Frank Lu

Dr. Pulimood, Professor Devlin

CSC415-01

Assignment 4 – Open Source Software: Analysis and Design

Project Info

Requirements

The main goal of ShelfAwareness is to provide users with three things – a place to catalog and take notes on books they've read, a place to discuss all things literature related, and a way to ger books to those who cannot afford them.

ShelfAwareness has a catalog function that allows users to record what they have read, are reading, and will read in the future. Further, users can also add details to books, make custom lists, and mark books as favorites,

The donation and wish list function simply allows users to put a book on their wishlist, while other users in the community are able to see a queue of all of the site's 'wishes' and donate books at will.

Finally, the forums of the site are where users can go to discuss any topic, so long as it is literature related. Forums are organized by general topics, and moderated by a team of users to make sure everyone follows the code of conduct and terms of service.

Diagrams

Use Case Descriptions

Use Case: Access and edit personal catalog of books

Iteration: 2 – Last modification: March 4 by Frank Lu

Primary Actor: 'Library' Owner

Goal in context: To view and edit personal catalog of books, marking titles as owned, read, or for future reading

Preconditions: User has an existing account they know the credentials of.

Trigger: The user decides to view the books they have read in the past, to add a book they have recently read, or to mark a book down to be read in the future.

Scenario:

1. The user logs onto the *ShelfAwareness* website
2. The user enters their user ID or email
3. The user enters their password (at least 8 characters, including 1 capital and 1 number)

4. The system displays all major function buttons
5. The user selects ‘catalog’ from the major function buttons on the home screen
6. The site displays all of the user’s books, in alphabetical order, separated into books that have been read (past), are in the process of being read (present), or books that they plan to read (future)
7. The user chooses to add a book to that list, marking down if its for the past, present, or future
8. The user removes a book from the list
9. The user sorts list alphabetically, by date added, by release date, by genre, etc.
10. The user marks certain books as ‘favorites’
11. The user selects ‘view details’ for a book
12. System displays basic information about the book (Author, genre, series, publication date, etc.)
13. System gives option to view ‘personal notes’ on selected book – see use case **Access and edit personal notes**
14. Users mark books as ‘favorites’
15. User makes custom lists of books

Exceptions:

1. ID or passwords are incorrect or not recognized – see use case **Validate ID and password**

Priority: High; this is one of two basic functions

When available: Third increment

Frequency of use: Extremely frequent

Channel to actor: Via PC-based browser and internet connection

Secondary actors: None

Channels to secondary actors: N/A

Open issues:

1. Is there an existing database of books to draw from? Or does the user need to find and implement details manually?
2. Will all information be customizable? How will this be managed/organized? Will poems, web serials, etc. be included?

Use Case: Validate ID and password

Iteration: 1

Primary Actor: ‘Library’ Owner

Goal in context: Allow users to reset password using email validation link

Preconditions: User has an existing account tied to an email address

Trigger: User has failed to provide login credentials more than twice and has confirmed prompt to be provided a new password.

Scenario:

1. ID or passwords are incorrect or not recognized
2. The user retries at least once and fails again
3. The user is prompted to reset their password
4. The user is asked for their email
5. The user provides an email which is linked to

Exceptions:

1. User does not have an existing account
2. User does not have access to the email address associated with their account

Priority: Low; Implement basic functionality before accounts

When available: Later increments

Frequency of use: Necessary for users to recover accounts

Channel to actor: Via PC-based browser and internet connection

Secondary actors: Administrators/moderators in case users need assistance

Channels to secondary actors: Via PC-based browser and internet connection

Open issues:

1. Functionality for user accounts not yet implemented
2. Keeping user information secure

Use Case: Access and edit personal notes

Iteration: 1

Primary Actor: 'Library' Owner

Goal in context: Allow users to record personal notes and favorite parts of books they have read

Preconditions: The user has an existing account, is logged in, and already has the desired book in their catalog.

Trigger: User decides there is a particular aspect of the book they wish to take note of or record.

Scenario:

1. The user selects a specified book from their catalog
2. The user records a favorite quote
    a. The user cites the quote
    b. The user makes a note about the quote
3. The user records their notes about the book, thoughts, details, etc.

4. Write an academic analysis/paper
5. Give the book a rating (1-10 scale)

Exceptions: The data/object storing already existing details is unable to be fetched or fetched incorrectly

Priority: Moderate to high; One of the main features of the application

When available: Iteration 2 or 3

Frequency of use: Moderate to high; depends on intentions of users

Channel to actor: Via PC-based browser and internet connection

Secondary actors: None

Channels to secondary actors: N/A

Open issues:

1. How will efficient storage of all notes be implemented? Will there be space limitations?

Use Case: Access and manage personal wish list

Iteration: 2

Primary Actor: 'Library Owners'

Goal in context: Allow users who don't have access to books receive donations of books.

Preconditions: User has an existing account and is logged in.

Trigger: Users sees book they want but cannot afford and asks for it as a donation

Scenario:

1. User adds a book to their wish list
2. User adds a note/description of why they want a book
3. User edits description/note
4. User removes a book from their wish list

Exceptions:

1. When user attempts to edit wish list, data is fetched incorrectly
2. User lists a book they have already read/owned
3. User lists a book that does not exist
4. User's request/note is flagged as inappropriate

Priority:

When available: Iteration 2

Frequency of use: High; Main feature of application

Channel to actor: Via PC-based browser and internet connection

Secondary actors: Donors, moderators/administrators of site

Channels to secondary actors: Via PC-based browser and internet connection

Open issues:

1. How will the users receive books? How will the private information of the users stay secure?

Use Case: Donate book

Iteration: 2

Primary Actor: 'Library' Owners

Goal in context: Allow owners of books to donate them to others, or buy them for others

Preconditions: Users have a book someone has requested, or has the money to buy them

Trigger: A user decides to donate a book or money to someone who wants it

Scenario:

1. User selects 'Donations' button from home page to access donations page
2. User views list of requests for books. Sorted by oldest first by default, but can also be sorted by most recent first, cost, both descending and ascending, or by users
3. User chooses book to donate
4. User specifies if they are paying for a copy of the book or donating a copy of the book
5. User specifies payment methods
6. User cancels donation
7. User puts a book up for adoption

Exceptions:

1. User attempts to donate to someone who has already received donation in the time it took to fill out information.

Priority: High; Primary purpose of application

When available: Iteration 2

Frequency of use: High

Channel to actor: Via PC-based browser and internet connection

Secondary actors: Users requesting donations

Channels to secondary actors: Via PC-based browser and internet connection

Open issues:

1. How will donors send books?

2. If donors choose to donate money, how will cost of books be determined? Where will books be purchased from?

Use Case: Access community forums/discussions

Iteration: 3

Primary Actor: 'Library' owners

Goal in context: Promote and foster discussion regarding various topics related to books and literature on an open forum

Preconditions: Users have access to an existing account and are logged in.

