EECS 2311 SecZ Winter 2024 Team 3 Take Home Assignment

Hashir Jamil
Mher Eric Gyulumyan
Ali Sina
Ahan Bhargava
Oscar Ye

Fully Implemented User Stories/Features

There are currently 20 fully implemented user stories. Each developer is responsible for reviewing 4 stories.

User Story/Feature	Associated Big User Story	Iteration	Main Developer(s)	Reviewer
Users can register for LongBox	User Login & Registration	1	Ahan Bhargava	Hashir Jamil
Users can login to Longbox	User Login & Registration	1	Ahan Bhargava	Hashir Jamil
Once users log in, users remain logged in and maintain this state.	User Login & Registration	1	Ahan Bhargava	Hashir Jamil
Once logged in, users can choose to logout at any time	User Login & Registration	1	Ahan Bhargava	Hashir Jamil
Users Can Login Using Email Address and Password	User Login and Registration	2	Mher Eric Gyulumyan	Ali Sina
Users Have Access To A Menu Of Links To All Aspects of the Home Dashboard/Profile	User Profile & Home Dashboard	1	Hashir Jamil	Mher Eric Gyulumyan
Users Can View Their Personal Profile Information	User Profile & Home Dashboard	1	Hashir Jamil	Mher Eric Gyulumyan
User can edit and save “about me” text box in profile page	User Profile & Home Dashboard	2	Oscar Ye	Ahan Bhargava
Users Can	Users Can	1	Mher Eric Gyulumyan &	Oscar Ye

Browse The Full Catalog of Comic Books In The System's Repository	Browse & Contribute to the Comic Book Catalog		Hashir Jamil	
Users Can Search For Individual Comic Books By Series Title	Users Can Browse & Contribute to the Comic Book Catalog	1	Hashir Jamil	Oscar Ye
Users Can Add A Comic Book To The System Repository	Users Can Browse & Contribute to the Comic Book Catalog	1	Ali Sina & Hashir Jamil	Ahan Bhargava
Users Can Sort A Comic Table Based on Every Attribute that a Comic Has	Users Can Browse & Contribute to the Comic Book Catalog	2	Mher Eric Gyulumyan	Ali Sina
User can Search for Comic by Title, Genre, Publisher, and Year	Users Can Browse & Contribute to the Comic Book Catalog	2	Mher Eric Gyulumyan	Ali Sina
User Can Navigate on a Comic Book Page by Clicking its Title if it Exists in a Results Table and Comic Collection	Users Can Browse & Contribute to the Comic Book Catalog	2	Mher Eric Gyulumyan	Ali Sina
Users Can Write Comments On Individual Comic Books	Users Can Read and Write Comments About Individual Comic Books	2	Ahan Bhargava	Mher Eric Gyulumyan
Users can View All Comments On Individual Comic Books	Users Can Read and Write Comments About Individual Comic Books	2	Ahan Bhargava	Mher Eric Gyulumyan

Users Can Add/Remove Comic Books To/From A List Of Finished Comic Books	Associated Big User Story: Users Can Build a Personal Comic Book Collection	2	Hashir Jamil	Ahan Bhargava
Users Can Add/Remove Comic Books To/From A List Of Reading In Progress Comic Books	Associated Big User Story: Users Can Build a Personal Comic Book Collection	2	Hashir Jamil	Ahan Bhargava
When adding a comic book to the repository a user can directly favorite it at the same time	Associated Big User Story: Users Can Build a Personal Comic Book Collection	2	Ali Sina	Oscar Ye
Users can remove a comic book from favorites	Associated Big User Story: Users Can Build a Personal Comic Book Collection	2	Ali Sina	Oscar Ye

End-2-End Tests By Developer/Reviewer

Ahan Bhargava

User can edit and save “about me” text box in profile page

E2E Test 1: Initial state verification

This test case ensures the current state of the feature corresponds with the user story’s description

1) Login into the app using valid credentials.
2) Navigate to the profile page where the about me text box should be there.
3) About me text box is present with two buttons ‘Cancel’ and ‘Edit’.
4) The text inside the box cannot be editable until and unless the ‘Edit’ button is clicked.
5) The ‘Cancel’ button is disabled, only the ‘Edit’ button is active.

E2E Test 2: Editing the about me

This test case ensures the about me textbox is editable upon clicking of the button, and the save option works

1) Login into the app using valid credentials.
2) Navigate to the profile page where the about me text box should be there.
3) The ‘Edit’ button is clicked.
4) The text box becomes editable.
5) The ‘Edit’ button changes to the ‘Save’ button.
6) After editing the text, clicking the ‘Save’ button, updates the text in the box, and the buttons are back to initial position, that is ‘Cancel’ and ‘Edit’ of which the ‘Cancel’ button is disabled.
7) The database is updated.

E2E Test 3: Editing the about me again after saving it

This test case ensures that the about me can be edited back to back

1) Login into the app using valid credentials.
2) Navigate to the profile page where the about me text box should be there.
3) The ‘Edit’ button is clicked.
4) The text box becomes editable.
5) The ‘Edit’ button changes to the ‘Save’ button.

6) After editing the text, clicking the ‘Save’ button, updates the text in the box, and the buttons are back to initial position, that is ‘Cancel’ and ‘Edit’ of which the ‘Cancel’ button is disabled.
7) The database is updated.
8) Again the ‘Edit’ button is clicked.
9) The text box is now editable.
10) The buttons change their state in the same way as mentioned above.
11) The text is saved, buttons back to default.
12) Database is updated.

E2E Test 4: Cancel button on about me works

This test case ensures that the cancel button under the about me works

1) Login into the app using valid credentials.
2) Navigate to the profile page where the about me text box should be there.
3) The ‘Edit’ button is clicked.
4) Changes are made to the text box.
5) ‘Cancel’ button is clicked.
6) Text resets back to what it was before editing.
7) Text box is no longer editable.
8) Database is not updated as no changes have been made.

E2E Test 5: Change to another page in middle of editing

This test case tests what happens in case a page is changed in the middle of editing.

1) Login into the app using valid credentials.
2) Navigate to the profile page where the about me text box should be there.
3) The ‘Edit’ button is clicked.
4) Changes are made to the text box.
5) The page is changed from profile page to any other page.
6) The changes are not saved, and the box is in edit mode.

E2E Test 6: Adding a big text in the text box

This test case ensures that a big text can be added and read.

1) Login into the app using valid credentials.
2) Navigate to the profile page where the about me text box should be there.
3) The ‘Edit’ button is clicked.
4) A big text is added to the text box.
5) Text is cut off after a certain length.
6) The database is updated correctly with the big paragraph.

Users Can Add A Comic Book To The System Repository

E2E Test 1: Adding all information with appropriate fields and not favorating it

This test case ensures that adding a comic book works if supplied all valid information and not favorating the book.

1) Login into the app using valid credentials.
2) Navigate to the ‘Add Comic Page’
3) Fill all fields with appropriate data (text in text fields and numbers in number fields).
4) Not selecting the is favorite check box.
5) Clicking the ‘Enter Comic Book’ button.
6) Popup showing successful entry of book.
7) Database updated with the details added.
8) Comic not visible in the favorites tab.
9) Added comic visible in the comic repository.

E2E Test 2: Adding all information with appropriate fields and favorating it

This test case ensures that adding a comic book works if supplied all valid information and favorating the book.

1) Login into the app using valid credentials.
2) Navigate to the ‘Add Comic Page’
3) Fill all fields with appropriate data (text in text fields and numbers in number fields).
4) Selecting the is favorite checkbox.
5) Clicking the ‘Enter Comic Book’ button.
6) Popup showing successful entry of book.
7) Database updated with the details added.
8) Comic visible in the favorites tab.
9) Added comic visible in the comic repository.

E2E Test 3: Adding some information and skipping fields (leaving some text fields empty)

This test case ensures that adding a comic book works if some text fields are empty.

1) Login into the app using valid credentials.
2) Navigate to the ‘Add Comic Page’
3) Fill some fields (skipped fields which required text).
4) Not selecting the is favorite checkbox.
5) Clicking the ‘Enter Comic Book’ button.
6) Popup showing successful entry of book.
7) Database updated with the details added, fields that were not entered are empty.
8) Comic not visible in the favorites tab.
9) Added comic visible in the comic repository.

E2E Test 3: Adding some information and skipping fields (leaving number of issues and/or year published field empty)

This test case ensures that adding a comic book works if the fields that require integers can be left empty.

1) Login into the app using valid credentials.
2) Navigate to the ‘Add Comic Page’
3) Fill some fields (skipped number of issues and/or year published field).
4) Not selecting the is favorite checkbox.
5) Clicking the ‘Enter Comic Book’ button.
6) Error thrown in the console of compiler
7) Database not updated.
8) Comic not visible in the favorites tab.
9) Comic not visible in the comic repository.

E2E Test 4: Adding inappropriate data types in fields (number of issues and/or year published field supplied with text)

This test case ensures that adding a comic book works if the fields that require integers can be supplied with text.

1) Login into the app using valid credentials.
2) Navigate to the ‘Add Comic Page’
3) In the year published/ number of issues enter text.
4) Not selecting the is favorite checkbox.
5) Clicking the ‘Enter Comic Book’ button.
6) Error thrown in the console of compiler.
7) Database not updated.
8) Comic not visible in the favorites tab.
9) Comic not visible in the comic repository.

