EDUCATIONAL PURPOSES ONLY
PRIVATE WORK
ALL RIGHTS RESERVED

Updated 24 Sep, 2021
Due 24 Sep, 2021

UNIVERSITY OF AUCKLAND ASSIGNMENT SUBMISSION DOCUMENT

COURSE : COMPSCI 235 (Second Semester, 2021)

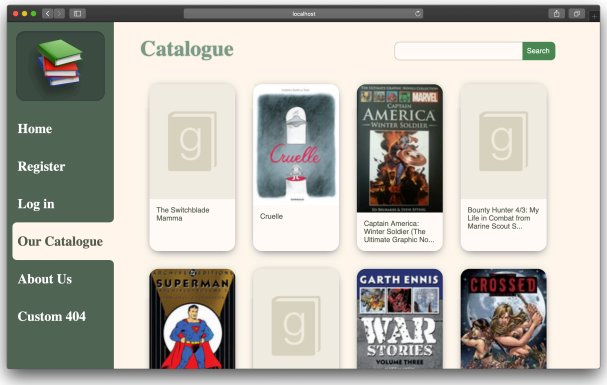
FILE : Assignment 2 Design Report

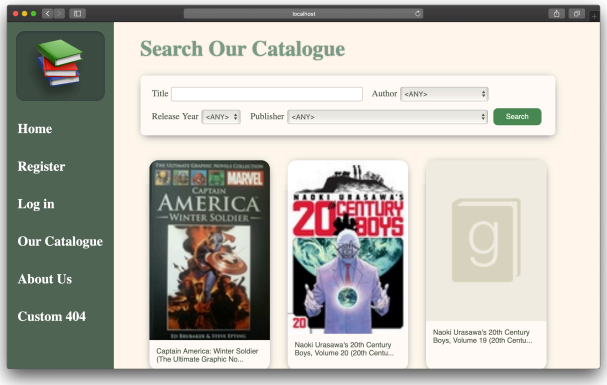
STUDENT : Simon Shan (441147157)

