

UNIVERSITY OF AUCKLAND ASSIGNMENT SUBMISSION DOCUMENT

COURSE : COMPSCI 235 (Second Semester, 2021)

FILE : Assignment 2 Design Report

STUDENT : Simon Shan (441147157)

Overview

Feature ONE: Book cover images and language

Feature TWO: LINQ like support

Feature THREE: Policy-based authorisation

Feature FOUR: Salt hashed password

Feature FIVE: Following other users, social page

Feature ONE | Book cover images and language

The figure consists of four screenshots of a web-based library application, likely built with ASP.NET Core, demonstrating various features related to book covers and language.

- Screenshot 1: Home Screen**
Shows the main landing page with a dark green header "Kia Ora, Welcome to Library™". It features three large icons: "About Us" (info bubble), "Log in" (user profile), and "Register" (pencil). Below is a section titled "Our Catalogue" displaying book covers for "The Switchblade Mammas", "Cruelle", "Captain America: Winter Soldier (The Ultimate Graphic Novel Collection: Publication Order, #7)", and "Superman".
- Screenshot 2: Catalogue View**
Shows a grid view of books in the catalogue. Each book entry includes a thumbnail, title, and a brief description. Books shown include "The Switchblade Mammas", "Cruelle", "Captain America: Winter Soldier (The Ultimate Graphic Novel Collection: Publication Order, #7)", "Bounty Hunter 4.0: My Life in Combat from Marine Scout S...", and others.
- Screenshot 3: Book Detail View**
Shows a detailed view of the book "Captain America: Winter Soldier (The Ultimate Graphic Novel Collection: Publication Order, #7)". It includes the book cover, author information ("By: Ed Brubaker"), a short description, and two buttons: "Mark as read" and "Write a review".
- Screenshot 4: Search Catalogue**
Shows a search interface titled "Search Our Catalogue". It includes a search bar and dropdown menus for "Title", "Author", "Release Year", and "Publisher". Below the search bar, results for "Captain America: Winter Soldier (The Ultimate Graphic Novel Collection: Publication Order, #7)" and "Naoki Urasawa's 20th Century Boys, Volume 19 (20th Centu..." are displayed.

Feature TWO | LINQ like support

Language Integrated Query (LINQ) is a popular backend package for Microsoft ASP.NET development. It is only available in C#, until now.

https://en.wikipedia.org/wiki/Language_Integrated_Query

In this project, I adapted the LINQ like features for python. The magic is in the `library/adapters/dataset.py` file. Benefits include reduced code duplication and time saved.

```
def get_publishers_names(self):
    '''get all publishers' names, no duplicates'''
    return self._database.books.select(lambda book: book.publisher) \
        .select(lambda publisher: publisher.name) \
        .remove_dups()
```

Books → Publishers → names → remove duplicates

```
discover = _repo.get_users() \
    .remove(user) \
    .where(lambda other: other not in user.following) \
    .order_by(lambda other: other.num_followers(), reverse=True)
```

All Users → remove current user → only keep users the current user is not following → order by number of followers, descending order

Feature THREE | Policy-based authorisation

Policy-based authorisation. <https://docs.microsoft.com/en-us/aspnet/core/security/authorization/policies?view=aspnetcore-5.0>

Once again, I adapted a C# package for python. The magic is in `library/handlers/authorisation.py`. Benefits include scalability to other policies, such as age restriction.

```
### authorisation policies ###

from .authorisation import Policy
from ..adapters import _repo

useronly = Policy(session_key='username',
                  verification_function=_repo.username_exists)
```

Creating a policy that only authorises users to perform a certain action.

```
@blueprint.route('/social')
@authorisation(policy=useronly)
def social():
    user = repo.get_user(username=ses
```

Other possible policies include only users with a certain score or followers or age may perform certain actions.

Feature FOUR | Salt hashed password

As is commonly known, storing passwords in plaintext is bad. What is less commonly known is that storing passwords in only hashes is also not good security practise. This is because if two users have the same password, their password hashes will also be the same. Therefore, knowing one of the passwords will compromise the other.

The solution to this issue is salting. [https://en.wikipedia.org/wiki/Salt_\(cryptography\)](https://en.wikipedia.org/wiki/Salt_(cryptography))

A salt is randomly generated at registration. It is then prepended to the plaintext password before hashing.