E2E Test 5: All fields empty

This test case ensures that a comic book can be added with all fields empty

1) Login into the app using valid credentials.
2) Navigate to the ‘Add Comic Page’
3) Not filling any of the fields.
4) Not selecting the is favorite checkbox.
5) Clicking the ‘Enter Comic Book’ button.
6) Error thrown in the console of compiler.
7) Database not updated.
8) Comic not visible in the favorites tab.
9) Comic not visible in the comic repository.

Users Can Add/Remove Comic Books To/From A List Of Finished Comic Books

E2E Test 1: Adding and removing a book to the finished list via opening the Comic Info Panel using the comic repository and seeing if it gets updated

This test case ensures that a user can open the comic info page from the comic repository and can use the add to finished button

1) Login into the app using valid credentials.
2) Navigate to the ‘Comic Repository’ page’
3) Open a comic by double clicking on its name.
4) Comic info panel opens up, there are three active buttons there, ‘Add to Favorites’, ‘Add to Finished’ and ‘Add to Reading’. Buttons ‘Remove Favorite’, ‘Remove from Finished’ are disabled.
5) Clicked the ‘Add to Finished’ button. The ‘Add to Finished’ button becomes disabled and ‘Remove from Favorites’ button becomes active.
6) Navigate to the profile page in mainframe, ‘Comics Finished’ field updated, count increased by one, and the ‘Comics Previously Finished’ table is also updated with the same information.
7) Database updated with the book added to finished.
8) Open the same comic book again, click on the ‘Remove from Finished’ button.
9) Navigate to the profile page in mainframe, ‘Comics Finished’ field updated, count decreased by one, and the comic is removed from the ‘Comics Previously Finished’ table.
10) Database updated with the book removed from finished.

E2E Test 2: Adding and removing a book to the finished list via opening the Comic Info Panel using the search and seeing if it gets updated

This test case ensures that a user can open the comic info page from the search and can use the add to finished button

1) Login into the app using valid credentials.
2) Navigate to the ‘Comic Repository’ page’
3) Search a comic by its title and open the comic by double clicking on its name.
4) Comic info panel opens up, there are three active buttons there, ‘Add to Favorites’, ‘Add to Finished’ and ‘Add to Reading’. Buttons ‘Remove Favorite’, ‘Remove from Finished’ are disabled.
5) Clicked the ‘Add to Finished’ button. The ‘Add to Finished’ button becomes disabled and ‘Remove from Favorites’ button becomes active.
6) Navigate to the profile page in mainframe, ‘Comics Finished’ field updated, count increased by one, and the ‘Comics Previously Finished’ table is also updated with the same information.
7) Database updated with the book added to finished.

8) Open the same comic book again, click on the ‘Remove from Finished’ button.
9) Navigate to the profile page in mainframe, ‘Comics Finished’ field updated, count decreased by one, and the comic is removed from the ‘Comics Previously Finished’ table.
10) Database updated with the book removed from finished.

Users Can Add/Remove Comic Books To/From A List Of Reading Comic Books

E2E Test 1: Adding and removing a book to the reading list via opening the Comic Info Panel using the comic repository and seeing if it gets updated

This test case ensures that a user can open the comic info page from the comic repository and can use the add to reading button

1) Login into the app using valid credentials.
2) Navigate to the ‘Comic Repository’ page.
3) Open a comic by double clicking on its name.
4) Comic info page opens up, there are three active buttons there, ‘Add to Favorites’, ‘Add to Finished’ and ‘Add to Reading’. Buttons ‘Remove Favorite’, ‘Remove from Finished’ are disabled.
5) Clicked the ‘Add to Reading’ button. The ‘Add to Reading’ button becomes disabled and ‘Remove from Reading’ button becomes active.
6) Navigate to the profile page in mainframe, ‘Comics Reading’ field updated, count increased by one, and the ‘Comics Currently Reading’ table is also updated with the same information.
7) Database updated with the book added to reading.
8) Open the same comic book again, click on the ‘Remove from Reading’ button.
9) Navigate to the profile page in mainframe, ‘Comics Reading’ field updated, count decreased by one, and the comic is removed from the ‘Comics Currently Reading’ table.
10) Database updated with the book removed from reading.

E2E Test 2: Adding and removing a book to the finished list via opening the Comic Info Panel using the search and seeing if it gets updated

This test case ensures that a user can open the comic info panel from the search and can use the add to reading button

1) Login into the app using valid credentials.
2) Navigate to the ‘Comic Repository’ page.
3) Search a comic by its title and open the comic by double clicking on its name.
4) Comic info page opens up, there are three active buttons there, ‘Add to Favorites’, ‘Add to Finished’ and ‘Add to Reading’. Buttons ‘Remove Favorite’, ‘Remove from Finished’ are disabled.

5) Clicked the ‘Add to Reading’ button. The ‘Add to Reading’ button becomes disabled and ‘Remove from Reading’ button becomes active.
6) Navigate to the profile page in mainframe, ‘Comics Reading’ field updated, count increased by one, and the ‘Comics Currently Reading’ table is also updated with the same information.
7) Database updated with the book added to reading.
8) Open the same comic book again, click on the ‘Remove from Reading’ button.
9) Navigate to the profile page in mainframe, ‘Comics Reading’ field updated, count decreased by one, and the comic is removed from the ‘Comics Currently Reading’ table.
10) Database updated with the book removed from reading.

Ali Sina

Users Can Login Using Email Address and Password

E2E Test 1: User Can Login with Valid Email and Valid Password

1. User opens the application and enters the login/entry page.
2. User inputs their email address and password into the designated fields.
3. They click the "Sign in!" button.
4. The system verifies the entered email address and password against the stored credentials in the database.
5. The user logs in

E2E Test 2: User Cannot Login with Invalid Email

1. User opens the application and enters the login/entry page.
2. User inputs their email address that has not previously registered with system
3. They click the "Sign in!" button.
4. The system couldn’t verify the entered email address against the stored credentials in the database.
5. The user gets an message which is covered behind the email text field and cannot login

E2E Test 3: User Cannot Login with Incorrect Password

1. User opens the application and navigates to the login/entry page.
2. User inputs their registered email address and an incorrect password into the designated fields.
3. They click the "Sign in!" button.
4. The system verifies the entered email address but fails to authenticate the password.
5. The user receives an error message which is covered behind the email text field indicating the incorrect password and is unable to login.

Users Can Sort A Comic Table Based on Every Attribute that a Comic Has

E2E Test 1: User Sorts Comic Table by Series Title

1. User opens the application and enters the login/entry page.
2. User inputs their email address and password into the designated fields.
3. They click the "Sign in!" button.
4. The system verifies the entered email address and password against the stored credentials in the database.
5. The user logs in
6. User lands to the Comic Repository within the application.
7. User locates the column header for "Series Title" in the comic table.
8. User clicks on the header to sort the table in ascending order based on the series title.
9. User verifies that the table is sorted alphabetically by series title from A to Z.

10. User clicks on the header again to sort the table in descending order.
11. User verifies that the table is sorted alphabetically by series title from Z to A.

E2E Test 2: User Sorts Comic Table by Author

1. User opens the application and enters the login/entry page.
2. User inputs their email address and password into the designated fields.
3. They click the "Sign in!" button.
4. The system verifies the entered email address and password against the stored credentials in the database.
5. The user logs in
6. User lands to the Comic Repository within the application.
7. User locates the column header for "Author" in the comic table.
8. User clicks on the header to sort the table in ascending order based on the author's name.
9. User verifies that the table is sorted alphabetically by author's name from A to Z.
10. User clicks on the header again to sort the table in descending order.
11. User verifies that the table is sorted alphabetically by author's name from Z to A.

E2E Test 3: User Sorts Comic Table by Artist

1. User opens the application and enters the login/entry page.
2. User inputs their email address and password into the designated fields.
3. They click the "Sign in!" button.
4. The system verifies the entered email address and password against the stored credentials in the database.
5. The user logs in
6. User lands to the Comic Repository within the application.
7. User locates the column header for "Artist" in the comic table.
8. User clicks on the header to sort the table in ascending order based on the artist's name.
9. User verifies that the table is sorted alphabetically by the artist's name from A to Z.
10. User clicks on the header again to sort the table in descending order.
11. User verifies that the table is sorted alphabetically by the artist's name from Z to A.

E2E Test 4: User Sorts Comic Table by Number of Issues

1. User opens the application and enters the login/entry page.
2. User inputs their email address and password into the designated fields.
3. They click the "Sign in!" button.
4. The system verifies the entered email address and password against the stored credentials in the database.
5. The user logs in
6. User lands to the Comic Repository within the application.
7. User locates the column header for "Number of Issues" in the comic table.
8. User clicks on the header to sort the table in ascending order based on the number of issues.

9. User verifies that the table is sorted numerically by the number of issues from lowest to highest.
12. User clicks on the header again to sort the table in descending order.
13. User verifies that the table is sorted numerically by the number of issues from highest to lowest.

E2E Test 4: User Sorts Comic Table by Publisher

1. User opens the application and enters the login/entry page.
2. User inputs their email address and password into the designated fields.
3. They click the "Sign in!" button.
4. The system verifies the entered email address and password against the stored credentials in the database.
5. The user logs in
6. User lands to the Comic Repository within the application.
7. User locates the column header for "Publisher" in the comic table.
8. User clicks on the header to sort the table in ascending order based on the publisher's name.
9. User verifies that the table is sorted alphabetically by publisher's name from A to Z.
10. User clicks on the header again to sort the table in descending order.
11. User verifies that the table is sorted alphabetically by publisher's name from Z to A.