Trigger: User wishes to discuss something they have read/learned

Scenario:

1. User selects 'Forums' button from home page to navigate to forums
2. User can view list of general topics of discussion, and pick which section their discussion belongs in
3. User creates a new discussion, creating a thread within that greater general topic
    a. User titles post
    b. User composes initial post
    c. User saves draft of post
    d. User completes, saves, and submits post
4. User adds onto an already existing thread
    a. User composes post
    b. User submits post
5. User flags a post as inappropriate
6. User searches all posts for keywords

Exceptions:

1. User attempts to comment on a post which has since been removed.
2. User is an administrator/moderator and able to review reported posts and delete them. **See Use Case: Moderation of forums and wish list**
3. Thread information is not fetched correctly

Priority: Low; basic functionality first

When available: Iteration 3 or 4

Frequency of use: Low

Channel to actor: Via PC-based browser and internet connection

Secondary actors: Any other users on the site

Channels to secondary actors: Via PC-based browser and internet connection

Open issues:

1. How will moderators be chosen? What exactly are moderators able to do? How will moderators ensure community guidelines are followed?
2. How will forum threads be organized?

Use Case: Moderation of forums and wish list

Iteration: 4+

Primary Actor: Site moderators or administrators

Goal in context: Allow specific users to moderate social aspects of site to ensure community guidelines are met.

Preconditions: Users have an existing account which is also marked as having moderator privileges

Trigger: Users decide to fulfill some moderator responsibilities

Scenario:

1. User views reports through home screen button visible only to them
2. User selects report and reads through description, before passing judgement
    a. Report is closed/resolved because no rules have been broken
    b. Offending user is banned from social aspects of site
    c. Offending user is suspended from social aspects of site
3. Moderator pins forum thread to front page for visibility
4. Moderator deletes thread/post
5. Moderator locks thread from any further replies

Exceptions:

1. Moderators disagree upon judgements
2. Threads are fetched incorrectly

Priority: Low; basic functionality must come first

When available: Iteration 4 or 5

Frequency of use: Low

Channel to actor: Via PC-based browser and internet connection

Secondary actors: All non-moderators

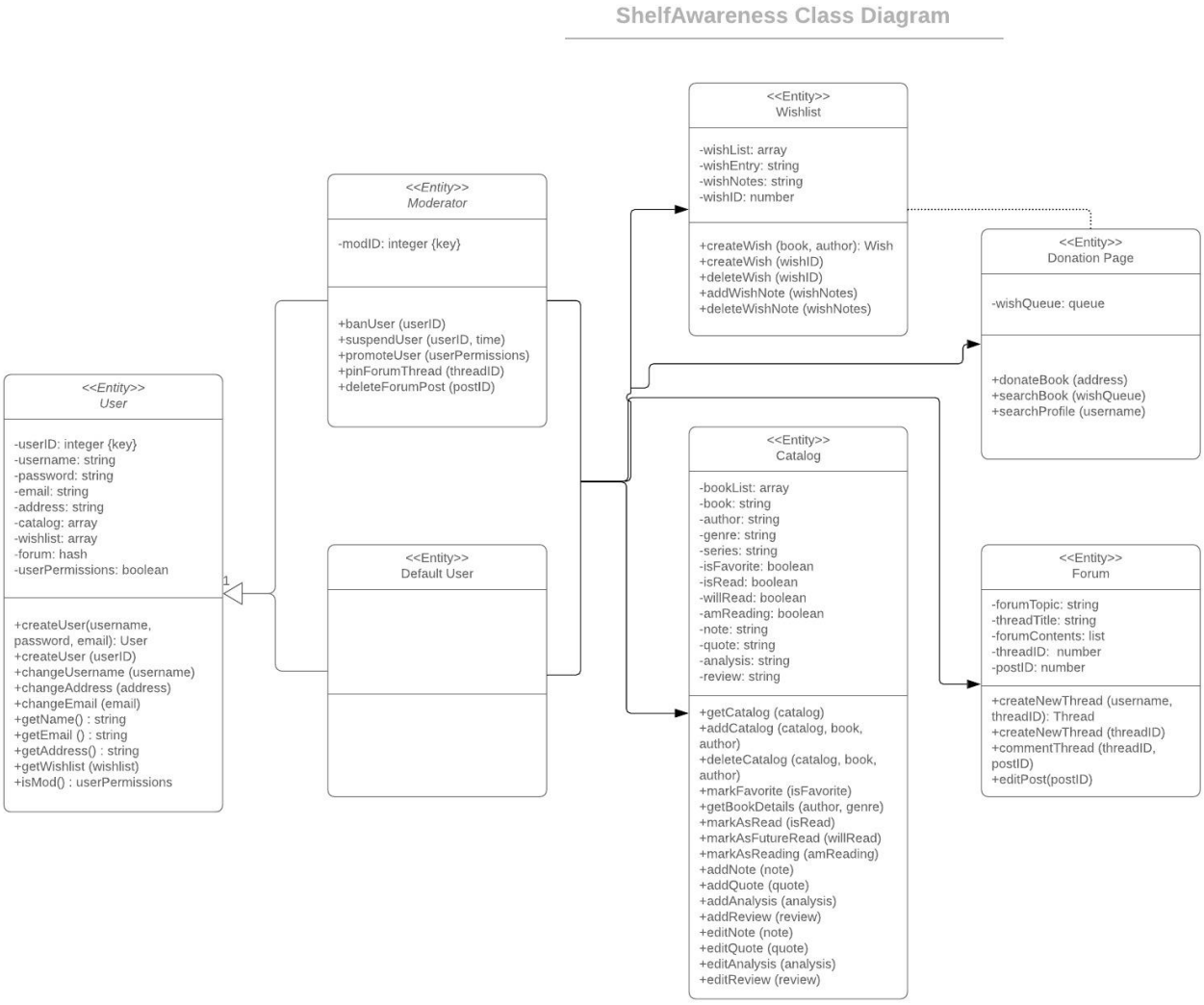
Channels to secondary actors: Via PC-based browser and internet connection

Open issues:

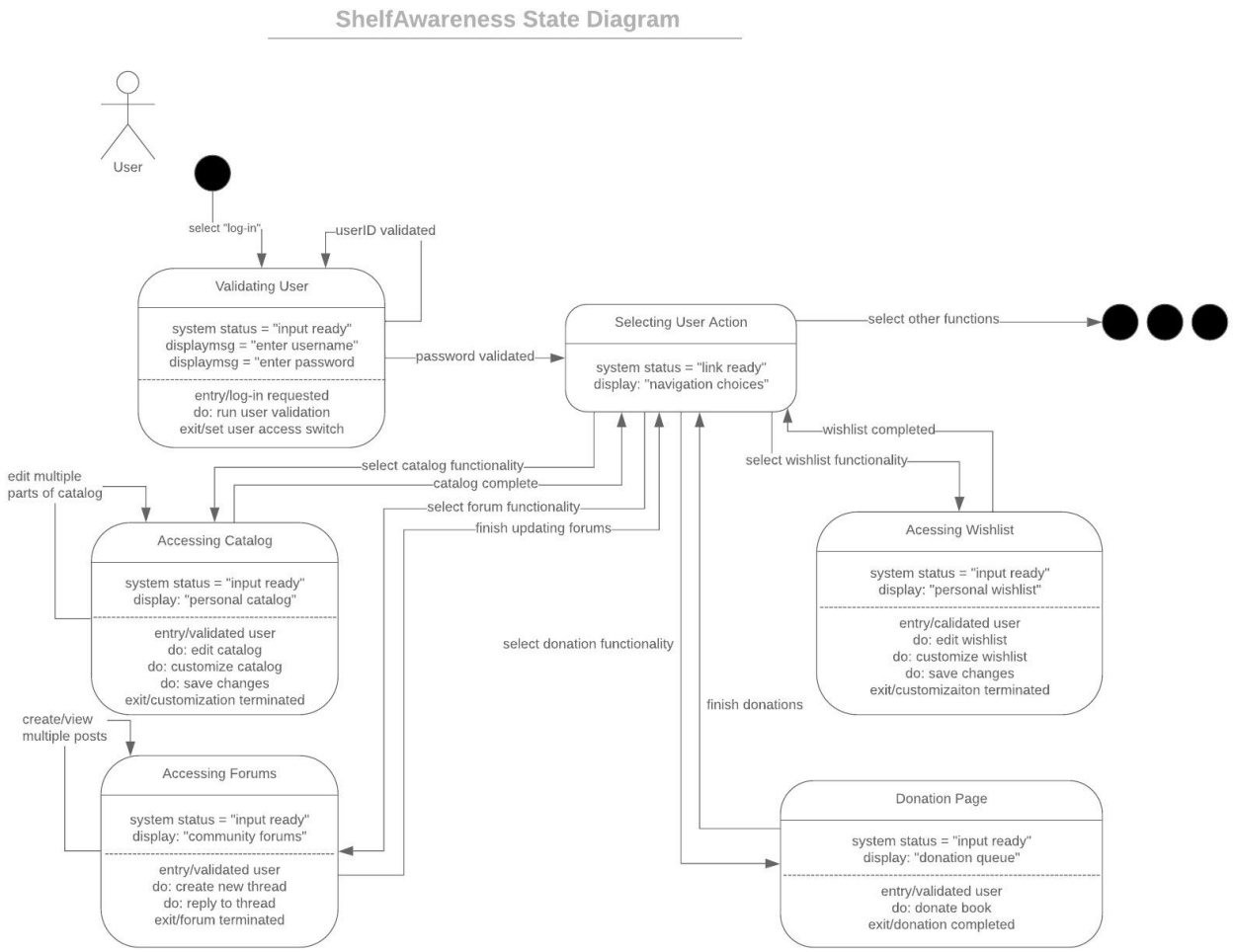
1. How will moderators be assigned?
2. How will accounts be promoted to moderators?

