一、界面简介
1、add_book_form.ui:
用于添加图书的所有相关信息，被 admin_form.ui 调用。
2、add_reader_form.ui:
用于添加读者的所有相关信息，被 admin_form.ui 调用。
3、admin_form.ui:
用于显示管理员的所有功能，是管理员的主界面。当用户以管理员账号登录图书管理系统时便会弹出这个界面，其中主要分为三个核心的功能组。第一个是图书管理，这里可以实现的功能有图书的简单检索，还有书籍的添加、修改和删除等。第二个是读者管理，可以实现的功能有读者信息的添加、修改和删除等。第三个是读者借阅管理，可以实现的功能有显示当前图书的借阅信息、并且管理员可以同意是否读者进行书籍的借阅、续借和规划等操作。它被 user_login.ui 调用。
4、alter_book_form.ui:
用于修改图书的相关信息，被 book_id_form.ui 调用。
5、alter_reader_form.ui:
用于修改读者的相关信息，被 reader_id_form.ui 调用。
6、book_id_form.ui:
用于在修改图书信息时提供验证，被 admin_form.ui 调用。
7、delete_book_form.ui:
用于删除指定的图书信息，被 admin_form.ui 调用。
8、delete_reader_form.ui:
用于删除指定的读者信息，被 admin_form.ui 调用。
9、delete_ensure_form.ui:
用于在删除数据前进行最后的确认，被 delete_book_form.ui 或 delete_reader_form.ui 调用。
10、reader_form.ui:
用于显示读者的所有功能，是读者操作的主界面。其中主要分为三个核心功能组。第一个是图书检索功能，可以提供比较丰富的检索方式。第二个是图书借阅，主要是选择所要借阅的书籍并向管理员提出借阅图书的申请。第三个是续借归还，这个功能是让读者可以向管理员提出图书续借或者是图书归还的请求的。它被 user_login.ui 调用。
11、reader_id_form.ui:
用于在修改读者信息时提供验证，被 admin_form.ui 调用。
12、user_login.ui:
用于登录图书管理系统，是整个系统运行时的初始界面。可以选择登录到管理员还是读者主界面。
13、user_register.ui:
用于读者在登录系统之前注册自己的相关信息。被 user_login.ui 调用。

二、实现功能
1、用户登录功能：
读者或者管理员可以通过预先设定的账号和密码来登录到管理系统中。对于读者来说，可以先通过注册界面进行信息的注册，之后再执行登录操作。程序中设定可以允许读者和管理员同时登录到图书系统中并保持在线状态，但只局限于一个读者和一个管理员。
2、管理员功能：
登录到管理员主页面之后可以看到有三个核心功能，分别是图书管理、读者管理和借阅管理。图书管理主要是完成对图书相关信息的添加、修改、删除等操作，界面中间的视图可以显示当前数据库中有关图书的相关信息。读者管理的功能和图书管理相类似，可以实现管理员对读者相关信息的增加、修改、删除等功能。最后的借阅管理实现的功能是当读者对书进行借阅、续借或归还操作时，管理员可以对读者的相关请求做出应答。
3、读者功能：
登录到读者主界面之后可以同样看到有三个核心功能，其分别是图书检索、图书借阅和续借归还。图书检索可以说是读者最重要的功能了，对于此，我们实现了四种检索方式：按书名、按出版社、按作者、按书号。除了按书号检索只能定位到唯一的一个结果外，其他三种检索方式可以多个一起使用来查找所需的图书。第二个核心功能是图书的借阅。读者可以直接点中界面左下图书信息表格中的任何一本书，然后按右边的添加按钮即可完成对借阅图书的添加，当然读者也可以在上面的输入框中输入图书号和数量来完成同样的工作。在完成所有图书的选择之后，只要输入读者的账号和相应的密码并按下完成按钮，便完成了借阅信息的最终添加。第三个核心功能是读者对所借图书的续借和归还操作。读者先提出续借或归还申请，之后需要管理员对申请进行决策。

三、程序原理：

1、我们此次图书管理系统用的是开源的 MySQL 数据库。我们选择使用 MySQL 的主要原因是它是开源的，有很多官方或第三方程序的 API 接口，很方便进行调用。当然最重要的是，对于一名开源运动的爱好者来说，Linux + MySQL 的组合方式绝对是最佳的。切入正题，我们的数据库中一共建有五个表，分别是管理员表、读者表、图书表、图书类型表和图书借阅表。这五个表之间存在着外键约束关系，而且我们在图书表上建立了视图，这样可以让读者在进行图书检索时能够对所需的结果有更清晰更直观的感受。对于触发器，我们定义了几个关于添加数据时对不满足约束条件的处理办法，以上有关数据库更加详细的设计请看/doc 目录下的数据库实验报告。
2、对于界面，我们选用的是 Qt。原因首先是 Qt 自带的 QDatabase 库提供了面向 MySQL 连接的接口，其次用 Qt Creator 进行界面编程确实比较简单方便，最后的原因我认为是跨平台性。相较于微软来说，Qt 可以提供在各个平台上接近统一的图形界面效果，而不需对代码做太大的改动，这一点的确非常强大。言归正传，对于熟悉 Qt 界面编程的人来说，其实现的核心机制是信号和槽，即将按钮与槽函数连接在一起。当按钮被按下时，发射信号，触发相应的槽函数完成特定的功能。
3、对于界面和数据库的连接和操作，我们用的是 Qt 中的 Sql 类。这个在《Linux 环境下 Qt4 图形界面与 MySQL 编程》书中有比较详细的用法讲解。其中主要用到的有 QSqlQuery 类，它提供了一种执行和操纵 SQL 语句的方式。执行 SQL 查询、插入、更新、删除数据等。除了 QSqlQuery 类之外，Qt 还提供了封装机制来避免直接使用 SQL 语句，其为我们提供了更加简单的数据库操作和数据显示模型。它们分别是只读的 QSqlQueryModel，操作单表的 QSqlTableModel 和可以支持外键的 QSqlRelationTableModel 这三种类。QSqlQueryModel 类为 SQL 结果集提供一个只读的数据类型。QSqlTableModel 类为数据库提供可读写 SQL 表的可编辑数据模型。它完全脱离了 SQL 语句，这个模型提供了缓冲区，可以先将修改保存起来，当我们提交函数时，再去真正地修改数据库。当然，这个模型比起前面的更高级。
QSqlRelationTableModel 类在 QSqlTableModel 的基础之上为单张表提供了一个可以编辑的数据模型，重要的是其添加了对数据库中外键的支持。