E2E Test 5: User Sorts Comic Table by Year Published

1. User opens the application and enters the login/entry page.
2. User inputs their email address and password into the designated fields.
3. They click the "Sign in!" button.
4. The system verifies the entered email address and password against the stored credentials in the database.
5. The user logs in
6. User lands to the Comic Repository within the application.
7. User locates the column header for "Year Published" in the comic table.
8. User clicks on the header to sort the table in ascending order based on the year of publication.
9. User verifies that the table is sorted chronologically by the year of publication from oldest to newest.
10. User clicks on the header again to sort the table in descending order.
11. User verifies that the table is sorted chronologically by the year of publication from newest to oldest.

User Can Navigate on a Comic Book Page by Clicking its Title if it Exists in a Results Table and Comic Collection

E2E Test 1: User Navigates to Comic Book Page from Results Table

1. Login into the app using valid credentials.

2. Navigate to the ‘Comic Repository’ page.
3. User performs a search or filter operation on the application to generate a results table containing comic book entries.
4. User locates a specific comic book title within the results table.
5. User double clicks on the title of the comic book.
6. The system navigates the user to the individual comic book page associated with the clicked title.
7. User verifies that they are directed to the correct comic book page, displaying detailed information about the selected comic book.
8. User interacts with various elements on the comic book page to ensure functionality and accuracy.

E2E Test 2: User Navigates to Comic Book Page from Comic Collection

1. Login into the app using valid credentials.
2. Navigate to the ‘Comic Repository’ page.
3. User accesses their comic collection within the application, containing a list of saved or favorite comic books.
4. User locates a specific comic book title within their collection.
5. User clicks on the title of the comic book.
6. The system navigates the user to the individual comic book page associated with the clicked title.
7. User verifies that they are directed to the correct comic book page, displaying detailed information about the selected comic book.
8. User interacts with various elements on the comic book page to ensure functionality and accuracy.

User can Search for Comic by Title, Genre, Publisher, and Year

E2E Test 1: User Searches for Comic by Title

1. Login into the app using valid credentials.
2. Navigate to the ‘Comic Repository’ page.
3. User locates the search bar on the page.
4. User enters the title of a specific comic they want to search for.
5. User opens the area of search dropdown and selects Title
6. User ensures that the search query is correctly entered.
7. User submits the search query by pressing the Enter key.
8. The system retrieves and displays relevant comic entries matching the provided title in a new panel.
9. User verifies that the search results include the comic they were looking for.
10. User double clicks on the desired comic to view its details and ensure accuracy.

E2E Test 2: User Searches for Comic by Genre

1. Login into the app using valid credentials.

2. Navigate to the ‘Comic Repository’ page.
3. User locates the search bar on the page.
4. User enters the name of a specific genre they want to search for.
5. User opens the area of the search dropdown and selects "Genre."
6. User ensures that the search query is correctly entered.
7. User submits the search query by pressing the Enter key.
8. The system retrieves and displays relevant comic entries belonging to the specified genre in a new panel.
9. User verifies that the search results include comics matching the searched genre.
10. User double clicks on a comic within the search results to view its details and confirm genre accuracy.

E2E Test 2: User Searches for Comic by Publisher

1. Login into the app using valid credentials.
2. Navigate to the ‘Comic Repository’ page.
3. User locates the search bar on the page.
4. User enters the name of a specific publisher they want to search for.
5. User opens the area of the search dropdown and selects "Publisher."
6. User ensures that the search query is correctly entered.
7. User submits the search query by pressing the Enter key.
8. The system retrieves and displays relevant comic entries published by the specified publisher in a new panel.
9. User verifies that the search results include comics published by the searched publisher.
10. User double clicks on a comic within the search results to view its details and confirm publisher accuracy.

E2E Test 3: User Searches for Comic by Year

1. Login into the app using valid credentials.
2. Navigate to the ‘Comic Repository’ page.
3. User locates the search bar on the page.
4. User enters a specific year they want to search for.
5. User opens the area of the search dropdown and selects "Year."
6. User ensures that the search query is correctly entered.
7. User submits the search query by pressing the Enter key.
8. The system retrieves and displays relevant comic entries published in the specified year in a new panel.
9. User verifies that the search results include comics published in the searched year.
10. User double clicks on a comic within the search results to view its details and confirm year accuracy.

E2E Test 4: User Searches for Comic by Combination of Criteria

1. Login into the app using valid credentials.
2. Navigate to the ‘Comic Repository’ page.

3. User locates the search bar on the page.
4. User enters a combination of title, genre, publisher, and year they want to search for.
5. User opens the area of the search dropdown and selects one of the "Title, Genre, Publisher, and Year."
6. User ensures that the search query is correctly entered.
7. User submits the search query by pressing the Enter key.
8. The system retrieves and displays an empty table with zero match found.

Hashir Jamil

Users can register for LongBox

E2E Test 1: An account can be registered as per the prompted fields

This test case measures the main scenario and expects the user to provide sanitized input data for the form that is presented. The steps and expected outputs are as follows:

6. User opens the application and enters the login/entry page.
7. User selects the “Sign Up!” button.
8. The user lands on the registration form with an unclickable register button.
9. User provides all registration information (username, password, first name, last name, date of birth, email, password & country). The username must not contain ‘@’ character and both the username plus the email must be unique, i.e., not in the database associated with another account. The password provided must be “strong” as defined by the prompt.
10. The red message saying “Please enter a valid user name, a valid email and a valid password!” becomes hidden.
11. The user selects the agreement radio button for terms and conditions.
12. The “Sign Up for LongBox!” button is now clickable.
13. The “Sign Up for LongBox!” button is clicked.
14. User is redirected to the login/entry page.
15. The user should now be able to login using the username (or email) and password combination provided in step 3.
16. The user can login and view the landing page.
17. The database is now confirmed to be updated for the user that just signed up with a record displaying all information found in step 3 as well with a unique primary key id.

E2E Test 2: An account cannot be registered if fields are missing.

This test case measures an exception scenario when fields are missing or not provided correctly. The steps and expected outputs are listed below:

1. User opens the application and enters the login/entry page.
2. User selects the “Sign Up!” button.
3. The user lands on the registration form with an unclickable register button.
4. User does does not provide all the information in the form (either provides a subset of some fields or no fields, either case is treated as the same).
5. The red message saying “Please enter a valid user name, a valid email and a valid password!” stays visible if all three of the corresponding fields are empty or incorrectly filled.
6. If any proper subset of the set of all three of the fields in 5 are filled correctly then the red message adjusts to exclude the fields that were written correctly.
7. The “Sign Up for LongBox!” button cannot be clicked and no account cannot be registered with an incomplete form.

E2E Test 3: An account cannot be registered without a compliant password and/or username and/or email

This test case measures the ability to prevent accounts to be made if the provided password/email/username does not conform to the provided regulation under the respective fields in the registration form. The steps are below:

1. User opens the application and enters the login/entry page.
2. User selects the “Sign Up!” button.
3. The user enters any subset of or all of the fields password, username, or email in an incorrect format.
4. The red message saying “Please enter a valid user name, a valid email and a valid password!” adjusts accordingly based on what is correct/incorrect
5. The “Sign Up for LongBox!” button remains unclickable.

E2E Test 4: An account cannot be made if the username and/or email are already in the system.

This test case measures the ability to exclude accounts that provide correct form information as per the contract of all inputs but the user provides an already used email or username (or both already used). The steps are below:

1. User opens the application and enters the login/entry page.
2. User selects the “Sign Up!” button.
3. The user enters all fields according to the instructions/contract except for:
    a. Case A: enters previously used email but new/unique username
    b. Case B: enters previously used username but new/unique email
    c. Case C: enters previously used email and previously used username
4. The “Sign Up for LongBox!” button becomes clickable, red message disappears and the user clicks the button.
5. The user is denied registration and told that this username and/or email is already taken.
6. The user is prompted to refill the form with a new username or email.

Users can login to Longbox

E2E Test 1: The user can login when using a previously registered account

This test handles users ability to login correctly by entering the correct username/email and password combination of a previously registered account. The steps are below:

1. User opens the application and enters the login/entry page.
2. The user enters the email or username for an existing account as well as the password for that account
3. The user presses the “Sign in!” button
4. The user lands on the trending page of the main home window of LongBox with their username displaying in the top right corner.

E2E Test 2: Entering the wrong username/email or email displays error.

This test is to confirm that username/email not associated to any account cannot be used to login to LongBox. The steps to do this are below:

1. User opens the application and enters the login/entry page.
2. The user enters an email or username that does not exist in the database.
3. The user writes anything in the password field (one that is perceived to be right or one that is known to be wrong).
4. The user presses the “Sign in!” button.
5. The “User does not exist” message is displayed and nothing else changes. The user can attempt to login again.

E2E Test 3: Entering the wrong password but correct username/email displays error.

This test is to show that a username/email being input correctly but a password being input incorrectly gives a special error. This error does not show up if the username/email given does not exist in the system. The steps to do this are below:

1. User opens the application and enters the login/entry page.
2. User enters an email or username that is associated with an account in the database.
3. User enters the incorrect password for the account.
4. The user presses the “Sign in!” button.
5. The user is displayed an error that says in red: “Password Incorrect.” Nothing else changes and the user can attempt to login again.

E2E Test 4: The user can login to the same account with either the email or username associated to that account.