3. How does one give different permissions to view different versions of the site using different account types?

Detailed Design Class Diagram

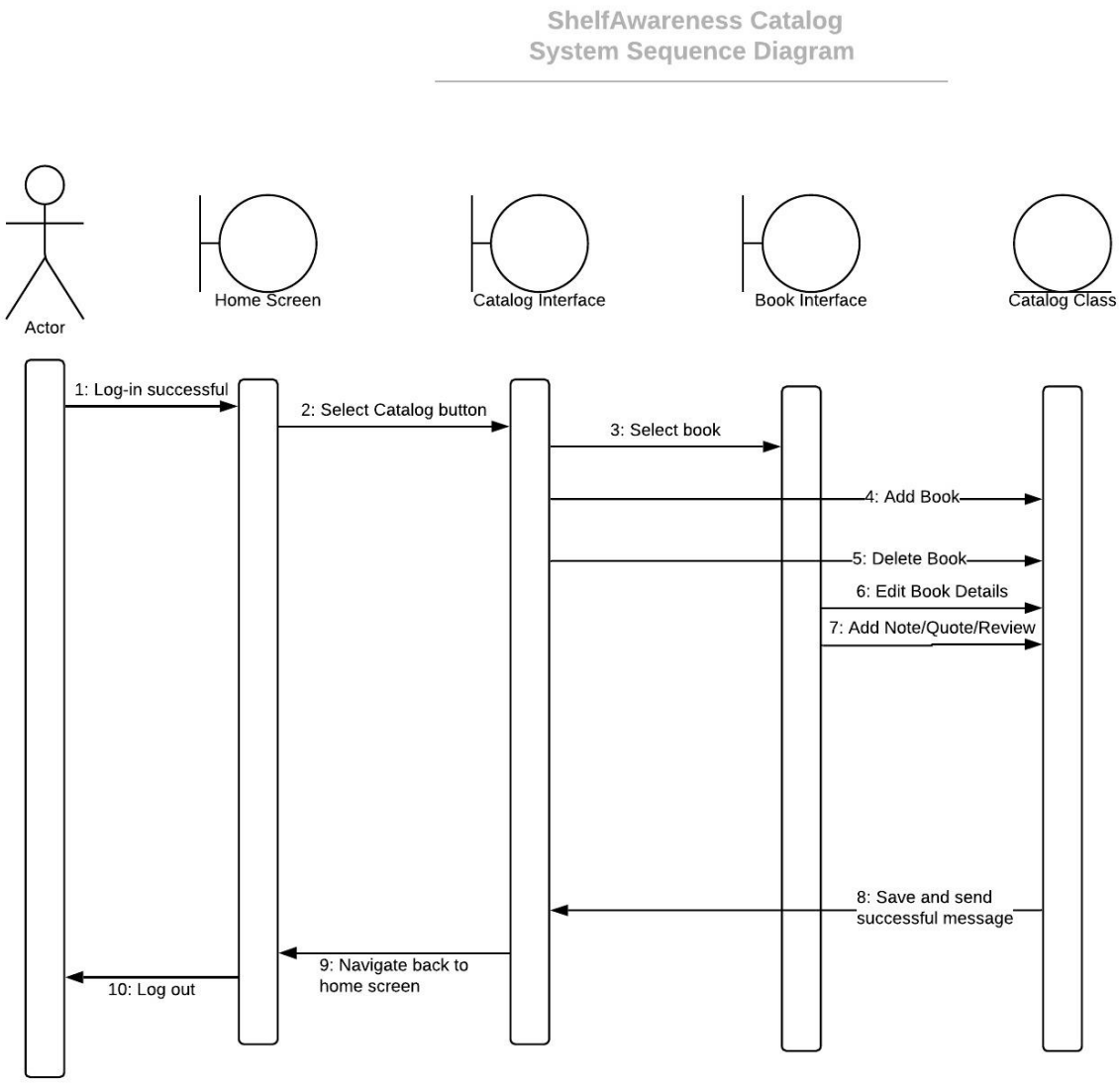


State chart

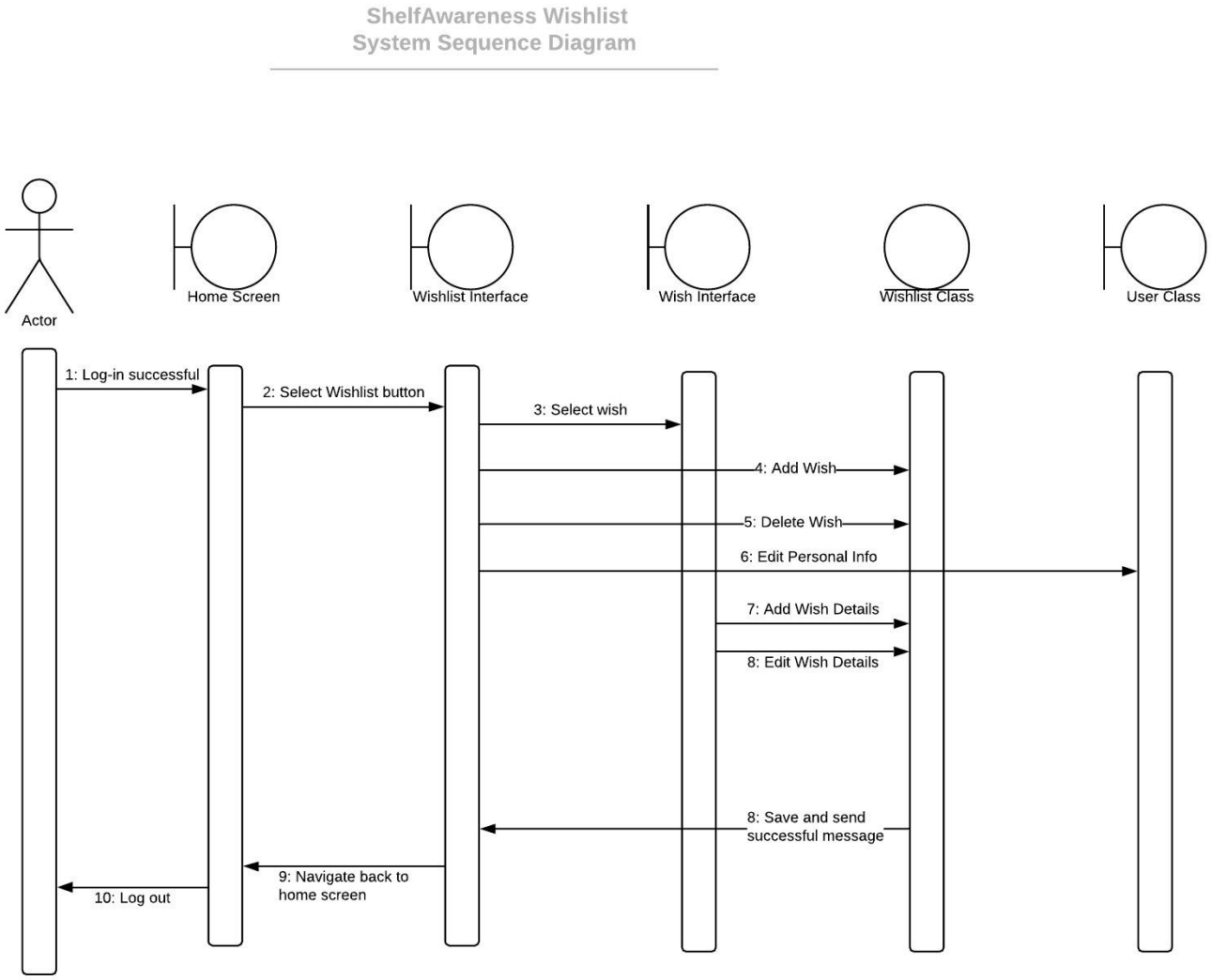

ShelfAwareness State Diagram
User
select "log-in"
userID validated
Validating User
system status = "input ready"
displaymsg = "enter username"
displaymsg = "enter password
entry/log-in requested
do: run user validation
exit/set user access switch
password validated
Selecting User Action
system status = "link ready"
display: "navigation choices"
select other functions
wishlist completed
select wishlist functionality
select catalog functionality
catalog complete
select forum functionality
finish updating forums
edit multiple
parts of catalog
Accessing Catalog
system status = "input ready"
display: "personal catalog"
entry/validated user
do: edit catalog
do: customize catalog
do: save changes
exit/customization terminated
Acessing Wishlist
system status = "input ready"
display: "personal wishlist"
entry/calidated user
do: edit wishlist
do: customize wishlist
do: save changes
exit/customizaiton terminated
select donation functionality
finish donations
create/view
multiple posts
Accessing Forums
system status = "input ready"
display: "community forums"
entry/validated user
do: create new thread
do: reply to thread
exit/forum terminated
Donation Page
system status = "input ready"
display: "donation queue"
entry/validated user
do: donate book
exit/donation completed

