

Drop booX

Description

Our team

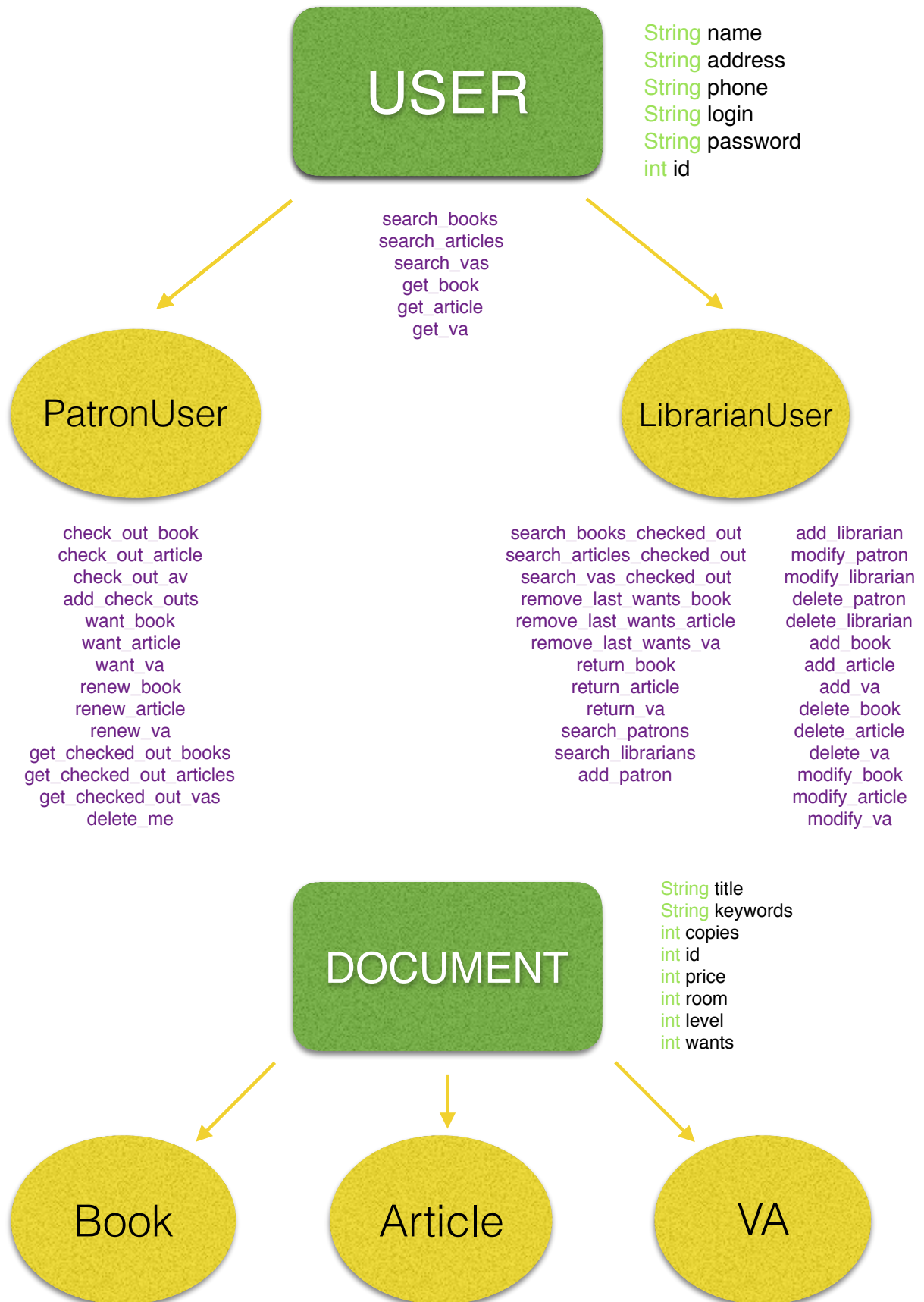
Nikolai Mikriukov

Nikola Novarlic

Ilnur Garipov

Kamil Shakirov





Class User:

- search_books, search_articles, search_vas

These three methods work both in LibrarianUser and in PatronUser. They search documents/vas in the system for various parameters. Here is an example of the code of one of the methods

```
QVector<Book> search_books(QString authors, QString title, QString keywords, QString publisher, int year, bool bestseller, bool available, bool or_and){
    QSqlQuery query;
    QString ins = or_and ? " AND " : " OR ";
    authors = authors.toLower();
    title = title.toLower();
    keywords = keywords.toLower();
    publisher = publisher.toLower();
    QString req = "SELECT * FROM books WHERE ";
    if (authors != "") req += "instr(lower(authors), '"+authors+"') > 0" + ins;
    if (title != "") req += "instr(lower(title), '"+title+"') > 0" + ins;
    if (keywords != "") req += "instr(lower(keywords), '"+keywords+"') > 0" + ins;
    if (publisher != "") req += "instr(lower(publisher), '"+publisher+"') > 0" + ins;
    if (year != 0) req += "instr(year, '"+QString::number(year)+"') > 0" + ins;
    if (bestseller) req += "bestseller = 1" + ins;
    if (available) req += "copies > 0" + ins;
    req += "1 = " + QString(or_and ? "1" : "0");//nice hack to finish statement correctly
    if (req.length() == 31)//no parameters given
        req = "SELECT * FROM books";
    query.exec(req);
    QVector<Book> ans;
    while (query.next()) {
        int id = query.value(0).toInt();
        QString title = query.value(1).toString();
        QString authors = query.value(2).toString();
        QString publisher = query.value(3).toString();
        QString keywords = query.value(4).toString();
        int year = query.value(5).toInt();
        int price = query.value(6).toInt();
        int room = query.value(7).toInt();
        int level = query.value(8).toInt();
        int copies = query.value(9).toInt();
        bool bestseller = query.value(10).toInt();
        bool reference = query.value(11).toInt();
        QString wants_str = query.value(12).toString();
        ans.push_back(Book(authors, title, keywords, publisher, id, year, copies, price, room, level, bestseller, reference, wants_str));
    }
    return ans;
}
```

Class User:

- `search_books`, `search_articles`, `search_vas`

Librarian

patrons librarians **books** articles audio/video checked out

author: title:

keywords: publisher:

year: ☐ bestseller ☐ available

☐ all criteria must be satisfied

	title	authors	publisher	keywords	year	price	room	level	copies	bestseller	reference	modify	delete
1	Turtles, all the way down	John Green	RELX Group	animal	2007	1400	5	15	12	no	no	<input type="button" value="modify"/>	<input type="button" value="delete"/>
2	The Hazel Wood	John Green	ThomsonReuters	history	1984	3000	5	21	2	no	no	<input type="button" value="modify"/>	<input type="button" value="delete"/>

Logged in as librarian: Master

Librarian

patrons librarians books articles **audio/video** checked out

author: title:

keywords: publisher:

☐ available

☐ all criteria must be satisfied

	title	authors	price	room	level	copies	reference	modify	delete
1	Revival	Eminem	3000	14	12	11	yes	<input type="button" value="modify"/>	<input type="button" value="delete"/>

Logged in as librarian: Master

Librarian

patrons librarians books articles **articles** audio/video checked out

author: title:

keywords: publisher:

editors: year: month: ☐ available

☐ all criteria must be satisfied

	title	authors	publisher	journal	editors	year
1	Work organization an...	Katia Levecqu...	Ghent Univ...	Research Poli...	Alain De Beuckelaer	2010

Logged in as librarian: Master

Class User:

- `get_book`, `get_article`, `get_va`

These three methods return the document/va by its identifier. Here is an example of the code of one of the methods

```
Book get_book(int book_id){
    QSqlQuery query;
    query.exec("SELECT * FROM books WHERE id = " + QString::number(book_id));
    query.next();
    QString title = query.value(1).toString();
    QString authors = query.value(2).toString();
    QString publisher = query.value(3).toString();
    QString keywords = query.value(4).toString();
    int year = query.value(5).toInt();
    int price = query.value(6).toInt();
    int room = query.value(7).toInt();
    int level = query.value(8).toInt();
    int copies = query.value(9).toInt();
    bool bestseller = query.value(10).toInt();
    bool reference = query.value(11).toInt();
    QString wants_str = query.value(12).toString();
    return Book(authors, title, keywords, publisher, book_id, year, copies, price, room ,level, bestseller, reference, wants_str);
}
```

Class PatronUser:

- `check_out_book`, `check_out_article`, `check_out_va`

These three methods realize the booking of documents/va. Here is an example of the code of one of the methods

```
void check_out_book(int document_id, QDate *gdate = NULL){
    QSqlQuery query;
    query.prepare("SELECT COUNT(*) FROM check_outs WHERE user_id = :user_id AND document_type = 1 AND document_id = :document_id AND year_end IS NULL");
    query.bindValue(":user_id", id);
    query.bindValue(":document_id", document_id);
    query.exec();
    query.next();
    bool already = query.value(0).toInt();
    if(already) return; //cant check out twice

    query.exec("SELECT copies, reference from books WHERE id = " + QString::number(document_id));
    if (!query.next()) return; //book doesnt exist
    int copies = query.value(0).toInt();
    bool reference = query.value(1).toInt();
    if (copies <= 0 || reference) return;

    QDate date;
    if(gdate == NULL)
        date = QDate::currentDate();
    else
        date = *gdate;
    int year_start = date.year();
    int month_start = date.month();
    int day_start = date.day();

    query.prepare("UPDATE books SET copies = copies - 1 WHERE id = :document_id");
    query.bindValue(":document_id", document_id);
    query.exec();

    query.prepare("INSERT INTO check_outs (user_id,document_type,document_id,year_start,month_start,day_start) VALUES(:user_id,:document_type,:document_id,:year_start,:month_start,:day_start)");
    query.bindValue(":user_id", id);
    query.bindValue(":document_type", BOOK);
    query.bindValue(":document_id", document_id);
    query.bindValue(":year_start", year_start);
    query.bindValue(":month_start", month_start);
    query.bindValue(":day_start", day_start);
    query.exec();

    //get last check out id
    query.exec("SELECT check_out_id FROM check_outs ORDER BY check_out_id DESC LIMIT 1");
    query.next();
    int check_out_id = query.value(0).toInt();

    query.prepare("UPDATE patrons SET check_outs = check_outs || :check_out_id WHERE id = :user_id");
    query.bindValue(":check_out_id", QString::number(check_out_id)+");");
    query.bindValue(":user_id", id);
    query.exec();
}
```