This test is to confirm that the same account is accessed after authentication with either the email address or username associated with that account in the database of the system. The steps to do this are below:

1. User opens the application and enters the login/entry page.
2. The user enters the username for an account that is already registered in the system.
3. The user enters the correct password for this account.
4. The user presses the “Sign in!” button
5. The user lands on the trending page of the main home window of LongBox with their username displaying in the top right corner and collection items/profile information in favorites/profile pages.
6. The user views the corresponding username in the top right corner of the main window and takes note of it.
7. The user presses the “Logout” button.
8. The user lands back on the entry page for login/registration.
9. The user enters the email for the account given in step 2 into the username field.
10. The user enters the same password as step 3.
11. The user presses the “Sign in!” button.

12. The user lands on the trending page of the main home window of LongBox with the same username as step 5 displaying in the top right corner as well as the same collection items and profile information in favorites/profile pages.

Once users log in, users remain logged in and maintain this state.

E2E Test 1: When logged in click every single button in every single page at least once.

The purpose of this test is to try out all features, except logout, at least once (essentially do a depth first search of all the pages in the application and click every button) to see if the system unexpectedly logs out or loses track of the current logged in user by not displaying their username in the . Step are below:

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page and scroll around the lists of trending comics.
3. Go to the comic repository page.
    a. Use the refresh button.
    b. Search repo using all options in the dropdown search filter at least once.
    c. Double click on two or more books and add all opened books to favorites, reading and finished lists. For one of the books use the remove from favorites, reading and finished comics buttons.
    d. Close the individual comic pages and use the repository's add to favorites button to add 3 or more books to favorites.
4. Go to add comic page and add one comic book to the repository without selecting the is favorite radio button then add another one and select the is favorite radio button. Next add 3 comments to a comic and see if the username is associated with that comment. Close this comic page and open it again and write one more comment to ensure the username is associated with it.
5. Go to the favorite page to see if the favorites were populated correctly and use the refresh button. Try to search favorites by all the filters at least once. Try to remove favorites.
6. Go to the profile page to see if the reading lists and favorites lists got updated. Next edit the about me and save. Next edit it again but cancel. Edit it one last time and save it.
7. If by now everything was kept consistent and user name always displayed and all the lists updated as intended and comments and about me were not wrong then the session is safely confirmed to be maintained.

Once logged in, users can choose to logout at any time

E2E Test 1: Go to every page within the system and try to logout

This test is to ensure that the user can logout from every page and stop having their profile updated in any way. If at any time logout is not possible or user actions are still conductible even though logout has occurred, then the log out is not working as intended. Steps are below:

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page and try to logout. Log back in.
3. Navigate to the profile page and try to logout. Log back in. Go back to the profile page and update about me then save then update again but cancel. Try to logout.
4. Log back in and go to the add a comic page and add one comic without the is favorite radio button selected and add one with it selected. Try to logout.
5. Log back in and go to the favorites page. Check to see if the comic book added to favorites made it in. Try to use all search filter options, use the refresh button and try to unfavorite a comic book. Then try to logout.
6. Log back in and go to the favorites page. Check to see if the comic book was unfavorite. Log out.
7. Log back in and go to the comic repository. Use all search options and favorite and refresh button 2 or more times then try to logout.
8. Log back in and go to the comic repository. Double click on a comic book to open a new window for the comic book for 3 comic books. Add the comic books to all three lists then add a comment. Close all but two comic book pages and remove this from lists. Log out.
9. Without logging back in or closing the sign in/sign up window. Try to write a new comment and add or remove from lists on the still open comic book page.
10. Log back in and don't close the comic book page. Try to add or remove and add a comment to the comic book. Log out.

Mher Eric Gyulumyan

Users Have Access To A Menu Of Links To All Aspects of the Home Dashboard/Profile

E2E Test 1: After a successful login, a user shall be greeted by the recommended page along with multiple buttons in the top for each page of the application.

The purpose of this test is to ensure upon a proper login to the application, the user will by default land on the recommended page and have the dashboard at the top of the application loaded with all appropriate buttons to navigate across the application.

1. Login to the application and be presented with the ‘Recommended Comic Books’ page
2. Look at the dashboard above, verify that the first button (left to right) is titled ‘Comic Repository’
3. Verify the second button is titled ‘Add Comic’
4. Verify the third button is titled ‘Favorites’
5. Verify the fourth button it titled ‘Profile’
6. Verify that fifth button is titled ‘Trending’
7. Verify the sixth button is titled ‘Recommended’
8. Verify the seventh button is titled ‘Social’
9. Verify the eighth button, located all the way to the right, is titled ‘Log Out’
10. Verify that in between the buttons ‘Social’ and ‘Log Out’, the users name exists

E2E Test 2: After a successful login, a user can navigate to any of the tabs within the dashboard

The purpose of this test is to ensure that not only the menu dashboard has been rendered before, but that is interactable. Such that the user can click on any of the buttons and have their appropriate page displayed.

1. Login to the application, you will by default be on the ‘Recommended’ page
2. Click on the first button ‘Comic Repository’ and verify the title of the page changes to ‘Comic Repository’
3. Click on the second button ‘Add Comic’ and verify the title of the page changes to ‘Add a Comic Book To The LongBox Repository’
4. Click on the third button ‘Favourites’ and verify the title of the page changes to ‘User Favourites’
5. Click on the fourth button ‘Profile’ and verify the title of the page changes to ‘Profile View’
6. Click on the fifth button ‘Trending’ and verify the title of the page changes to ‘Trending’
7. Click on the sixth button ‘Recommended’ and verify the title of the page changes to ‘Recommended Comic Books’

8. Click on the seventh button ‘Social’ and verify the title of the page changes to ‘Nerd Hall of Fame’

E2E Test 3: When logged in, a user can log out with the ‘logout’ button at the top right

The purpose of this test is to ensure that the logout functionality within the dashboard is properly functionally.

1. Login to the application
2. On the top right of the application, verify that a button titled ‘Log Out’ is present
3. Click on this button and verify a small box appears with the message ‘Are you sure you want to log out?’
4. Click on the ‘Yes’ option
5. Verify that the user gets redirected to the login page, and the title ‘Welcome Back to LongBox’ is present

Users Can View Their Personal Profile Information

E2E Test 1: After logging in, a user can navigate to the profile section by clicking on the ‘Profile’ button at the top dashboard

The purpose of this test is to ensure that the user can navigate to the profile page from the dashboard from any of the other pages, and to verify that the profile page has loaded appropriately.

1. Login to the application
2. Verify that the fourth button on the top dashboard is titled ‘Profile’
3. Click on that button
4. Verify that the title of the page changes to ‘Profile View’
5. Verify that a divider appears, dividing the profile section into a left and right part
6. Verify on the right side, the About Me box appears on top, followed by the ‘Comics Currently Reading’ scroll table and ‘Comics Previously Finished’ scroll table underneath it
7. Verify on the left side, the Users username appears and matches the username that is to the left of the ‘Log Out’ button (located on the top right of the dashboard)

E2E Test 2: When at the profile page, a user can view all relevant information about themselves on the left side of the profile panel

The purpose of this test is to ensure that when a user navigates to their profile page, all the information within the left side of the panel is relevant and accurate to the specific user.

1. Login to the application
2. Navigate to the profile page by clicking on the button titled ‘Profile’
3. Verify that a divider appears, dividing the profile section into a left and right part
4. Verify on the left side, verify that the following sections are displayed

    a. Username, First Name, Last Name, Date of Birth, Email, Country, Joined On, Comics Reading, Comics FInished
5. Verify that for each of these sections, the users appropriate information is displayed
    a. For the user ‘123’, the following should be displayed for the respective sessions mentioned in 4a
        i. 123, Quick, Access, 2003-02-10 00:00:00.0, 123@emai.com, India, 2024-02-15 00:00:00.0, 0, 0

Users Can Write Comments On Individual Comic Books

E2E Test 1: Leave a comment on a comic book page

This test is to ensure that the basic functionality of commenting is functioning properly, with the user being able to add a comment exactly as written in the upper box and appearing appropriately in the bottom box.

1. Login to the application
2. Navigate to a comic book either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites
3. Verify if on the right side of the comic book page, two boxes appear, one with the title ‘Share your thoughts’ and the one under it ‘What others think about this comics’
4. In the top box, write down a comment for the comic page.
5. After you are satisfied with you comment, click the button ‘Add Comment’
6. Verify that your comment appears in the second box

E2E Test 2: Blank new lines in a comment get converted into a regular space

This test is to ensure that when a user leaves a comment, that any empty new lines will be replaced with spaces, and that the comment left on the comic page will not display these blank new lines.

1. Login to the application
2. Navigate to a comic book either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites
3. In the top box, write a comment such that the first line consists of test, press enter, then enter again, and write another line of text on the third line such that the second line is a blank new line
4. Press ‘Add Comment’
5. Verify that you comment gets posted on the second box
6. Verify that the comment that gets posted on the second box has the blank newline replaced with a space, such that, the comment is now only one line

E2E Test 3: Newest comment will be on top of all older comments

This test is done to ensure that when a user does leave a comment, it will be considered the newest one and appear on top, until another comment is added.

1. Login to the application
2. Navigate to a comic book either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites
3. Leave a comment under the comic
4. If a comment already exists, make sure your new comment appears on top of it
5. Leave another comment, verify that this comment is above your previous comment

E2E Test 4: Logout and verify users comments persist on the comic book page

This test is done to ensure that a user can not only leave a comment, but also ensure that if the user logs out and logs in again, the previously left comments will still be present with the respective comic book page

1. Login to the application
2. Navigate to a comic book either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites
3. Leave a comment on the comic page
4. Verify that the comment appears appropriately (above other ones, no blank new lines)
5. Close the comic page
6. Click the ‘Log Out’ button at the top right of the dashboard
7. Click ‘Yes’ when asked to log out
8. Verify user gets redirected to the log in page
9. Login to the application
10. Navigate to the previous comic book either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites
11. Verify that the comments you left previously on that comic is present and accurate

E2E Test 5: Many comments on a comic, to display the vertical scroll bar

This test is done to ensure that when a user does leave a comment, it will be considered the newest one and appear on top, until another comment is added.