System Sequence Diagrams

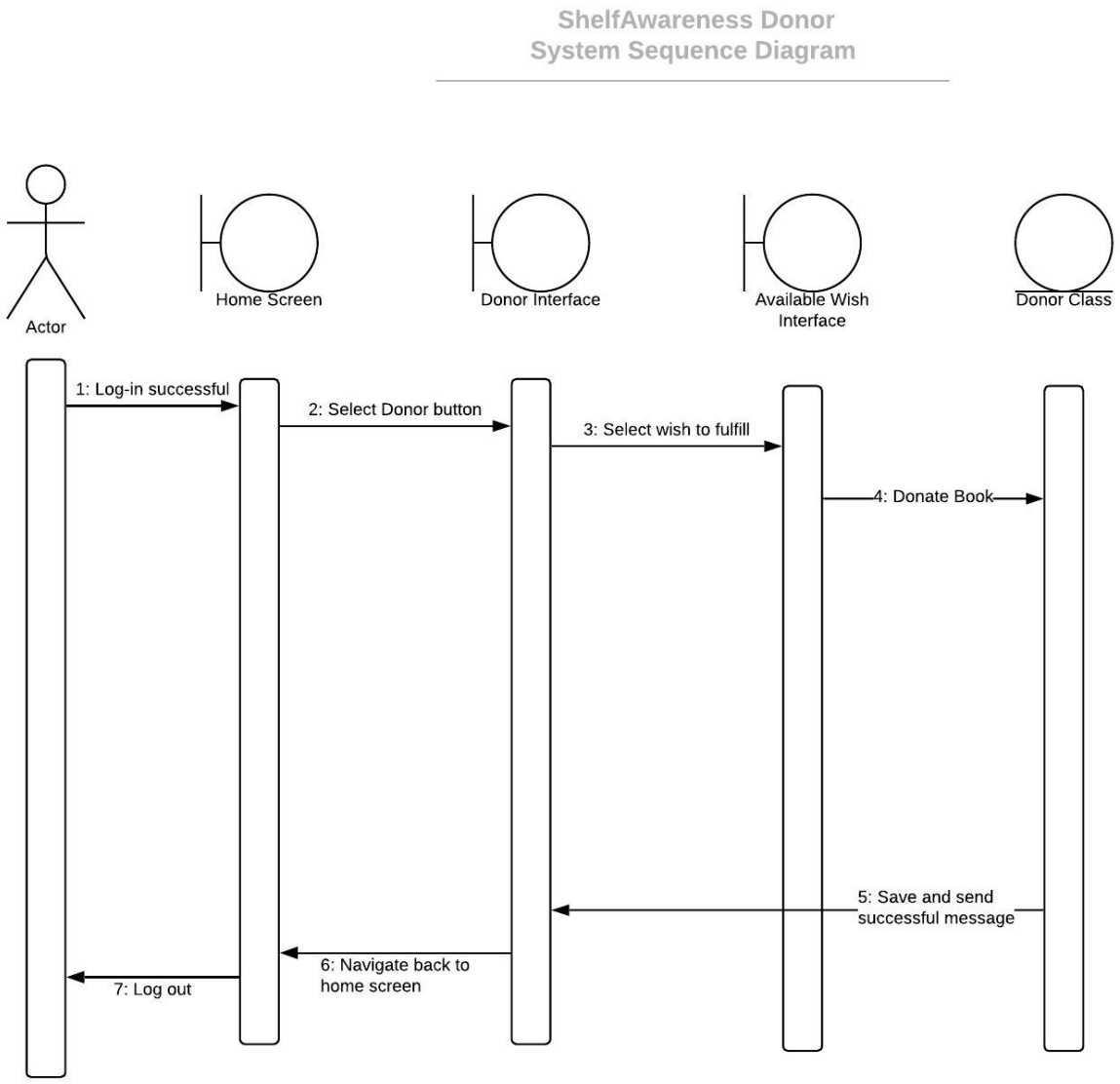
Catalog



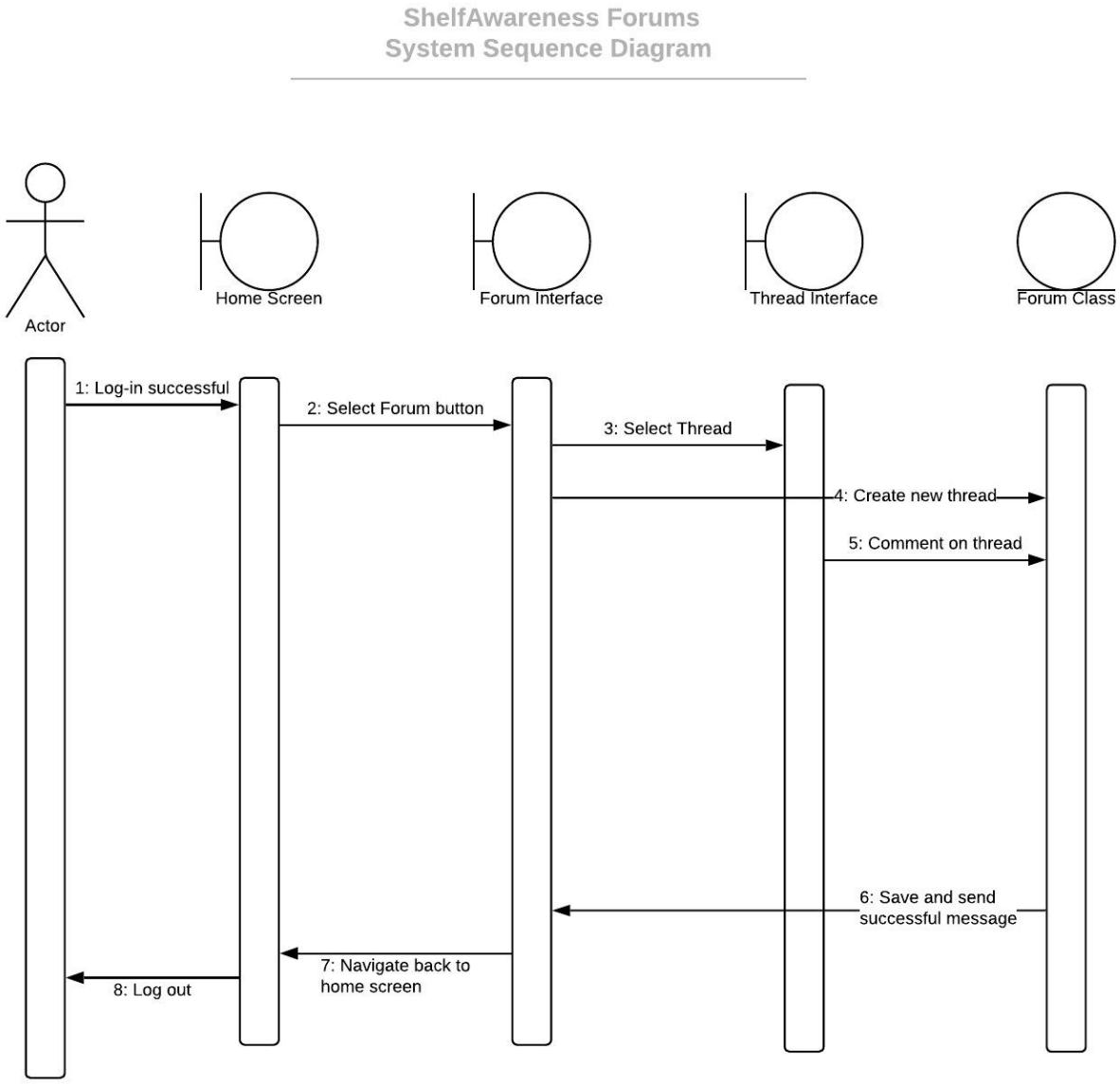
Wishlist


ShelfAwareness Wishlist
System Sequence Diagram
Actor
Home Screen
Wishlist Interface
Wish Interface
Wishlist Class
User Class
1: Log-in successful
2: Select Wishlist button
3: Select wish
4: Add Wish
5: Delete Wish
6: Edit Personal Info
7: Add Wish Details
8: Edit Wish Details
8: Save and send successful message
9: Navigate back to home screen
10: Log out