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





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_dupes()
```

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

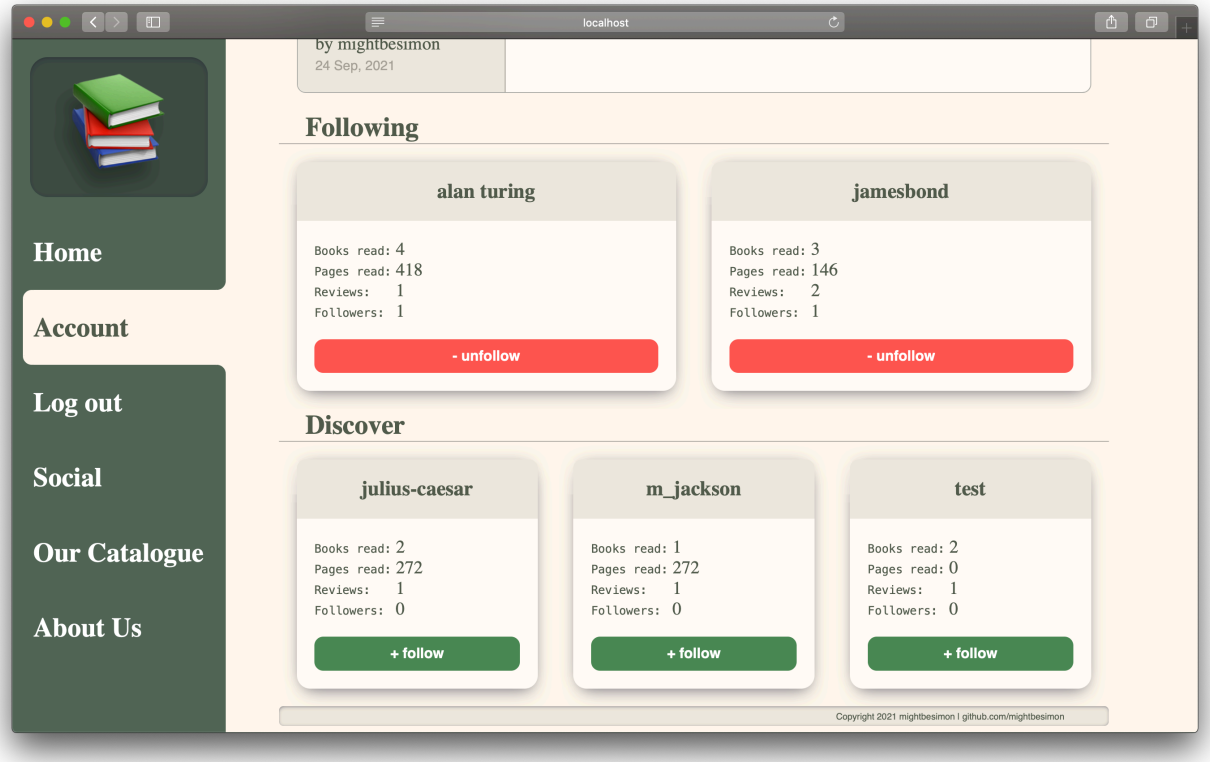
Bad	`sha256(plaintext) → hash`
Good	`sha256(salt+plaintext) → hash`
Stored as	`f'{hash}:{salt}'`

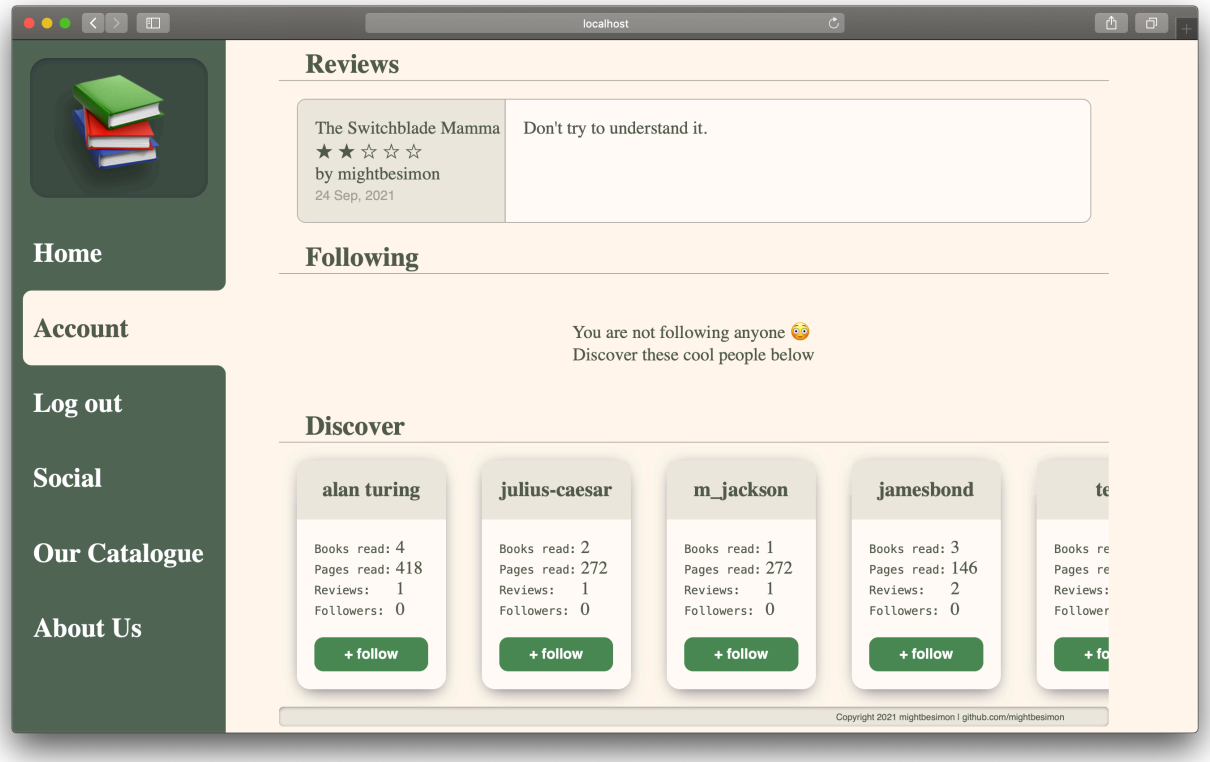
Since the salt is randomly generated, the same passwords will not produce the same hash. The salt is stored along side with the password hash. At authentication, we just need to re-salt the inputted password with the stored salt and see if it matches the stored hash.

```
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 abject the width based on the number of items. Benefits include user-user interaction and marketability.





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.