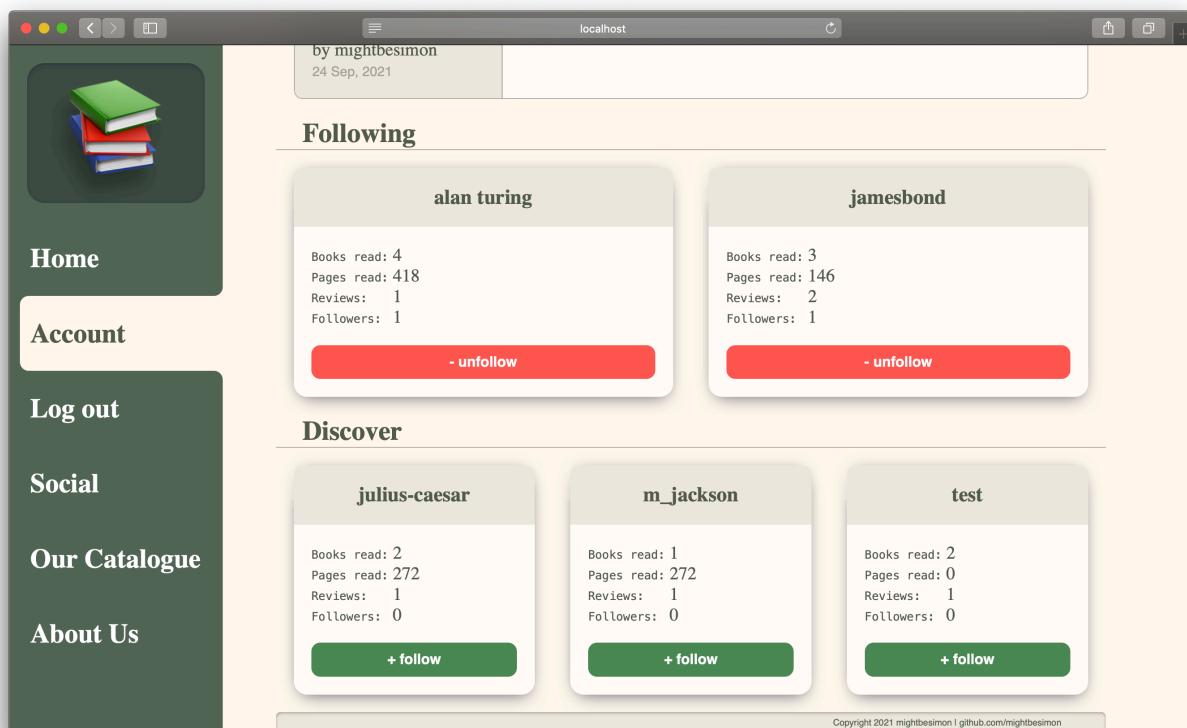
Bad	<code>sha256(plaintext) → hash</code>
Good	<code>sha256(salt+plaintext) → hash</code>
Stored as	<code>f'{hash}:{salt}'</code>

Since the salt is randomly generated, the same passwords will not produce the same hash. The salt is stored along side with the password hash. At authentication, we just need to re-salt the inputted password with the stored salt and see if it matches the stored hash.

`sha256(stored_salt+input) → stored_hash`

Feature FIVE | Following other users, social page

Basically like Instagram for books. The user can follow and unfollow other people. Each little card displays some information on the other users, namely, number of, books read, pages read, reviews wrote and followers. Cards are displayed using flex box, so they reactively adjust the width based on the number of items. Benefits include user-user interaction and marketability.



The screenshot shows the Bookshelf application's Account page. On the left, a sidebar menu has 'Account' selected. The main content area displays a 'Reviews' section with one item from 'The Switchblade Mamma' and a 'Following' section which is currently empty. A 'Discover' section shows profiles for 'alan turing', 'julius-caesar', 'm_jackson', and 'jamesbond', each with a '+ follow' button. The bottom right corner of the page includes a copyright notice.

User	Books read	Pages read	Reviews	Followers
alan turing	4	418	1	0
julius-caesar	2	272	1	0
m_jackson	1	272	1	0
jamesbond	3	146	2	0

Copyright 2021 mightbesimon | github.com/mightbesimon

In the Account page, if the number of people is too large, it does not overflow down the page. The user can scroll horizontally to view more, just like Netflix.

The screenshot shows the Bookshelf application's Account page with a large number of users followed. The 'Following' section is filled with multiple user profiles, each with a red '- unfollow' button. The 'Discover' section is also populated with many profiles. A message at the bottom indicates that the user is following everyone.

User	Books read	Pages read	Reviews	Followers
alan turing	4	418	1	1
jamesbond	3	146	2	1
julius-caesar	2	272	1	1
m_jackson	1	272	1	1

You are following everyone 😳
There is no one else left.

Copyright 2021 mightbesimon | github.com/mightbesimon