1. Login to the application
2. Navigate to a comic book either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites
3. Begin to add so many comics that the ‘What others think about this comics’ section begins to shift all of its comments down and no longer show the older ones
4. Verify that once this happens, the vertical scroll bar becomes present
5. Verify that the user is able to user the scroll bar to view older comments

Users Can View All Comments On Individual Comic Books

E2E Test 1: Viewing comic that already has comments on it

This test is done to ensure that comments load properly on a comic book page, including their proper location and already preloaded comments

1. Login to the application
2. Navigate to the comic book Zot! either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites (if already favorited)
3. Verify on the comments section that contains two comments, one from the user ‘ahan’ at the very bottom and another from the user ‘Always_Scheming’
4. Close the comic page
5. Navigate to the comic book Sanctuary either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites (if already favorited)
6. Verify on the comments section that contains two comments, one from the user ‘ahan’ at the very bottom and another from the user 'Phoenix’

E2E Test 2: Viewing comments of a recently added comic

This test is done to ensure that once a user adds a new comic, that it will have no comments initially on it, and that a user can still comment on it

1. Login to the application
2. Navigate to the ‘Add Comic’ section by clicking on the appropriate button
3. Fill the information required to enter a comic book into the database. This information does not need to be accurate, since it is only for testing
4. When available, click the ‘Enter Comic Book’ button
5. Navigate to the recently added either through the Comic Repository section (by either double clicking on one or advanced searching) or through Favorites (if already favorited)
6. Verify that no comments exist on the comic

Oscar Ye

Users Can Browse The Full Catalog of Comic Books In The System's Repository

E2E Test 1: When logged in, navigate to the comic repository page and browse comic books.
This test measures the main scenario of a user signing into Longbox and heading straight to comic repository to see the comic books available.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the comic repository page.
4. All the comics are displayed.
5. Double clicking on any comic's title brings up its comic page.

E2E Test 2: When logged in, navigate to other pages before the comic repository page and browse comic books.
This test measures the alternative scenarios of checking out other pages before navigating to the comic repository page to see the comic books available.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the profile page.
4. Navigate to the comic repository page and check to see if the comic books are displayed.
5. Navigate to the trending page and back to the comic repository page and check if the comic books are displayed.
6. Navigate to the add comics page and back to the comic repository page and check if the comic books are displayed.
7. Navigate to the Favourites page and back to the comic repository page and check if the comic books are displayed.

E2E Test 3: When logged in, add a new comic book to the repository and browse the repository.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the 'Add Comic Page'.
4. Fill all fields with appropriate data (text in text fields and numbers in number fields).
5. Selecting the is favorite checkbox.
6. Clicking the 'Enter Comic Book' button.

7. Popup showing successful entry of book.
8. Database updated with the details added.
9. Navigate to the comic repository tab and check if the comic is visible in the repository.

Users Can Search For Individual Comic Books By Series Title

E2E Test 1: When logged in, go to the comic repository page and search for a comic by whole title.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the comic repository page.
4. Enter “Zot!” in the search bar.
5. Check to see if Zot! shows up as a result.

E2E Test 2: When logged in, go to the comic repository page and search for a comic by first letter.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the comic repository page.
4. Enter “Z” in the search bar.
5. Check to see if Zot! shows up as a result.

E2E Test 3: When logged in, go to the comic repository page and search for a comic with lowercase letters.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the comic repository page.
4. Enter “zot” in the search bar.
5. Check to see if Zot! shows up as a result.

When adding a comic book to the repository a user can directly favorite it at the same time

E2E Test 1: When logged in, go to the “add comic” page and add a new comic with the “is favorite” button checked.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the add comic page.
4. Fill all fields with appropriate data.
5. Select the “is favorite” checkbox.
6. Click the ‘Enter Comic Book’ button.
7. Popup showing successful entry of book.
8. Check the database if it is updated with the details added.
9. Check if the comic is visible in the favorites tab.
10. Check if the comic is visible in the comic repository.

E2E Test 2: When logged in, go to the “add comic” page and add a new comic without the “is favorite” button checked.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the add comic page.
4. Fill all fields with appropriate data.
5. Do not select the “is favorite” check box.
6. Click the ‘Enter Comic Book’ button.
7. Popup showing successful entry of book.
8. Check the database is updated with the details added.
9. Check if the comic is not visible in the favorites tab.
10. Check if the comic is visible in the comic repository.

Users can remove a comic book from favorites

E2E Test 1: Favoriting and unfavouriting with the comic repository “add to favorites” button and comic page’s “remove favorite button”.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the comic repository page.
4. Select Zot!
5. Click on “add to favorites” button in repository page
6. Check to see if it shows up in the favorites page.
7. Remove from favorites in Zot! comic page with “remove favorite” button.
8. Check to see if it disappears from the favorites page.

E2E Test 2: Favoriting and unfavoriting with the comic repository “add to favorites” button and favorites page’s “unfavorite” button.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the comic repository page.
4. Select Zot!
5. Click on “add to favorites” button in repository page
6. Check to see if it shows up in the favorites page.
7. Remove Zot! from favorites with the “unfavorite” button in the favorites page.
8. Confirm the pop up.
9. Check if Zot! is removed from the favorites list.

E2E Test 3: Favorite a comic from the comic page, keep it open, and unfavorite with the button in the favorites page.

1. Open application and login using any registered account or register a new account and login with it.
2. Land on the trending page.
3. Navigate to the comic repository page.
4. Select and open Zot!’s comic page
5. Click the “add to favorites” button.
6. Keep the comic page open and navigate to the favorites page.
7. Remove Zot! from favorites with the “unfavorite” button in the favorites page.
8. Confirm the pop up.
9. Check if Zot! Is removed from the favorites list.
10. Check if Zot! Is unfavorite in the comic page.
11. Zot! is still favored in the comic page.

Bug Reports By Developer/Reviewer

Ahan Bhargava

[BR4] The text is cut off in the about me text box if a big paragraph is added.

Problem Report Number	BR4
Reported By	Ahan Bhargava
Date Reported	2024-03-16
Program (or Component) Name	LongBox: Edit about me text box
Release Number	ITR2
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	macOS Sonoma 14.3.1. MacBook Pro 2021 M1 Pro Chip, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	Medium
Priority	High
Problem Summary	If the about me is bigger than the size of the textbox, the text cannot be viewed completely and is cut off, but is updated correctly in the database.
Key Words	Bug, about me, text box, editable
Problem Description & How To Reproduce It	This test case ensures that a big text can be added and read. 1) Login into the app using valid credentials. 2) Navigate to the profile page where the about me text box should be there. 3) The ‘Edit’ button is clicked. 4) A big text is added to the text box. 5) Text is cut off after a certain length.

	6) The database is updated correctly with the big paragraph.
Suggested Fix	Placing the JTextArea inside a JScrollPane
Status (Open or Closed)	Open
Resolution	
Resolved By	

[BR5] Adding a comic book with number of issues or year published empty

Problem Report Number	BR5
Reported By	Ahan Bhargava
Date Reported	2024-03-16
Program (or Component) Name	LongBox: Add a comic book to repository
Release Number	ITR1
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	macOS Sonoma 14.3.1. MacBook Pro 2021 M1 Pro Chip, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	High
Priority	High
Problem Summary	If the number of issues or year published is left empty a NumberFormatException is thrown
Key Words	Bug, add a comic book, exception
Problem Description & How To Reproduce It	This test case ensures that a big text can be added and read. 1) Login into the app using valid credentials. 2) Navigate to the ‘Add Comic Page’

	3) Fill some fields (skipped number of issues and/or year published field). 4) Not selecting the is favorite checkbox. 5) Clicking the 'Enter Comic Book' button. 6) Error thrown in the console of compiler 7) Database not updated.
Suggested Fix	Setting the year published and number of issues to 0 (default value) if a value is not supplied
Status (Open or Closed)	Closed
Resolution	Number of issues and year published can no longer be left empty or to a non positive integer value. They now have a contract associated with them to be a positive integer for the add comic book button to be clickable
Resolved By	Hashir Jamil 2024-03-18 in commit 0b93bd4

[BR6] Adding a comic book with number of issues or year published as text

Problem Report Number	BR6
Reported By	Ahan Bhargava
Date Reported	2024-03-16
Program (or Component) Name	LongBox: Add a comic book to repository
Release Number	ITR1
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	macOS Sonoma 14.3.1. MacBook Pro 2021 M1 Pro Chip, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	High
Priority	High
Problem Summary	If the number of issues or year published is

	supplied with text a NumberFormatException is thrown
Key Words	Bug, add a comic book, exception
Problem Description & How To Reproduce It	This test case ensures that a big text can be added and read. favorating the book. 1) Login into the app using valid credentials. 2) Navigate to the 'Add Comic Page' 3) In the year published/ number of issues enter text. 4) Not selecting the is favorite checkbox. 5) Clicking the 'Enter Comic Book' button. 6) Error thrown in the console of compiler. 7) Database not updated.
Suggested Fix	Setting the year published and number of issues to 0 (default value) if a value is not supplied
Status (Open or Closed)	Closed
Resolution	Number of issues and year published can no longer be entered as non positive integer text. They now have a contract associated with them to be a positive integer for the add comic book button to be clickable
Resolved By	Hashir Jamil 2024-03-18 in commit 0b93bd4