Donation

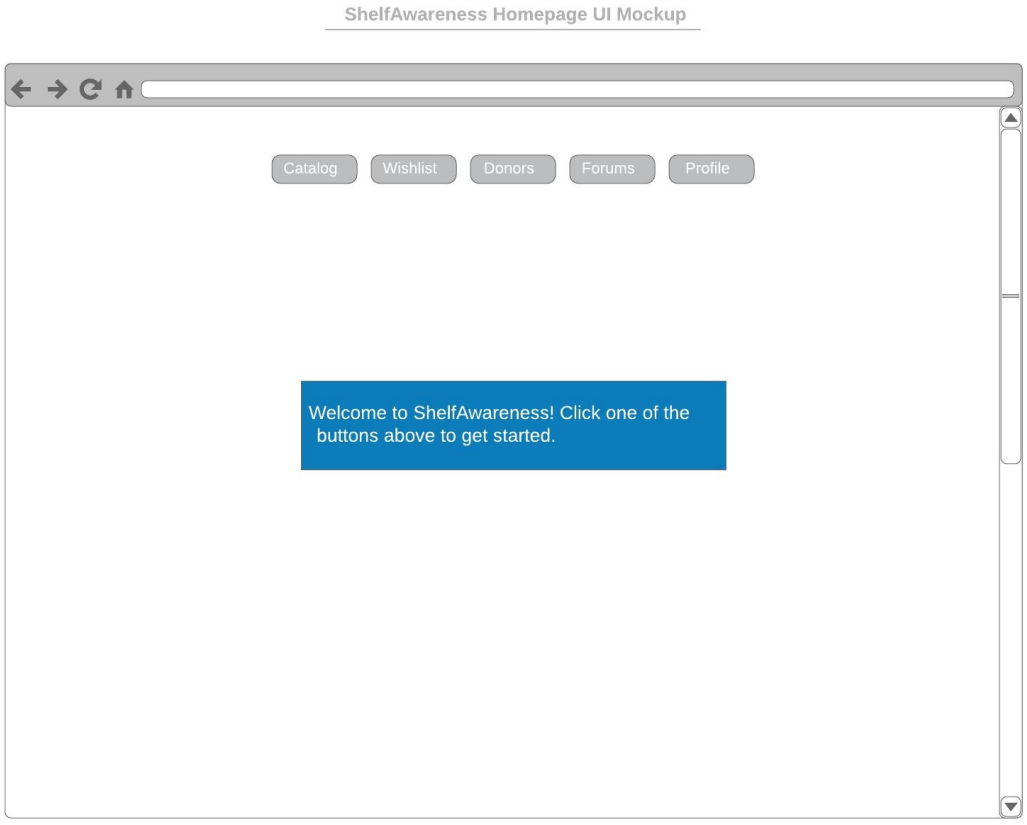

ShelfAwareness Donor
System Sequence Diagram
Actor
Home Screen
Donor Interface
Available Wish Interface
Donor Class
1: Log-in successful
2: Select Donor button
3: Select wish to fulfill
4: Donate Book
5: Save and send successful message
6: Navigate back to home screen
7: Log out

Forum

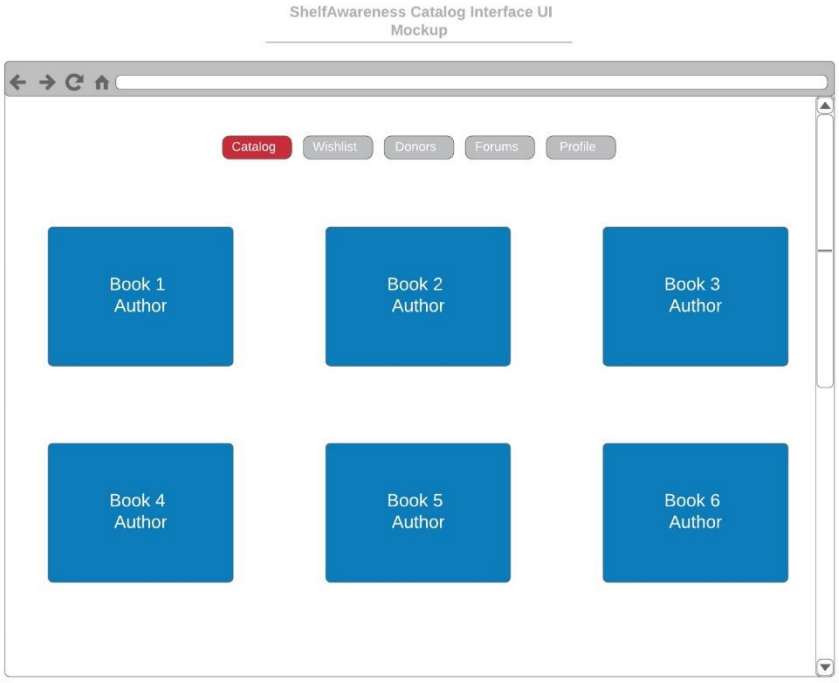

ShelfAwareness Forums
System Sequence Diagram
Actor
Home Screen
Forum Interface
Thread Interface
Forum Class
1: Log-in successful
2: Select Forum button
3: Select Thread
4: Create new thread
5: Comment on thread
6: Save and send
successful message
7: Navigate back to
home screen
8: Log out

