Description of the methods

1. class Login

   ```
   static LibrarianUser login_librarian(QString login, QString password);

   static PatronUser login_patron(QString login, QString password);
   ```

   return (user_id, name, address, phone, faculty, user_login, user_password, check_outs) if user with such login and password exist, otherwise NULL

2. Modify articles

   ```
   bool modify_va(int document_id);

   bool modify_article(int document_id);

   bool modify_book(int document_id);
   ```

   Returning true if article with that document_id exist and modify operation is done, otherwise false

3. Delete articles

   ```
   bool delete_book(int document_id);

   bool delete_article(int document_id);

   bool delete_va(int document_id);
   ```

   Returns true if article is found and deleted

4. Inserting new articles

   ```
   bool add_book(QString title, QString authors, QString keywords, int price, int room, int level, int copies, bool reference);

   bool add_article(QString title, QString authors, QString keywords, int price, int room, int level, int copies, bool reference);

   bool add_va(QString title, QString authors, QString keywords, int price, int room, int level, int copies, bool reference);
   ```

   Return false if same article already exist, otherwise insert that article in db and return true

5. Delete users

   ```
   bool delete_patron(int user_id);

   bool delete_librarian(int user_id);
   ```

   Return false in user with that id is not found, otherwise delete that user from db and return true

6. Get information about users based on their user_id

   ```
   LibrarianUser get_librarian(int user_id);

   PatronUser get_patron(int user_id);
   ```

   Get all data about Library user with *user_id* if such user exist, otherwise return NULL

7. Modify data about users

   ```
   bool modify_patron(int user_id, QString name, QString address, QString phone, int patron_type, QString login, QString password)

   bool modify_librarian(int user_id, QString name, QString address, QString phone, QString login, QString password)
   ```

   It will return false if user don't exist in the data base, otherwise retruns true and replaces old data with this given as input parameters

8. Adding new users

   ```
   void add_librarian(QString name, QString address, QString phone, QString login, QString password)

   void add_patron(QString name, QString address, QString phone, int patron_type, QString login, QString password)
   ```

   Insert input parameters into db

9. Get the list of users which satisfy given conditions

   ```
   QVector<LibrarianUser> search_librarians(int user_id, QString name, QString address, QString phone, bool or_and)

   QVector<PatronUser> search_patrons(int user_id, QString name, QString address, QString phone, int patron_type, bool or_and)
   ```

   Get users with: this.user_id = user_id and/or
   this.name = name and/or
   this.adress = addres and/or
   this.phone = phone

10. Returning articles

    ```
    std::pair<int, int> return_va(int check_out_id)

    std::pair<int, int> return_article(int check_out_id)

    std::pair<int, int> return_book(int check_out_id)
    ```

    When user returns back the article, functions return pair of integers, first integer is fine, second is, if there are no more articles of that kind in library, waiting id - pointing on first user in priority queue who wants that book, or -1 if there are more availible copies of the article

11. Query: How many days are predicted for renew

    ```
    int get_days_add_renew();
    ```

12. Set new value for days renew

    ```
    void set_settings(int days_add_renew);
    ```

13. Remove last member from article queue

    ```
    int remove_last_wants_book(int document_id);

    int remove_last_wants_article(int document_id);

    int remove_last_wants_va(int document_id);
    ```

    If no one wants that article function returns -1, otherwise removes that member from the queue and returns his user_id

14. Search for checked out items, also possibility to search for items checked out by certain user

    ```
    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)

    QVector<std::pair<Check_out, Article> > search_articles_checked_out(...);

    QVector<std::pair<Check_out, VA> > search_vas_checked_out(...);
    ```

    Returns list of all pairs (check out, item) requested by conditions

15. User deletes his own account

    ```
    bool delete_me();
    ```

    Returns false if user have some check outs, otherwise deletes user from the db and returns true

16. User asks for his checked out items

```
QVector<std::pair<Check_out, VA> > get_checked_out_vas();

QVector<std::pair<Check_out, Article> > get_checked_out_articles();

QVector<std::pair<Check_out, Book> > get_checked_out_books();
```

Function returns list of all items that user own

17. User wants to renew some item

```
int renew_va(int check_out_id);

int renew_article(int check_out_id);

int renew_book(int check_out_id);
```

Returns 0 if there is no check out with that id;
Returns 1 if someone else wants that book;
Returns 2 if it is too late or too early for renew;
Renews the check out and returns 3 otherwise.

18. User wants some item

```
void want_book(int document_id);

void want_article(int document_id);

void want_va(int document_id);
```

When all items of that id are taken, procedure updates renew state of item and inserts user in queue for that item

19. User check out item

```
void check_out_book(int document_id, QDate *gdate = NULL);

void check_out_article(int document_id, QDate *gdate = NULL);

void check_out_av(int document_id, QDate *gdate = NULL);
```

First checks if user already have that item and doensn’t allow to take it twice. Then checks if item is reference or if there are no more copies (calls f.e. want_book in that case). Otherwise decreases number of item copies, inserts data inside check_outs and adds new *check out* to user’s *check_outs*.

```
void add_check_outs(QString str);
```

This procedure parses string and adds user's check outs.

20. Get items

    ```
    VA get_va(int va_id);

    Article get_article(int article_id);

    Book get_book(int book_id);
    ```

    Functions take item id and return all atributes from db about that item.

21. Get list of items with certain conditions

    ```
    QVector<Book> search_books(QString authors, QString title, QString keywords, QString publisher, int year, bool bestseller, bool available, bool or_and)

    QVector<Article> search_articles(QString authors, QString title, QString keywords, QString journal, QString publisher, QString editors, int year, int month, bool available, bool or_and)

    QVector<VA> search_vas(QString authors, QString title, QString keywords, bool available, bool or_and)
    ```

    As result function will return list of items that satisfy all or just one condition given by user (depends on or_and variable)

22. Calculate date on which user should return item

    ```
    std::pair<QDate, int> calculate_check_out(int document_type, int year_start, int month_start, int day_start, int patron_type, bool bestseller, int price, int renew_state)
    ```

    At, first program will check how many days should add depends on item type and user's job, check if item is renewed and add more days that's true. Calculates return date by adding days_to_add to start date. Calculate days overdue and fine. Returns end_date and fine.

23. Get queue of users that want certain *book/article/av*

    ```
    static Queue getFromDB(int item_id, int item_type);
    ```