[BR7] Adding a comic book and leaving all fields empty

Problem Report Number	BR7
Reported By	Ahan Bhargava
Date Reported	2024-03-16
Program (or Component) Name	LongBox: Add a comic book to repository
Release Number	ITR1
Version (Build) Identifier	Main Branch

Configuration(s) (HW and SW)	macOS Sonoma 14.3.1. MacBook Pro 2021 M1 Pro Chip, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	High
Priority	High
Problem Summary	If all the fields of the book are left empty a NumberFormatException is thrown
Key Words	Bug, add a comic book, exception
Problem Description & How To Reproduce It	This test case ensures that a big text can be added and read. favorating the book. 1) Login into the app using valid credentials. 2) Navigate to the 'Add Comic Page' 3) Not filling any of the fields. 4) Not selecting the is favorite checkbox. 5) Clicking the 'Enter Comic Book' button. 6) Error thrown in the console of compiler. 7) Database not updated.
Suggested Fix	Having the series title as a required field, and the integer fields are set to 0 as a default value.
Status (Open or Closed)	Closed
Resolution	This form can no longer be left empty, it must be filled in for the add comic book button to be enabled.
Resolved By	Hashir Jamil 2024-03-18 in commit 0b93bd4

Ali Sina

[BR15] Login errors covered behind username text field

Problem Report Number	BR15
Reported By	Ali Sina
Date Reported	2024-03-20
Program (or Component) Name	LongBox: Login/entry panel
Release Number	ITR2
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	macOS Ventura 13.2. MacBook Air 2021 M1 Chip, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	Medium
Priority	Medium
Problem Summary	Error message covered behind username text field during testing.
Key Words	UI Bug, Error Message, Username Text Field, Testing
Problem Description & How To Reproduce It	During testing, it was observed that the error message intended to display when invalid credentials are entered is being covered by the username text field. To reproduce: 1. Navigate to the login page. 2. Enter invalid credentials in the username and password fields. 3. Observe that the error message appears but is obscured by the username text field.
Suggested Fix	Adjust the layout or positioning of the error message element to ensure it is displayed prominently and is not covered by other UI elements.

Status (Open or Closed)	Closed
Resolution	The issue has been resolved by adjusting the layout of the error message element.
Resolved By	Hashir Jamil

[BR19] Too big method initComicCollectionPage()

Problem Report Number	BR19
Reported By	Ali Sina
Date Reported	2024-03-24
Program (or Component) Name	`org.longbox.presentation.profile.ComicRepositoryPanel`
Release Number	ITR1
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	macOS Ventura 13.2. MacBook Air 2021 M1 Chip, Java 19
Report Type	Code/Design Smell
Can Reproduce	N/A
Severity	Low
Priority	Low
Problem Summary	In the `ComicRepositoryPanel` the method `initComicCollectionPage()`is a long method
Key Words	Long method, Code smell, Testing
Problem Description & How To Reproduce It	N/A
Suggested Fix	Consider breaking it down to multiple self explanatory methods.
Status (Open or Closed)	Closed
Resolution	Divided initComicCollectionLogic() into separate functions
Resolved By	Mher Eric Gyulumyan on commit 269049f

[BR20] Use Dependency Injection to access Dao in ComicRepositoryPanel

Problem Report Number	BR20
Reported By	Ali Sina
Date Reported	2024-03-24
Program (or Component) Name	`org.longbox.presentation.profile.ComicRepositoryPanel`
Release Number	ITR2
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	macOS Ventura 13.2. MacBook Air 2021 M1 Chip, Java 19
Report Type	Code/Design Smell
Can Reproduce	N/A
Severity	Low
Priority	Low
Problem Summary	In the `ComicRepositoryPanel` ComicBookDaoImpl and ComicBookFavouritesListDaoImpl are directly instantiated within the class.
Key Words	Dependency Injection, Loose Coupling, ComicRepositoryPanel
Problem Description & How To Reproduce It	N/A
Suggested Fix	Consider using dependency injection using service layer to provide these dependencies externally, which promotes loose coupling and easier unit testing.
Status (Open or Closed)	Closed
Resolution	All instances of .*DaoImpl being used in

	ComicRepository, Trending, and ComicBookSearchResults is removed and not uses ComicBookService for their respective Controller and GUI files. Now uses service layer
Resolved By	Mher Eric Gyulumyan on commit 8e6816f

Hashir Jamil Problem Reports

In this section reports are all identified with the same schema. The codes are BR followed by an integer. These are unique report id's. The bug reports are classified as type bugs and are based on end-2-end test runs. The component name for these is given as the user story the e2e test was performed on. The

[BR1] Registration with wrong email address/username leads to NullPointerException & Exception is Not Displayed

Problem Report Number	BR1
Reported By	Hashir Jamil
Date Reported	2024-03-15
Program (or Component) Name	LongBox: Registration Subsystem
Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	Windows 10 Pro 64 Bit, 1440p monitor, x86 processor, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	High
Priority	High
Problem Summary	The user is not told by the system if they try to register for the system using a previously taken username/email.
Key Words	Bug, Registration, Exception, Username, Email
Problem Description & How To Reproduce It	1. User opens the application and enters the login/entry page. 2. User selects the "Sign Up!" button. 3. The user enters all fields according to the instructions/contract except for: a. Case A: enters previously used email but new/unique username b. Case B: enters previously used username but new/unique email

	c. Case C: enters previously used email and previously used username 4. The “Sign Up for LongBox!” button becomes clickable, red message disappears and the user clicks the button. 5. The user is denied registration and told that this username and/or email is already taken. 6. The user is prompted to refill the form with a new username or email.
Suggested Fix	This problem may be a result of the new mapper classes in the domain layer being used to transform DTO/Entity objects back and forth between each other. The trace shows that the UserMapper class is implicated in a NullPointerException on the userId field. Investigate this mapping to fix this or go back to old constructor based interconversion between DTO/Entity objects.
Status (Open or Closed)	Closed
Resolution	The mapper package was initializing user Id fields on the DTO side and propagating this to the entity once being mapped. However the generation of primary key id’s for database objects must be delegated to postgresql and not created intentionally or indirectly by initializing the id field for the user DTO and entity classes. The UserMapper class was adjusted accordingly.
Resolved By	Ahan Bhargava on 2024-03-16 in commit 128976e.

[BR2] Logout + Comic Book Page Left Open Allows Comments With Exception

Problem Report Number	BR2
Reported By	Hashir Jamil
Date Reported	2024-03-16
Program (or Component) Name	Longbox Comment Service

Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	Windows 10 Pro 64 Bit, 1440p monitor, x86 processor, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	Medium
Priority	Low
Problem Summary	After logging out with the comic book page left open on a comic book, the comments feature can still be accessed but throws a null pointer exception. Even if you log back in this persists.
Key Words	Null, Pointer, Exception, Comment, Logout, Comic Book Page, Buttons, Listeners
Problem Description & How To Reproduce It	1. Log in 2. Open a comic book by double clicking a comic in the comic repository table. 3. Log out but do not close the sign in/sign up window. 4. Try to add a comment on the still open comic book page. 5. You will get a NullPointerException for a null user/userId 6. Log back in and try to comment again. 7. The same original unclosed comic book window still throws this error. 8. Only opening a new window of the same comic book resynchronizes user actions to be correct.
Suggested Fix	Close comic book page upon logout or handle all null pointer exceptions. If the page is to be left open then nullify all buttons once logged out so no one can comment.
Status (Open or Closed)	Closed
Resolution	Upon logout all active windows get closed
Resolved By	Ahan Bhargava on 2024-03-16 in commit bfde318.

[BR3] Logout + Comic Book Page Left Open Allows Add or Remove from Collection With Exception

Problem Report Number	BR3
Reported By	Hashir Jamil
Date Reported	2024-03-16
Program (or Component) Name	LongBox Comic Book Collection Service
Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	Windows 10 Pro 64 Bit, 1440p monitor, x86 processor, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	Medium
Priority	Low
Problem Summary	After logging out with the comic book page left open on a comic book, the add to favorites/reading/finished list features can still be accessed but throws a null pointer exception. Even if you log back in this persists.
Key Words	Null, Pointer, Exception, Comic Collection, Logout, Comic Book Page, Buttons, Listeners
Problem Description & How To Reproduce It	1. Log in 2. Open a comic book by double clicking a comic in the comic repository table. 3. Log out but do not close the sign in/sign up window. 4. Try to add/remove the still open comic book page to any of the lists. 5. You will get a NullPointerException for a null user/userId 6. Log back in and try to do the same again. 7. The same original unclosed comic book window still throws this error. 8. Only opening a new window of the same

	comic book resynchronizes user actions to be correct.
Suggested Fix	Close comic book page upon logout or handle all null pointer exceptions. If the page is to be left open then nullify all buttons once logged out so no one can add or remove from lists.
Status (Open or Closed)	Closed
Resolution	Upon logout all active windows get closed
Resolved By	Ahan Bhargava on 2024-03-16 in commit bfde318.

[BR8] GUI Classes Do Not Consistently Use Constants for String and Dimensions To Define Components

Problem Report Number	[BR8]
Reported By	Hashir Jamil
Date Reported	2024-03-18

Program (or Component) Name	`org.longbox.presentation.authentication.AuthenticationFrame`
	`org.longbox.presentation.authentication.RegistrationPanel`
	`org.longbox.presentation.authentication.LoginPanel`

Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	N/A
Report Type	Code/Design Smell
Can Reproduce	N/A
Severity	Low
Priority	Low
Problem Summary	Constants are not used. Too many fields are manually set and this leads to repeat code and difficult edits.