Class PatronUser:

- `check_out_book`, `check_out_article`, `check_out_va`

If user wants to reserve the document/va, and it is available, he clicks on the «check out" button and then this document/va is added the list «checked out»

Patron

books articles audio/video checked out

author: Jesmyn Ward title:

keyword: publisher:

year: ☐ bestseller ☐ available ☐ all criteria must be satisfied search

	title	authors	publisher	year	price	room	level	copies	bestseller	check out
1	Sing, Unburied, Sing	Jesmyn Ward	ThomsonReuters	2015	1560	5	5	3	yes	check out
2	The cruel prince	Jesmyn Ward	RELX Group	2011	2500	5	8	6	yes	check out

Logged in as patron: Kamil Shakirov delete my account Logout

Patron

books articles audio/video checked out

books

	title	authors	publisher	year	price	room	level	bestseller	date checked out	date to return	renew
1	Sing, Unburied, Sing	Jesmyn Ward	ThomsonReuters	2015	1560	5	5	yes	31.3.2018	14.4.2018	renew

articles

audio/video

Logged in as patron: Kamil Shakirov delete my account Logout

Class PatronUser:

- want_book, want_article, want_va

If user wants to reserve the document/va, but it is unavailable he clicks on the «want it» button and then this user is added to the queue to get this document/va. Here is an example of the code of one of the methods

```
void want_book(int document_id){
    //set renew_state to 1
    QSqlQuery query;
    query.prepare("UPDATE check_outs SET renew_state = 1 WHERE document_type = 1 AND renew_state = 0 AND year_end IS NULL AND document_id = :document_id");
    query.bindValue(":document_id", document_id);
    query.exec();

    Book book = get_book(document_id);
    if (!book.wants.contains(id)){
        query.prepare("UPDATE books SET wants = wants || :user_id WHERE id = :document_id");
        query.bindValue(":user_id", QString::number(id)+");");
        query.bindValue(":document_id", document_id);
        query.exec();
    }
}
```

The screenshot shows a web application window titled "Patron". It has a navigation bar with tabs: "books" (selected), "articles", "audio/video", and "checked out". Below the tabs are search fields for "author:" (containing "Finn, A. J."), "keyword:", "year:", "title:", and "publisher:". There are also checkboxes for "bestseller", "available", and "all criteria must be satisfied", along with a "search" button. Below the search fields is a table with the following data:

	title	authors	publisher	year	price	room	level	copies	bestseller	check out
1	The woman in the window	Finn, A. J.	Pearson	2018	1200	5	2	0	yes	<input type="button" value="want it"/>

The "want it" button in the "check out" column of the first row is highlighted with a yellow circle. At the bottom of the window, it says "Logged in as patron: Kamil Shakirov" and has two buttons: "delete my account" and "Logout".

Class PatronUser:

- `renew_book`, `renew_article`, `renew_va`

If user wants to extend the deadline for the document/va, and there are no other users in the booking queue for this document/va, user can click on the "renew" button and extend the deadline for the document/va for a certain period,

```
int renew_book(int check_out_id){
    QSqlQuery query;
    QDate today = QDate::currentDate();

    query.prepare("SELECT * FROM check_outs WHERE check_out_id = :check_out_id");
    query.bindValue(":check_out_id", check_out_id);
    query.exec();
    if (!query.next()) return 0; //no such check out

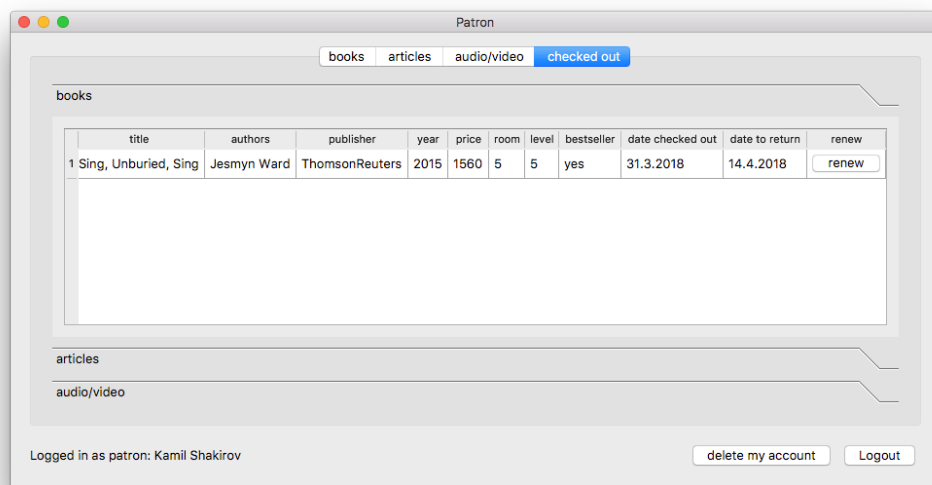
    int book_id = query.value(3).toInt();
    int year_start = query.value(4).toInt();
    int month_start = query.value(5).toInt();
    int day_start = query.value(6).toInt();
    int renew_state = query.value(10).toInt();

    if (renew_state != 0) return 1; //someone wants

    Book book = get_book(book_id);
    std::pair<QDate, int> end = calculate_check_out(1, year_start, month_start, day_start, faculty, book.bestseller, book.price, renew_state);

    if (today.daysTo(end.first) != 0 && today.daysTo(end.first) != 1)
        return 2; //too late or early

    query.prepare("UPDATE check_outs SET renew_state = 2 WHERE check_out_id = :check_out_id");
    query.bindValue(":check_out_id", check_out_id);
    query.exec();
    return 3;
}
```

