```ruby
# return the login page
get '/login' do
    erb :"Login/index"
end

# post login form data, evaluate and set session values as required
# params;
#   username
#   password
post '/login' do

    @username = params[:username]
    @password = params[:password]

    success = User.authenticate(@username, @password)

    if success then
        user = User.getByUsername(@username)

        session[:loggedIn] = true
        session[:username] = user.username
        session[:userId] = user.userId
        session[:admin] = user.admin
        redirect '/bookmarks/all'
    end

    @loginError = "Username or password isn't correct."

    erb :"Login/index"
end

# logout the currently logged in user
get '/login/logout' do
    session.clear
    redirect '/login'
end
```

Controllers Login.rb

```erb
    <div class="col quart side-nav">
        <h5>
            Information
        </h5>
        <p>Author <span class="right-align"><%= @user.username %></span></p>
        <p>Date <span class="right-align"><%= @bookmark.createdAt %></span></p>

        <br>
        <ul>
            <li><a class="btn favourite">Favourite</a></li>
            <li><a href="/bookmarks/report/<%= @bookmark.bookmarkId %>"class="btn report">Report</a></li>
            <li><a class="btn edit">Edit</a></li>
        </ul>
    </div>
</div>

<%= erb :"Shared/footer" %>

<script>
    var elem = document.getElementById('toggle-dropdown');

    elem.onclick = function () {

        var dropElem = document.querySelector('.new-tags-container');

        dropElem.classList.toggle('active');
    };
</script>
```

Views/Bookmarks/view.erb

```css
form input[type="text"], input[type="password"], input[type="url"] {
    font-size: 14px;
    padding: 5px 0;
    background: transparent;
    border: none;
    border-bottom: solid black 1px;
}

form input[type="submit"] {
    font-family: 'Sen';
    font-size: 16px;
    padding: 7.5px 15px;
    color: white;
    background-color: black;
    border: black 1px solid;
    cursor: pointer;
}
form input[type="submit"]:hover {
    transition: 0.2s;
    background-color: white;
    color: black;
}
```

Public/css/style.css

```ruby
class Bookmark

    DB = SQLite3::Database.new 'database.db'

    def initialize(bookmarkId, createdAt, title, description, resource, archived, userId)
    end
    def self.newBookmark(title, description, resource, archived, userId)

        query = "INSERT INTO bookmarks('title', 'description', 'resource', 'archived', 'userId', 'createdAt') VALUES(?, ?, ?, ?, ?, ?);"

        begin
            DB.execute query, title, description, resource, archived, userId, Time.now.inspect
        rescue SQLite3::Exception
            return false
        end

        return true
    end
    # Return all the known bookmarks in the database as Bookmark objects
    # Returns: an array of Bookmark objects
    def self.getAll

        toReturn = []

        result = DB.execute "SELECT * FROM bookmarks;"

        for bookmark in result do

            bookmarkObj = Bookmark.new(bookmark[0], bookmark[1], bookmark[2], bookmark[3], bookmark[4], bookmark[5], bookmark[6])
            toReturn.push(bookmarkObj)

        end

        return toReturn
    end

    # Return all the known bookmark  in the database as Bookmark objects
    # Returns: an array of Bookmark objects
    def self.getByTitle
        toReturn = []

        result = DB.execute "SELECT title, createdAt FROM bookmarks ORDER BY createdAt ASC;"

        for bookmark in result do

            bookmarkObj = Bookmark.new(nil, bookmark[1], bookmark[2], nil, nil, nil, nil)
            toReturn.push(bookmarkObj)

        end
        return toReturn
    end
```

Models Bookmark.rb

```ruby
class Tag

    DB = SQLite3::Database.new 'database.db'

    def initialize(tagId, tag)↔
    def self.getByBookmarkId(bookmarkId)

        toReturn = []

        query = "SELECT
                    bookmarks_to_tags.bookmarkId,
                    bookmarks_to_tags.tagId,
                    tags.tag
                FROM
                    bookmarks_to_tags
                INNER JOIN tags ON bookmarks_to_tags.tagId = tags.tagId
                WHERE
                    bookmarks_to_tags.bookmarkId = ?;"

        result = DB.execute query, bookmarkId

        for tag in result do

            newTag = Tag.new(tag[1], tag[2])
            toReturn.push(newTag)

        end

        return toReturn
    end
    def self.newTag(tag, bookmarkId)

        tagId = Tag.doesTagExist(tag)

        if tagId != nil then

            if Tag.doesLinkExist(bookmarkId, tagId) == false then

                Tag.createLink(bookmarkId, tagId)

            else

                return false

            end
        else

            Tag.createTag(tag)
            tagId = doesTagExist(tag)
            Tag.createLink(bookmarkId, tagId)

        end

        return true
    end

```

Models Tag.rb