Key Words	Dependency Injection, Loose Coupling, ComicRepositoryPanel
Problem Description & How To Reproduce It	N/A
Suggested Fix	Create constants to hold common strings and dimension values.
Status (Open or Closed)	Closed
Resolution	Added constants for all strings and common integer values in the authentication package of the presentation layer.
Resolved By	Ahan Bhargava on 2024-03-19 in commit 67cc8e9 and commit 0062e74

[BR9] Service Layer is Not Used Consistently For Database Interaction and User Session Management

Problem Report Number	[BR9]
Reported By	Hashir Jamil
Date Reported	2024-03-18

Program (or Component) Name	`org.longbox.businesslogic.controller.AuthenticationController`
	`org.longbox.businesslogic.UserSession`
	`org.longbox.businesslogic.service.UserService`
	`org.longbox.persistence.dao.UserDao`
	`org.longbox.persistence.dao.UserDaoImpl`
	`org.longbox.persistence.stubdatabase.UserStubDb`

Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	N/A
Report Type	Code/Design Smell
Can Reproduce	N/A
Severity	High

Priority	High
Problem Summary	In the Authentication Controller class, the data access layer is not fully abstracted away from the controller layer by using the service layer as the middle class.
Key Words	Service, Controller, DAO, Dependency Injection
Problem Description & How To Reproduce It	N/A
Suggested Fix	The major suggestion is to use dependency injection to only create a user service instance in the authentication controller and let the service layer be given the data access object independently of the controller.
Status (Open or Closed)	Closed
Resolution	Removed UserDao instance from controller, it is now passed via the main method, making it abstract.
Resolved By	Ahan Bhargava on 2024-03-19 in commit 248a879

[BR10] Inadequate Error Logging in User Registration & Login Features

Problem Report Number	[BR10]
Reported By	Hashir Jamil
Date Reported	2024-03-18
Program (or Component) Name	`org.longbox.persistence.dao.UserDaoImpl` `org.longbox.persistence.stubdatabase.UserStubDb`
Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	N/A
Report Type	Code/Design Smell
Can Reproduce	N/A

Severity	High
Priority	High
Problem Summary	The database access methods associated with the user service are only returning generic traces.
Key Words	Logging, Exceptions, User Service
Problem Description & How To Reproduce It	N/A
Suggested Fix	The suggested fix is to create more detailed stack traces in the error states and to perhaps use formal logging.
Status (Open or Closed)	Closed
Resolution	Closed as not planned issue. Reason is that Hibernate throws multiple errors and not every error can be covered. That is why the design is chosen as such and at this time will not be refactored/fixed.
Resolved By	Ahan Bhargava on 2024-03-19

[BR11] UserDto & User Entity Mapping Too Complex

Problem Report Number	[BR11]
Reported By	Hashir Jamil
Date Reported	2024-03-18

Program (or Component) Name	`org.longbox.domainobjects.dto.UserDto`
	`org.longbox.domainobjects.entity.User`
	`org.longbox.domainobjects.mapper.UserMapper`

Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	N/A
Report Type	Code/Design Smell

Can Reproduce	N/A
Severity	Medium
Priority	High
Problem Summary	The mapping between DTO and entity should not be done using constructors. It should be done using the mapping classes. Dto and entity are Java Beans and should only have NoArgsConstructors
Key Words	Mapper, Mapping, DTO, Entity, Java Beans,
Problem Description & How To Reproduce It	N/A
Suggested Fix	Remove all overloaded constructors for both the UserDto and User entity classes. Only use setter methods and mapping classes to map between these two objects.
Status (Open or Closed)	Closed
Resolution	Removed all constructors from UserDto and User entity classes, replaced all its occurrences with Java Beans compliant classes.
Resolved By	Ahan Bhargava on 2024-03-19 in commit 2a3f726.

Mher Eric Gyulumyan

[BR15] Text for ‘Comic Repository’ button is cut-off

Problem Report Number	BR15
Reported By	Mher Eric Gyulumyan
Date Reported	22/03/2024
Program (or Component) Name	LongBox:
Release Number	ITR1
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	Windows 10, Dell G3 15, i5-9300H, Java 17
Report Type	Bug
Can Reproduce	Yes
Severity	Low
Priority	Low
Problem Summary	The leftmost button in the top dashboard titled ‘Comic Repository’ is cut off because the button is too small
Key Words	Bug, Button, Comic Repository
Problem Description & How To Reproduce It	During development, some more buttons were added than originally intended. This caused the already existing buttons to get smaller in order to fit them all in the dashboard. To reproduce this: 1. Login to the application 2. Observe that the leftmost button, ‘Comic Repository’ is not displaying its name in its entirety 3. Observe that in actuality, it says ‘Comic Reposit…’
Suggested Fix	Potential Fixes: - Should not make buttons bigger, since it would require making the other ones bigger as well

	- Best to rename it to ‘Search’
Status (Open or Closed)	Closed
Resolution	Fixed this by resizing the GUI elements on the home frame.
Resolved By	Hashir Jamil on commit 525f74a

[BR16] Username in dashboard is covered by Log Out button

Problem Report Number	BR16
Reported By	Mher Eric Gyulumyan
Date Reported	22/03/2024
Program (or Component) Name	LongBox:
Release Number	ITR2
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	Windows 10, Dell G3 15, i5-9300H, Java 17
Report Type	Bug
Can Reproduce	Yes
Severity	Low
Priority	Low
Problem Summary	The users username, which will display in between the buttons ‘Social’ and ‘Log Out’ is partially covered by the ‘Log Out’ button
Key Words	Dashboard, Username, Log Out, Button
Problem Description & How To Reproduce It	The users username is being covered by the ‘Log Out’ button on the dashboard To reproduce this: 1. Login to the application 2. Observe how on the dashboard, the users username is covered by the ‘Log Out’ button
Suggested Fix	Move the label of the users username a little more to the left

Status (Open or Closed)	Closed
Resolution	Fixed this on windows. I resized the GUI elements on the home frame
Resolved By	Fixed by Hashir Jamil on commit 525f74a

[BR17] For comments that exceed the size of the ‘Share your thoughts’ box, no vertical scroll bar gets rendered

Problem Report Number	BR17
Reported By	Mher Eric Gyulumyan
Date Reported	22/03/2024
Program (or Component) Name	LongBox:
Release Number	ITR2
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	Windows 10, Dell G3 15, i5-9300H, Java 17
Report Type	Bug
Can Reproduce	Yes
Severity	Medium
Priority	Medium
Problem Summary	When adding a lengthy comment (ie: one that will require more space than the current ‘Share your thoughts’ box) will get cut off at the bottom of the box. The user will not be able to see their comment beyond this point
Key Words	Comments, Edge Cases, Text Box
Problem Description & How To Reproduce It	The ‘Share your thoughts’ box does not render a vertical scroll bar once the given space to write down a comic becomes exhausted To reproduce this: 1. Login to the application 2. Navigate any comic book page 3. In the ‘Share your thoughts’ box, write a length comment such that it

	contains spaces and exceeds the given space 4. Observe how the text gets cut off at the bottom, and no vertical scroll bar becomes present to check out the hidden parts
Suggested Fix	Comic fans can be very territorial about their beloved stories. At certain times, they may want to express extravagant emotions about their favorite titles. To satisfy these customers, the ‘Share your thoughts’ box should render a vertical scroll bar, similar to the ‘What others think about this comics’ once more space is required to display the data
Status (Open or Closed)	Closed
Resolution	Added JTextArea inside a JScrollPane
Resolved By	Ahan Bhargava

[BR18] Comments that are single long strings (no spaces) get cut off at the right side once added

Problem Report Number	BR18
Reported By	Mher Eric Gyulumyan
Date Reported	22/03/2024
Program (or Component) Name	LongBox:
Release Number	ITR2
Version (Build) Identifier	Main Branch
Configuration(s) (HW and SW)	Windows 10, Dell G3 15, i5-9300H, Java 17
Report Type	Bug
Can Reproduce	Yes
Severity	Medium
Priority	Medium
Problem Summary	Comments that are single long strings, as in they have no spaces, appear unexpectedly

	once they get added to the comic book. That is, they get cut off on the right side of the ‘What others think about this comics’ box, likely because they contain no spaces
Key Words	Comments, Edge Case, Horizontal
Problem Description & How To Reproduce It	When adding comments to comics, edge cases like this should be considered, even if they are uncommon. To reproduce this: 1. Login to application 2. Navigate to any comic book page 3. Place a long comment with no spaces (ie. hold the ‘g’ key for seven seconds) 4. Click ‘Add Comment’ 5. Observe that despite the original comment being “multiple lines”, the posted comment below now is only one line and gets cut off on the right
Suggested Fix	Potential Fixes: - Add a check to verify a single ‘word’ in a comment does not exceed a certain number of characters - Do not allow for comments to have no spaces
Status (Open or Closed)	Open
Resolution	Not resolved
Resolved By	

Oscar Ye

[BR12] Favouriting and unfavouriting comics from favorites page does not update it in the comic's page