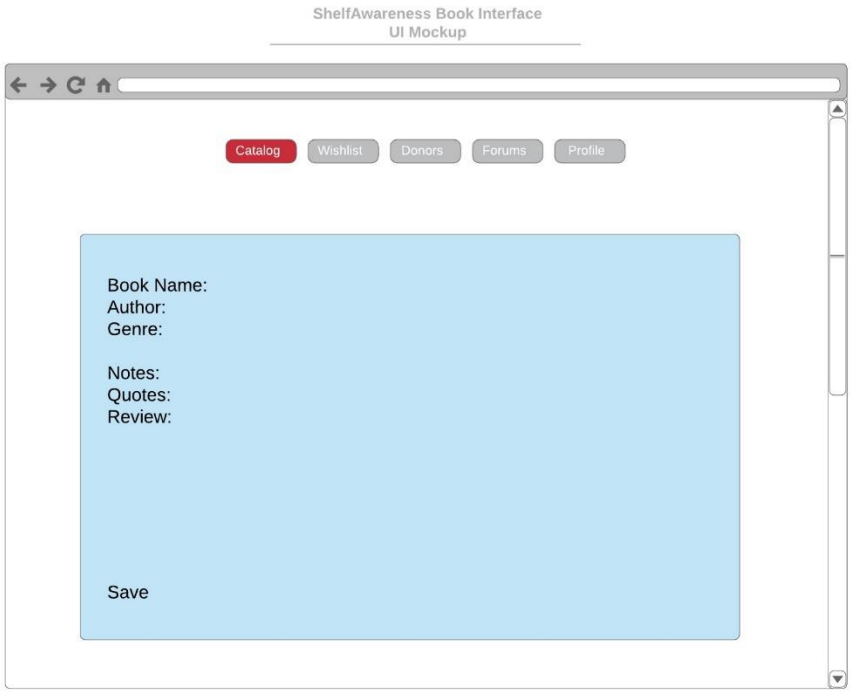
User Interfaces

Home Page

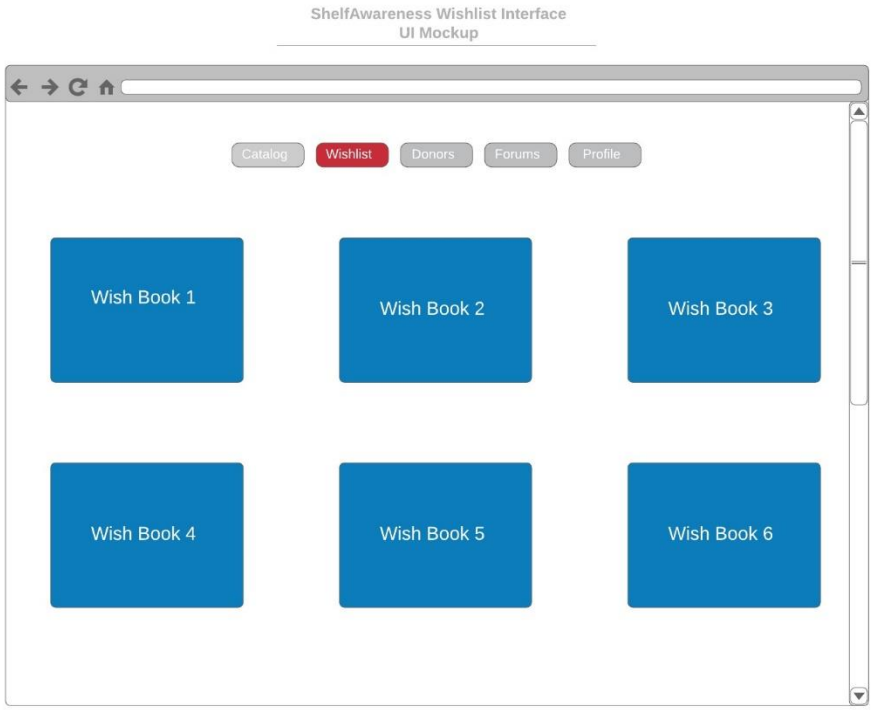


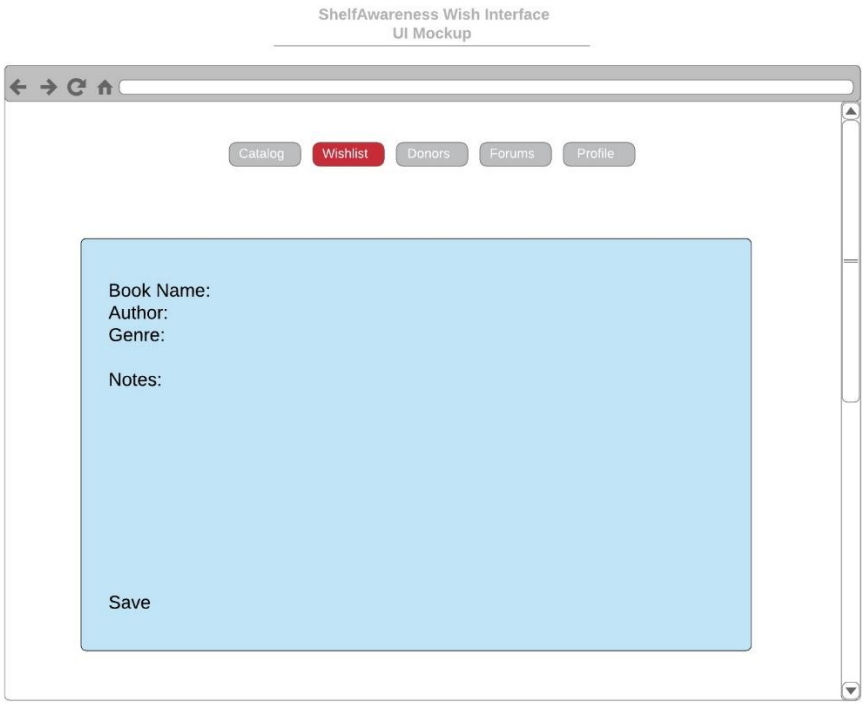
Catalog Pages



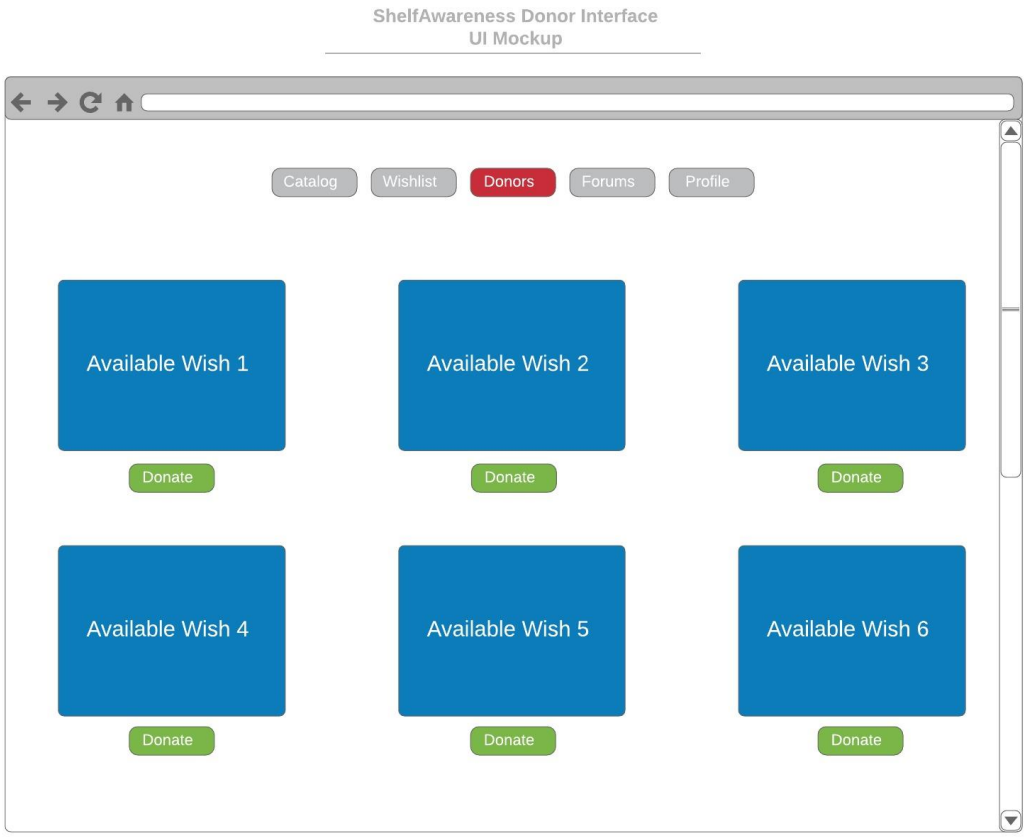


Wishlist Pages



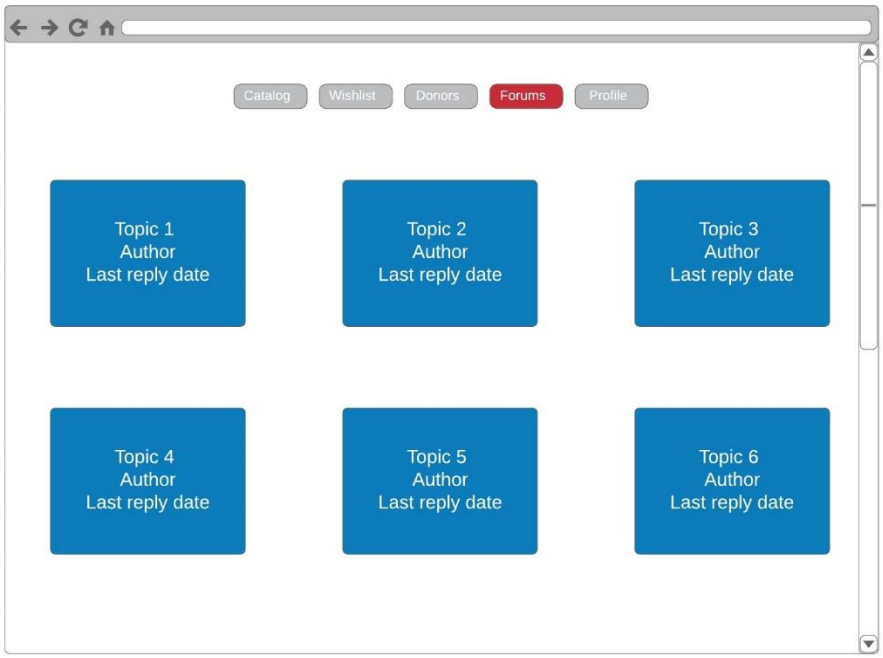


Donation Pages

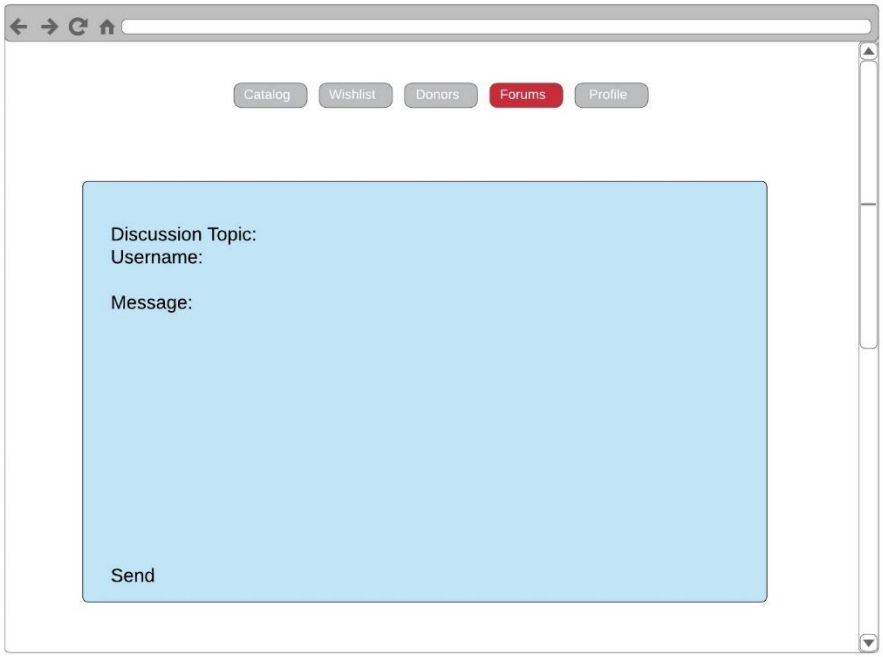


Forum Pages

ShelfAwareness Forum Interface UI Mockup



ShelfAwareness Thread Interface UI Mockup



UI Design Writeup

The Eight Golden Rules:

1. Strive for consistency

   ShelfAwareness is consistent in both design and aesthetic. On every single page of the site, the menu bar is in a fixed location. The functionality the user is using is highlighted red. The color scheme is also consistent throughout the site – boxes the user selects are in dark blue, while text field are contained within a light pastel blue area. All elements are labeled clearly with their purpose.

2. Enable frequent users to use shortcuts

   The standard web browser (back, forwards, refresh) and text editing (italics, bold, underline) shortcuts all apply. The simple and concise design of the site ensures that both new users and experts can use the application to its fullest potential.

3. Offer informative feedback

   Between the different main pages of the site, the menu bar button which the user is currently using is highlighted red, while the other buttons are in grayscale. This clearly denotes what the user is doing.

4. Design dialogs to yield closure

   Whenever uses must enter information into text fields, there is a popup dialog that will confirm that their action has been successfully saved. When a user donates a book, upon the recipient's confirmation of having gotten the book, the donor will also get a notification.

5. Offer simple error handling

   If users do not enter an existing or valid detail in their book catalogs or personal profiles, they will be prompted to try again. In addition, failure to provide correct login credentials will result in being prompted to try again for initiate account recovery.

6. Permit easy reversal of actions

   All information fields in all major functionalities can be changed at will. Forum posts and donations can be edited and cancelled as the user's desire.

7. Support internal locus of control

   The user is entirely in control of all of the information that goes into the application, as well as given control over how the information is organized. There are no background processes that change things without being prompted by the user.

8. Reduce short-term memory load

   All buttons are simple and do exactly what their labels imply. There are no hidden menus or buttons, and everything the application advertises is clearly visible no matter what part of the application the user is currently on.

Test Case Design

Unit Testing

In order to accomplish unit testing, I will use Test::Unit. This tool will be used because ShelfAwareness is a relatively simple tool, and does not need anything particularly advanced. In addition Test::Unit is automatically installed with Ruby and is convenient to use.

Integration Testing

Because there is a relatively limited number of functions that ShelfAwareness performs, a **bottom up** approach would be most appropriate to use. I will not be using a tool for integration testing because there are few enough functions in ShelfAwareness that it can be tested manually.

System Testing

System testing will also be done manually. Once Integration testing has finished, the modules do not have much interaction at all between each other, so minimal System testing should be needed to determine functionality.