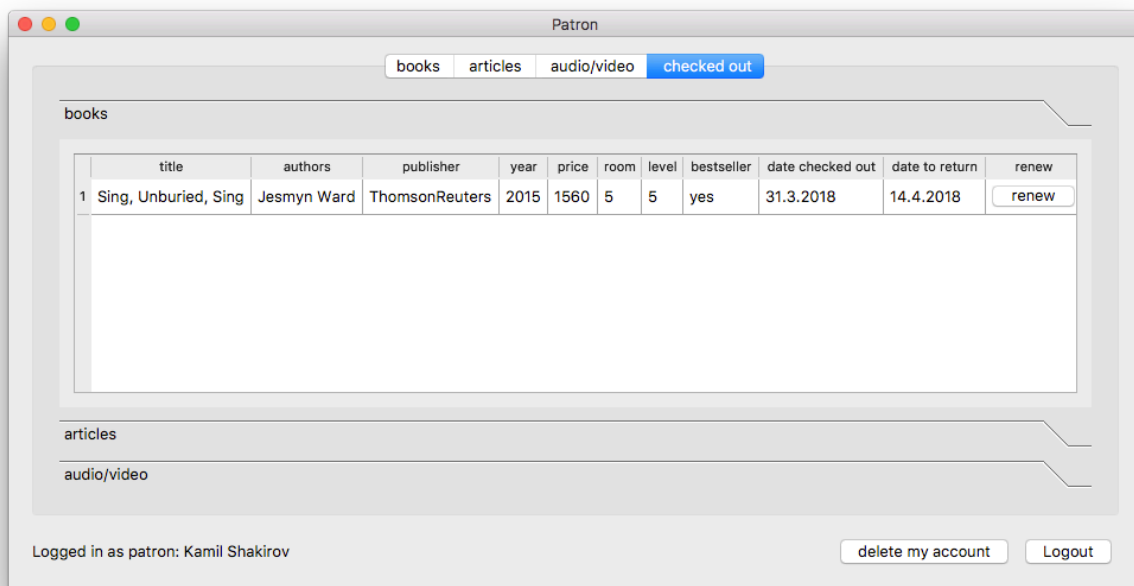


Class PatronUser:

- `get_check_out_book, get_check_out_article, get_check_out_va`

This method shows your checked out documents/vas

```
QVector<std::pair<Check_out, Book> > get_checked_out_books(){
    QSqlQuery query;
    QVector<std::pair<Check_out, Book> > ans;
    query.exec("SELECT * FROM check_outs WHERE document_type = 1 AND year_end IS NULL AND user_id = " + QString::number(id));
    while (query.next()) {
        int check_out_id = query.value(0).toInt();
        int book_id = query.value(3).toInt();
        int year_start = query.value(4).toInt();
        int month_start = query.value(5).toInt();
        int day_start = query.value(6).toInt();
        int state_renew = query.value(10).toInt();
        Book book = get_book(book_id);
        std::pair<QDate, int> end = calculate_check_out(1, year_start, month_start, day_start, faculty, book.bestseller, book.price, state_renew);
        ans.push_back(make_pair(Check_out(id, 1, book_id, check_out_id, year_start, month_start, day_start, end.first.year(), end.first.month(), end.first.day(), end.second), book));
    }
    return ans;
}
```

