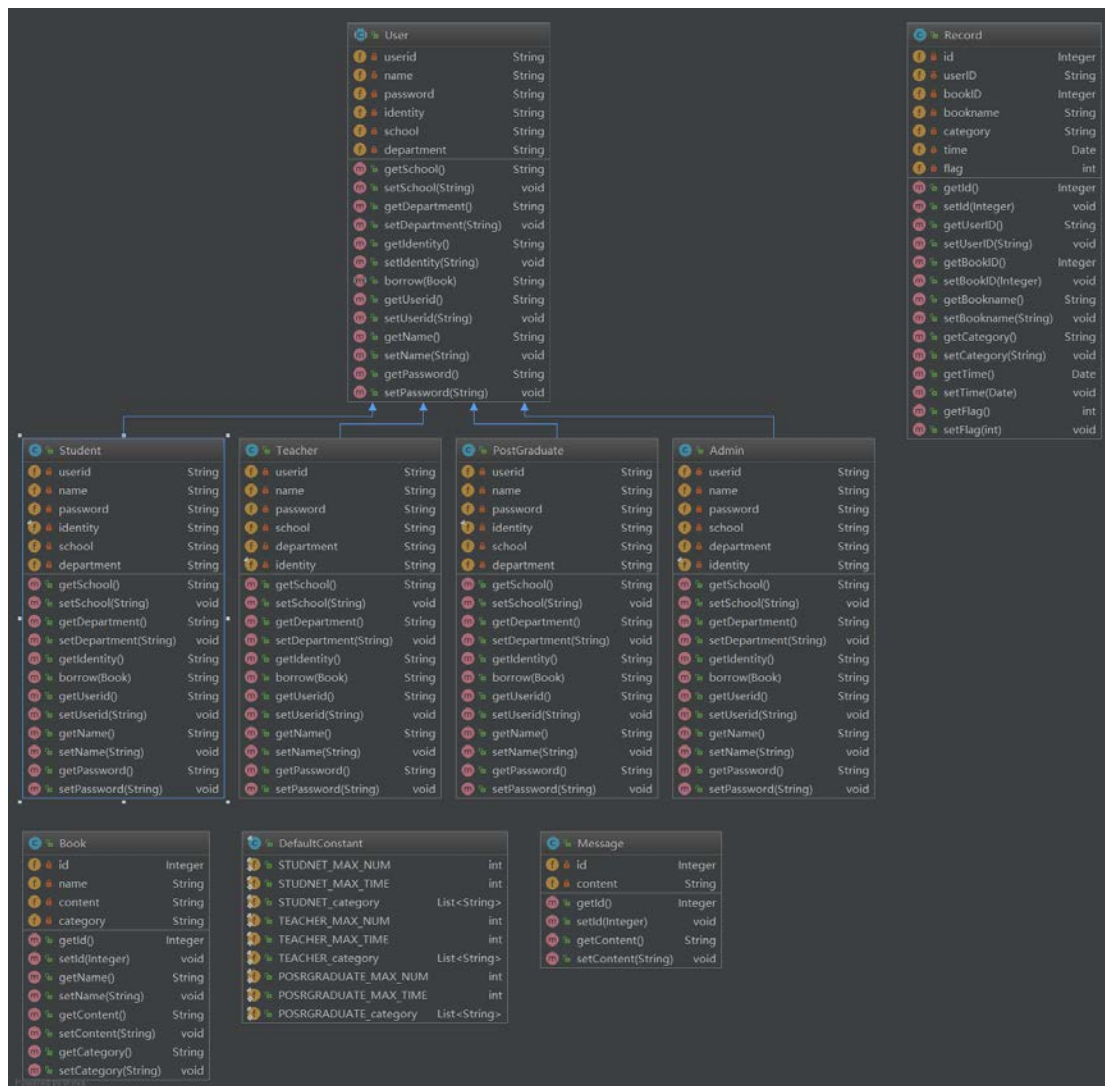
# Software Structured Design & Architecture



























## Assignment 3

161250032 顾诗玉























161250084 陆梅临

# 一.类的设计与说明



		UserController
		userDao UserDao
		messageDao MessageDao
		login(String, String) String
		changeUserInfo(String, int) void
		notifyAdmin(String) void
		getFine(String) double
		insertUser(User) void
		getAllMessage() List<Message>
		isRegister(String) boolean
		getUsers(int) List<User>
		getUserByUserID(String) User
		updateUser(User) void

		BookServiceImpl
		bookDao BookDao
		recordDao RecordDao
		userDao UserDao
		borrow(User, Book) String
		getAllBook() List<Book>
		returnBook(String, int) void
		getAllBook(String) List<Book>
		getRecordByUserID(String) List<Record>
		updateBook(Book) void
		getAllLateRecord(String) List<Record>

总体设计采用了 MVC 模式，实现了前后端分离，依靠 controller 来进行客户端与服务器的交互

组件一在借书的部分采用策略模式，每种不同的用户都实现了借书的方法，在 client 只需要使用 service.borrow(user,book)这样的方式就可

以实现用户借书的扩展性

组件二采用了中介者模式，中介者模式使我们在修改时只需要去关注被修改的对象，而不会对其他造成影响，并且在完成将来设计时，在添加新的身份，给予新的身份管理权限时，可以有效的减小对其他代码的影响，便于拓展

组件三采用了状态模式，根据所选择的书籍，其书的类型的不同，而采用不同的阅读器来进行阅读,只根据单一属性进行状态模式的改变，更利于代码的编写与维护

组件四采用了观察者模式，在用户信息变更的时候，通知管理员

## 二.个人贡献与经历评价

陆梅临：

个人贡献:主要实现后端

顾诗玉：

个人贡献:负责前端以及 controller,并在集成时进行测试，发现并修复了后端在数据库的 4 个 bug 以及在 service 层的 6 个 bug。前端代码总行数:2372,后端代码总行数:2208,其中 controller 层代码 400 行，总共负责代码 2772 行,其中前端采用了 vue 框架,有大量的依赖代码，因此在提交时仅提交 vue 的源代码，具体怎么运行可以看一同提交的程序说明文档。

个人经历:在此次代码编写中，因为在寒假，所以交流采用线上交流方式，感觉效率不是很高，很多设计的细节并不是很到位，并且在集成

时，后端出现了问题，无法及时联系并且修改，只能自己重新阅读后端代码并且理解后端代码在对出现 bug 的地方进行了修复并继续进行集成，导致效率低下以及时间浪费。希望在以后的代码编写时，可以有效改善这些问题。能在设计上花更多的时间，将设计模式使用的更好，更好的设计代码，更有助于对日后程序代码的发展与拓展更新。