Functionality Tested	**Inputs**	**Expected Output**	Actual Output
Navigate to Catalog page	User clicks labelled button in menu bar	User is directed to Catalog page	
Add new book	User clicks “new book” and then saves new entry under title “The Great Gatsby”	A new entry titled “The Great Gatsby” is in user catalog	
Edit details of book	User edits post to add that the author is F. Scott. Fitzgerald and saves/exits	In details for entry, author F. Scott Fitzgerald should be listed	
Delete book	User selects entry once again and deletes it	Entry is deleted from catalog	
Navigate to Wishlist page	User clicks labelled button in menu bar	User is directed to Wishlist page	
Create new wish	User clicks “new book” and then saves new entry under title “Frankenstein” by Mary Shelly, before saving and exiting.	Frankenstein by Mary Shelly is now in wishlist.	
Edit wish	User selects wish again to add a note saying “test” before saving and exiting.	User can now view note in wishlist entry containing “test”	
Delete wish	User selects entry and deletes it.	Wish is deleted from wishlist	
Navigate to Donor page	User clicks labelled button in menu bar	User is directed to Donor page and can view all wishes	
Donate books	User clicks button labelled “donate” under any entry on page	User successfully donates book	
Navigate to Forum page	User clicks labelled button in menu bar	User is directed to forums page	
User creates new thread	User creates a new forum post titled “Test”, saving and submitting it.	User can now view post titled “Test” and respond to it	
User comments on post	User selects same post, adding another comment to it and saving and submitting it	User can now view second message in thread.	

Licensing

GNU General Purpose License

Users are able to freely modify and redistribute software and its source code as desired, so long as the license is attached. There is no warranty on the software. Patents cannot be used to render the program non-free. There also more specific situations in which the free usage of the software may be revoked.

MIT License

There is no limitation at all on rights to use, copy, modify, merge, publish, distribute, sublicense, or sell, so long as the software has this notification attached to it.

Apache License

Similar to the GNU, except there is more protection from patents possibly hampering the free usage of the software.

ShelfAwareness

ShelfAwareness will utilize the MIT license. There is no new or revolutionary research going into the algorithms behind the application, and the purpose of the project is to help those in need, not profit off of the software itself.

Open-source Maintenance and Communication

Rules

1. Contributors will utilize GitHub to contribute to ShelfAwareness
2. All changes must be done in a separate branch.
3. All changes/contributions must be thoroughly and clearly documented
4. All versions of software must go through rigorous Unit, Integration, and Systems testing.
5. Software may then be submitted through a pull request and reviewed by moderators/administrators of the site.
6. A page of the ShelfAwareness forums may be dedicated to contributors.
7. All contributors must follow the same code of conduct that is expected of the users on the site.
    a. Be polite
    b. Offer constructive criticism, or none at all
    c. Prioritize helping others and the betterment of the product
8. Failure to comply by these rules will result in bans