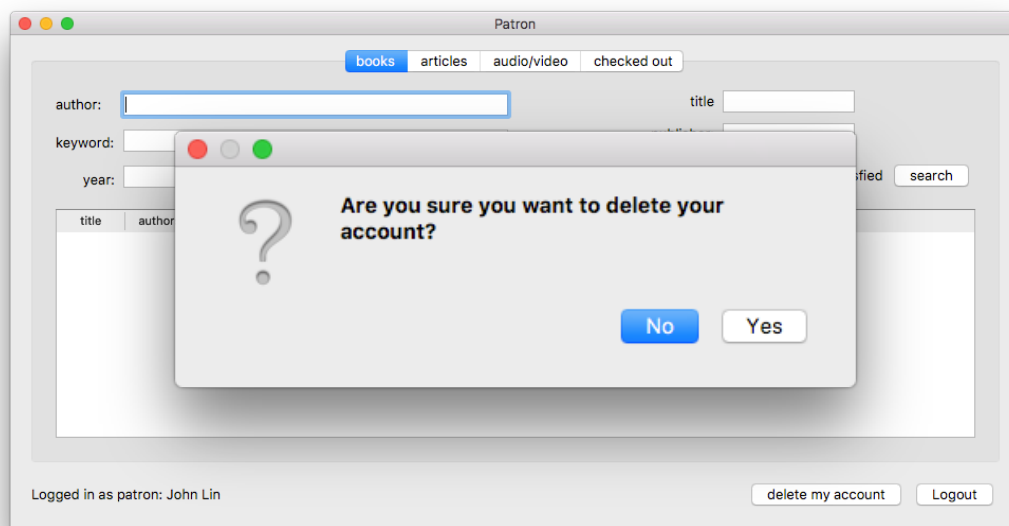


Class PatronUser:

- delete_me

Account deleting

```
bool delete_me(){
    QSqlQuery query;
    if (check_outs.size() > 0)
        return 0; //cant delete if has check outs
    query.exec("DELETE FROM patrons WHERE id = " + QString::number(id));
    return 1;
}
```



Class LibrarianUser:

- `search_books_checked_out`, `search_articles_checked_out`, `search_vas_checked_out`

Methods for searching checked out documents/vas

```
QVector<std::pair<Check_out, Book> > search_books_checked_out(int user_id, QString authors, QString title, QString keywords, QString publisher, int year, bool bestseller, bool overdue, bool or_and){
    QSqlQuery query;
    QVector<Book> books = search_books(authors, title, keywords, publisher, year, bestseller, 0, or_and);
    QVector<std::pair<Check_out, Book> > ans;
    for (int i = 0; i < books.size(); i++){
        QString req = "SELECT * FROM check_outs WHERE document_type = 1 AND year_end IS NULL AND document_id = " + QString::number(books[i].id);
        if (user_id != 0) req += QString(or_and ? " AND" : " OR") + " user_id = " + QString::number(user_id);
        query.exec(req);
        while (query.next()) {
            int check_out_id = query.value(0).toInt();
            int current_user_id = query.value(1).toInt();
            int book_id = query.value(3).toInt();
            int year_start = query.value(4).toInt();
            int month_start = query.value(5).toInt();
            int day_start = query.value(6).toInt();
            int state_renew = query.value(10).toInt();
            Book book = get_book(book_id);
            PatronUser patron = get_patron(current_user_id);
            std::pair<QDate, int> end = calculate_check_out(1, year_start, month_start, day_start, patron.faculty, book.bestseller, book.price, state_renew);
            QDate return_date;
            return_date.setDate(end.first.year(), end.first.month(), end.first.day());
            if (overdue && QDate::currentDate().daysTo(return_date) >= 0) continue;
            ans.push_back(make_pair(Check_out(current_user_id, 1, book_id, check_out_id, year_start, month_start, day_start, end.first.year(), end.first.month(), end.first.day(), end.second), book));
        }
    }
    return ans;
}
```

The screenshot shows the 'Librarian' application window with the 'checked out' tab selected. The interface includes search filters for books, a table of checked out items, and navigation buttons for other sections like patrons, librarians, books, articles, and audio/video.

Search Filters:

- author: Jesmyn Ward
- title:
- keywords:
- publisher:
- user id:
- year:
- ☐ bestseller ☐ overdue
- ☐ all criteria must be satisfied
-

Table of Checked Out Books:

	user id	title	authors	publisher	year	price	room	level	bestseller	date checked out	date to return	fine	return
1	4	Sing, Unburied, Sing	Jesmyn Ward	ThomsonReuters	2015	1560	5	5	yes	22.2.2018	22.3.2018	1000	<input type="button" value="return"/>
2	5	Sing, Unburied, Sing	Jesmyn Ward	ThomsonReuters	2015	1560	5	5	yes	31.3.2018	14.4.2018	0	<input type="button" value="return"/>

Navigation and Status:

- Logged in as librarian: Master
-

Class LibrarianUser:

- return_book, return_article, return_va

Returning system

```
std::pair<int, int> return_book(int check_out_id){
    QSqlQuery query;
    query.exec("SELECT * FROM check_outs WHERE document_type = 1 AND check_out_id = " + QString::number(check_out_id));
    if (!query.next()) return make_pair(-1, -1);
    int user_id = query.value(1).toInt();
    int book_id = query.value(3).toInt();
    int year_start = query.value(4).toInt();
    int month_start = query.value(5).toInt();
    int day_start = query.value(6).toInt();
    int renew_state = query.value(10).toInt();

    PatronUser patron = get_patron(user_id);
    Book book = get_book(book_id);
    std::pair<QDate, int> end = calculate_check_out(1, year_start, month_start, day_start, patron.faculty, book.bestseller, book.price, renew_state);
    int fine = end.second;

    query.prepare("UPDATE check_outs SET year_end = :year_end, month_end = :month_end, day_end = :day_end WHERE check_out_id = :check_out_id");
    query.bindValue(":check_out_id", check_out_id);
    query.bindValue(":year_end", end.first.year());
    query.bindValue(":month_end", end.first.month());
    query.bindValue(":day_end", end.first.day());
    query.exec();

    query.prepare("UPDATE patrons SET check_outs = replace(check_outs, :check_out_id_str, '') WHERE id = :user_id");
    query.bindValue(":check_out_id_str", ":" + QString::number(check_out_id));
    query.bindValue(":user_id", user_id);
    query.exec();

    query.prepare("UPDATE books SET copies = copies + 1 WHERE id = :document_id");
    query.bindValue(":document_id", book.id);
    query.exec();

    query.prepare("DELETE FROM check_outs WHERE check_out_id = :check_out_id");
    query.bindValue(":check_out_id", check_out_id);
    query.exec();

    int wants_id = remove_last_wants_book(book.id);
    if (wants_id != -1)
        return make_pair(fine, wants_id);
    return make_pair(fine, -1);
}
```

Librarian

patrons librarians books articles audio/video **checked out**

books

author: Jesmyn Ward title: _____

keywords: _____ publisher: _____

user id: _____ year: _____ ☐ bestseller ☐ overdue

☐ all criteria must be satisfied

	user id	title	authors	publisher	year	price	room	level	bestseller	date checked out	date to return	fine	return
1	4	Sing, Unburied, Sing	Jesmyn Ward	ThomsonReuters	2015	1560	5	5	yes	22.2.2018	22.3.2018	1000	<input type="button" value="return"/>
2	5	Sing, Unburied, Sing	Jesmyn Ward	ThomsonReuters	2015	1560	5	5	yes	31.3.2018	14.4.2018	0	<input type="button" value="return"/>

articles

audio/video

Logged in as librarian: Master

Class LibrarianUser:

- search_patrons, search_librarians

These methods are search users in the system

```
QVector<PatronUser> search_patrons(int user_id, QString name, QString address, QString phone, bool faculty, bool or_and){
    QSqlQuery query;
    QString ins = or_and ? " AND " : " OR ";
    name = name.toLower();
    address = address.toLower();
    phone = phone.toLower();
    QString req = "SELECT * FROM patrons WHERE ";
    if (user_id != 0) req += "id = " + QString::number(user_id) + ins;
    if (name != "") req += "instr(lower(name), '"+name+"') > 0" + ins;
    if (address != "") req += "instr(lower(address), '"+address+"') > 0" + ins;
    if (phone != "") req += "instr(lower(phone), '"+phone+"') > 0" + ins;
    if (faculty) req += "faculty = 1" + ins;
    req += "1 = " + QString(or_and ? "1" : "0"); //nice hack to finish statement correctly
    if (req.length() == 33) //no parameters given
        req = "SELECT * FROM patrons";
    query.exec(req);
    QVector<PatronUser> ans;
    while (query.next()) {
        int user_id = query.value(0).toInt();
        QString name = query.value(1).toString();
        QString address = query.value(2).toString();
        QString phone = query.value(3).toString();
        bool faculty = query.value(4).toInt();
        QString check_outs = query.value(5).toString();
        QString login = query.value(6).toString();
        QString password = query.value(7).toString();
        ans.push_back(PatronUser(user_id, name, address, phone, faculty, login, password, check_outs));
    }
    return ans;
}
```

The screenshot shows a window titled "Librarian" with several tabs: "patrons" (selected), "librarians", "books", "articles", "audio/video", and "checked out". Below the tabs are input fields for "name:" (containing "Rollo"), "address:", "phone:", and "id:". There are checkboxes for "all criteria must be satisfied" and "faculty". "search" and "new patron" buttons are present. Below these is a table with columns: id, name, address, phone, faculty, login, modify, and delete. The table contains one row with the data: 1, 3, Rollo Drake, Kazan Street 12, 202-555-0173, no, drake66. The "modify" and "delete" buttons are visible next to this row. At the bottom, it says "Logged in as librarian: Master" and has "settings" and "Logout" buttons.

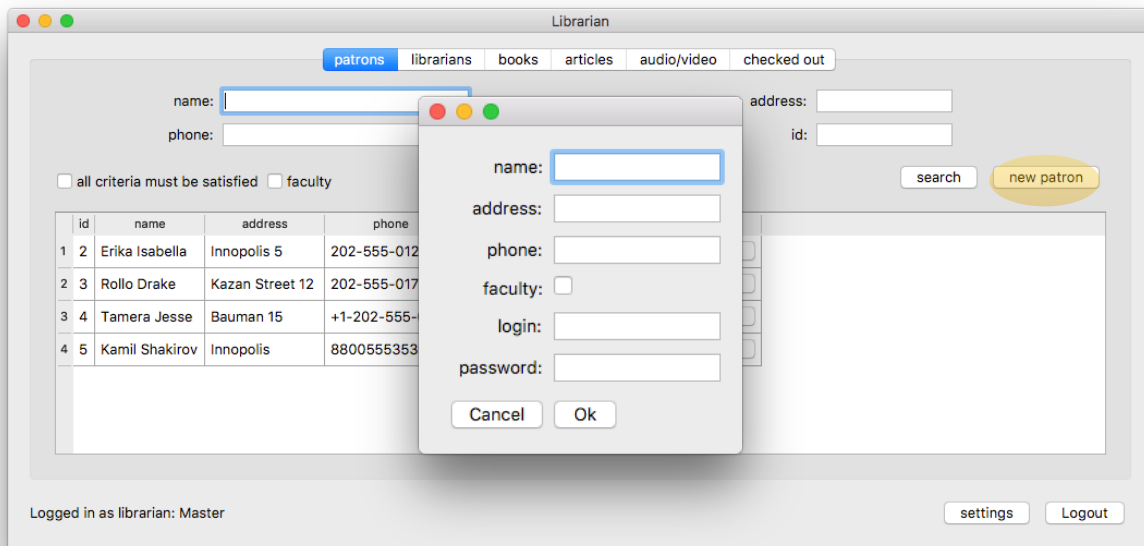
id	name	address	phone	faculty	login	modify	delete
1	3	Rollo Drake	Kazan Street 12	202-555-0173	no	drake66	modify delete

Class LibrarianUser:

- add_patron, add_librarian

With these methods, librarian can add new users

```
void add_patron(QString name, QString address, QString phone, bool faculty, QString login, QString password){
    password = Hasher::hash_password(login,password);
    QSqlQuery query;
    query.prepare("INSERT INTO patrons (name, address, phone, faculty, login, password) VALUES(:name, :address, :phone, :faculty, :login, :password)");
    query.bindValue(":name", name);
    query.bindValue(":address", address);
    query.bindValue(":phone", phone);
    query.bindValue(":faculty", (faculty ? 1 : 0));
    query.bindValue(":login", login);
    query.bindValue(":password", password);
    query.exec();
}
```

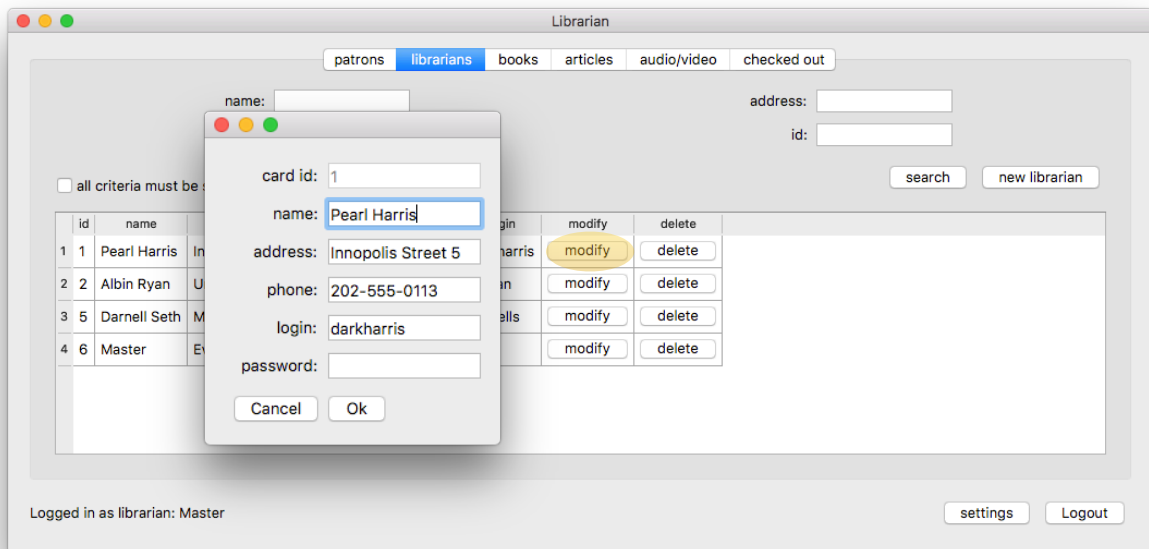


Class LibrarianUser:

- `modify_patron`, `modify_librarian`

With these methods, librarian can modify users

```
bool modify_librarian(int user_id, QString name, QString address, QString phone, QString login, QString password){
    password = Hasher::hash_password(login,password);
    QSqlQuery query;
    query.prepare("UPDATE librarians SET name = :name, address = :address, phone = :phone, login = :login, password = :password WHERE id = :user_id");
    query.bindValue(":name", name);
    query.bindValue(":address", address);
    query.bindValue(":phone", phone);
    query.bindValue(":login", login);
    query.bindValue(":password", password);
    query.bindValue(":user_id", user_id);
    query.exec();
    return 1;
}
```



Class LibrarianUser:

- delete_patron, delete_librarian

Deleting users

```
bool delete_patron(int user_id){
    QSqlQuery query;
    if (get_patron(user_id).check_outs.size() > 0)
        return 0; //cant delete if has check outs
    query.exec("DELETE FROM patrons WHERE id = " + QString::number(user_id));
    return 1;
}
bool delete_librarian(int user_id){
    QSqlQuery query;
    query.exec("DELETE FROM librarians WHERE id = " + QString::number(user_id));
    return 1;
}
```

The screenshot shows a web application window titled "Librarian". At the top, there are tabs: "patrons", "librarians" (which is selected), "books", "articles", "audio/video", and "checked out". Below the tabs, there are input fields for "name:", "address:", "phone:", and "id:". Below these fields is a checkbox labeled "all criteria must be satisfied". To the right of the checkbox are two buttons: "search" and "new librarian". Below this is a table with the following data:

	id	name	address	phone	login	modify	delete
1	1	Pearl Harris	Innopolis Street 5	202-555-0113	darkharris	<button>modify</button>	<button>delete</button>
2	2	Albin Ryan	University Square 8	+1-202-555-0161	alRyan	<button>modify</button>	<button>delete</button>
3	5	Darnell Seth	Moscow Street 16	+1-202-555-0155	darnells	<button>modify</button>	<button>delete</button>
4	6	Master	Everywhere	89196655567	2	<button>modify</button>	<button>delete</button>

At the bottom of the window, there is a status bar that says "Logged in as librarian: Master". To the right of the status bar are two buttons: "settings" and "Logout".

Class LibrarianUser:

- add_book, add_article, add_va

Methods for adding new document/va to the system

```
bool add_va(QString title, QString authors, QString keywords, int price, int room, int level, int copies, bool reference){
    QSqlQuery query;
    query.prepare("INSERT INTO vas (title, authors, keywords, price, room, level, copies, reference) VALUES(:title, :authors, :keywords, :price, :room, :level, :copies, :reference)");
    query.bindValue(":title", title);
    query.bindValue(":authors", authors);
    query.bindValue(":keywords", keywords);
    query.bindValue(":price", price);
    query.bindValue(":room", room);
    query.bindValue(":level", level);
    query.bindValue(":copies", copies);
    query.bindValue(":reference", reference);
    query.exec();
    return 1;
}
```

The screenshot shows the 'Librarian' application window with the 'audio/video' tab selected. The main window contains input fields for 'author:', 'keywords:', 'title:', and 'publisher:'. There is a checkbox for 'all criteria must be satisfied' and a 'search' button. A modal dialog is open in the center, allowing the user to add a new audio/video item. The dialog has input fields for 'title:', 'authors:', 'keywords:', 'price:', 'room:', 'level:', 'copies:', and a checkbox for 'reference:'. The 'new audio/video' button in the main window is highlighted with a yellow oval. At the bottom of the main window, it says 'Logged in as librarian: Master' and has 'settings' and 'Logout' buttons.

Class LibrarianUser:

- delete_book, delete_article, delete_va

Methods for deleting documents/vas from the system

```
bool delete_book(int document_id){
    QSqlQuery query;
    query.exec("DELETE FROM books WHERE id = " + QString::number(document_id));
    return 1;
}
bool delete_article(int document_id){
    QSqlQuery query;
    query.exec("DELETE FROM articles WHERE id = " + QString::number(document_id));
    return 1;
}
bool delete_va(int document_id){
    QSqlQuery query;
    query.exec("DELETE FROM vas WHERE id = " + QString::number(document_id));
    return 1;
}
```

The screenshot shows a window titled "Librarian" with a tabbed interface. The "books" tab is selected. The search area includes fields for "author:" (Jesmyn Ward), "keywords:", "title:", "publisher:", and "year:". There are checkboxes for "bestseller" and "available", and a checkbox for "all criteria must be satisfied". "search" and "new book" buttons are present. Below is a table with 14 columns: title, authors, publisher, keywords, year, price, room, level, copies, bestseller, reference, modify, and delete. Two rows are visible, both for books by Jesmyn Ward. The "delete" button in the first row is highlighted.

	title	authors	publisher	keywords	year	price	room	level	copies	bestseller	reference	modify	delete
1	Sing, Unburied, Sing	Jesmyn Ward	ThomsonReuters	fiction science	2015	1560	5	5	2	yes	no	modify	delete
2	The cruel prince	Jesmyn Ward	RELX Group	love nature history	2011	2500	5	8	6	yes	no	modify	delete

Logged in as librarian: Master

settings Logout

Class LibrarianUser:

- `modify_book`, `modify_article`, `modify_va`

Methods for documents/vas modification

```
bool modify_va(int document_id, QString title, QString authors, QString keywords, int price, int room, int level, int copies, bool reference){
    QSqlQuery query;
    query.prepare("UPDATE vas SET title = :title, authors = :authors, keywords = :keywords, price = :price, room = :room, level = :level, copies = :copies, reference = :reference WHERE id = :document_id");
    query.bindValue(":title", title);
    query.bindValue(":authors", authors);
    query.bindValue(":keywords", keywords);
    query.bindValue(":price", price);
    query.bindValue(":room", room);
    query.bindValue(":level", level);
    query.bindValue(":copies", copies);
    query.bindValue(":document_id", document_id);
    query.bindValue(":reference", reference);
    query.exec();
    return 1;
}
```