Problem Report Number	BR12
Reported By	Oscar Ye
Date Reported	2024-03-20
Program (or Component) Name	LongBox: Favoriting system
Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	Windows 10 Home 64 Bit, 1440p monitor, x86 processor, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	Low
Priority	Low
Problem Summary	Favoriting or unfavoriting a comic with any of the buttons in the comic repository page or the favorites page does not update the information in the comic’s page
Key Words	Favorite, unfavourite, refresh
Problem Description & How To Reproduce It	Open a comic’s page from the comic repository. I Like it on the comic’s page. Move to the favorites page. Unfavourite the comic from the favorites page. The comic is still marked as favorites in the comic’s page.
Suggested Fix	Add a way to refresh the comic’s page when the comic is favorites or favorite.
Status (Open or Closed)	Open
Resolution	
Resolved By	

[BR13] Cannot unfavourite more than one comic in a row

Problem Report Number	BR13
Reported By	Oscar Ye
Date Reported	2024-03-15
Program (or Component) Name	LongBox: Favoriting system
Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	Windows 10 Home 64 Bit, 1440p monitor, x86 processor, Java 19
Report Type	Bug
Can Reproduce	Yes
Severity	Medium
Priority	Medium
Problem Summary	Cannot unfavourite more than one comic at a time in the favorites page.
Key Words	Favorite, unfavorite, refresh, FavoritesController, FavoritesPanel
Problem Description & How To Reproduce It	Add multiple comics to favorites from the comic repository. Navigate to the favorites panel. Cannot unfavorite more than a single comic at a time.
Suggested Fix	n/a
Status (Open or Closed)	Closed
Resolution	Resolved the issue by adding a conditional check to ensure that the selected row is not equal to -1 after removing the row from the database.
Resolved By	Ali Sina on 2024-03-15 in commit 5711fb9.

[BR14] Inconsistency of how favorite is spelled throughout the whole repo

Problem Report Number	BR14
Reported By	Oscar Ye
Date Reported	2024-03-20
Program (or Component) Name	LongBox: Favoriting system
Release Number	ITR2
Version (Build) Identifier	Main branch
Configuration(s) (HW and SW)	n/a
Report Type	Code Smell
Can Reproduce	n/a
Severity	Low
Priority	Low
Problem Summary	Sometimes favorites is spelled like “favorite” and sometimes it’s spelled “favorite”
Key Words	Favorite, favourite, spelling, refactor
Problem Description & How To Reproduce It	n/a
Suggested Fix	Change all instances of “favorite” to “favorite”
Status (Open or Closed)	Open
Resolution	The Canadian spelling of “favorite” is now employed.
Resolved By	Hashir Jamil

Code Review By Developer

User Stories & Artifacts Division

User Story/Feature	**Reviewer**	Relevant Artifacts (Plus All Tests With Same Name)

Users can register for LongBox	Hashir Jamil	`org.longbox.businesslogic.controller.AuthenticationController`
		`org.longbox.businesslogic.controller.HomeController`
		`org.longbox.businesslogic.UserSession`
		`org.longbox.businesslogic.exception`
		`org.longbox.businesslogic.service.UserService`
		`org.longbox.domainobjects.dto.UserDto`
		`org.longbox.domainobjects.entity.User`
		`org.longbox.domainobjects.mapper.UserMapper`
		`org.longbox.persistence.dao.UserDao`
		`org.longbox.persistence.dao.UserDaoImpl`
		`org.longbox.persistence.stubdatabase.UserStubDb`
		`org.longbox.presentation.authentication.AuthenticationFrame`
		`org.longbox.presentation.authentication.LoginPanel`
		`org.longbox.presentation.authentication.RegistrationPanel`

Users can login to Longbox	Hashir Jamil	`org.longbox.businesslogic.controller.AuthenticationController`
		`org.longbox.businesslogic.controller.HomeController`
		`org.longbox.businesslogic.UserSession`
		`org.longbox.businesslogic.exception`
		`org.longbox.businesslogic.service.UserService`
		`org.longbox.domainobjects.dto.UserDto`
		`org.longbox.domainobjects.entity.User`
		`org.longbox.domainobjects.mapper.UserMapper`
		`org.longbox.persistence.dao.UserDao`
		`org.longbox.persistence.dao.UserDaoImpl`
		`org.longbox.persistence.stubdatabase.UserStubDb`
		`org.longbox.presentation.authentication.AuthenticationFrame`
		`org.longbox.presentation.authentication.LoginPanel`
		`org.longbox.presentation.authentication.RegistrationPanel`

Once users log in, users	Hashir Jamil	`org.longbox.businesslogic.controller.AuthenticationController`

remain logged in and maintain this state.		org.longbox.businesslogic.controller.HomeController org.longbox.businesslogic.UserSession org.longbox.presentation.profile.HomeFrame
Once logged in, users can choose to logout at any time.	Hashir Jamil	org.longbox.businesslogic.controller.HomeController org.longbox.businesslogic.UserSession org.longbox.presentation.profile.HomeFrame
Users Can Login Using Email Address and Password	Ali Sina	org.longbox.businesslogic.controller.AuthenticationController org.longbox.businesslogic.service.UserService org.longbox.domainobjects.dto.UserDto org.longbox.domainobjects.entity.User org.longbox.domainobjects.mapper.UserMapper org.longbox.persistence.dao.UserDao org.longbox.persistence.dao.UserDaoImpl org.longbox.persistence.stubdatabase.UserStubDb org.longbox.presentation.authentication.AuthenticationFrame org.longbox.presentation.authentication.LoginPanel org.longbox.presentation.authentication.RegistrationPanel org.longbox.presentation.profile.HomeFrame
Users Have Access To A Menu Of Links To All Aspects of the Home Dashboard/Profile	Mher Eric Gyulumyan	org.longbox.businesslogic.controller.AuthenticationController org.longbox.businesslogic.controller.HomeController org.longbox.presentation.profile.HomeFrame org.longbox.presentation.profile.SearchPanel org.longbox.presentation.profile.TrendingPanel org.longbox.presentation.profile.FavoritesPanel org.longbox.presentation.profile.ComicRepositoryPanel org.longbox.presentation.profile.AddComicToRepoPanel
Users Can View Their Personal Profile Information	Mher Eric Gyulumyan	org.longbox.businesslogic.controller.HomeController org.longbox.domainobjects.dto.UserDto org.longbox.domainobjects.entity.User org.longbox.domainobjects.mapper.UserMapper org.longbox.persistence.dao.UserDao org.longbox.persistence.dao.UserDaoImpl org.longbox.persistence.stubdatabase.UserStubDb org.longbox.presentation.profile.ProfilePanel

User can edit and save “about me” text box in profile page (Ahan)	Ahan Bhargava	org.longbox.presentation.profile (about me boxes and buttons) org.longbox.domainobjects.dto.UserDto (about me) org.longbox.persistence.dao.UserDaoImpl (updateAboutMeString) org.longbox.businesslogic.controller.ProfileController
Users Can Browse The Full Catalog of Comic Books In The System’s Repository (Oscar)	Oscar Ye	org.longbox.businesslogic.controller.ComicRepositoryController org.longbox.presentation.tablemodels.ComicBookTableModels org.longbox.presentation.profile.ComicRepositoryPanel
Users Can Search For Individual Comic Books By Series Title (Oscar)	Oscar Ye	org.longbox.presentation.profile.ComicRepositoryPanel org.longbox.businesslogic.controller.ComicRepositoryController org.longbox.presentation.tablemodels.ComicBookTableModel
Users Can Add A Comic Book To The System Repository	Ahan Bhargava	org.longbox.businesslogic.controller.HomeController org.longbox.businesslogic.service.ComicBookService org.longbox.domainobjects.dto.ComicBookDto org.longbox.domainobjects.entity.ComicBook org.longbox.persistence.dao.ComicBookDao org.longbox.persistence.dao.ComicBookDaoImpl org.longbox.persistence.dao.stubdatabase.ComicBookStubDb org.longbox.presentation.profile.AddComicToRepoPanel
Users Can Sort A Comic Table Based on Every Attribute that a Comic Has	Ali Sina	org.longbox.presentation.profile.ComicRepositoryPanel org.longbox.presentation.tablemodels.ComicBookTableModel org.longbox.domainobjects.dto.ComicBookDto org.longbox.domainobjects.entity.ComicBook
User can Search for Comic by Title, Genre, Publisher, and Year	Ali Sina	org.longbox.presentation.profile.ComicRepositoryPanel org.longbox.businesslogic.controller.ComicRepositoryController org.longbox.businesslogic.controller.ComicBookSearc

		hResultsController org.longbox.presentation.comicbook.CoicBookSearchResultsFrame org.longbox.businesslogic.utils.ComicBookSearchUtils org.longbox.businesslogic.controller.ComicBookInfoController org.longbox.presentation.comicbook.ComicBookInfoPanel org.longbox.presentation.comicbook.ComicBookFrame
User Can Navigate on a Comic Book Page by Clicking its Title if it Exists in a Results Table and Comic Collection	Ali Sina	org.longbox.businesslogic.controller.ComicRepositoryController org.longbox.businesslogic.utils.ComicBookSearchUtils org.longbox.presentation.tablemodels.ComicBookTableModel org.longbox.businesslogic.controller.ComicBookInfoController org.longbox.presentation.comicbook.ComicBookInfoPanel org.longbox.presentation.comicbook.ComicBookFrame
Users Can Write Comments On Individual Comic Books	Mher Eric Gyulumyan	Org.longbox.presentation.comicbook.ComicBookInfoPanel org.longbox.businesslogic.service.CommentService org.longbox.businesslogic.utils.MultiLineCellRenderer org.longbox.domainobjects.dto.CommentDto Org.longbox.businesslogic.controller.ComicBookInfoController org.longbox.persistence.dao.CommentDaoImpl ComicBookInfoController :: ActionPerformed (first if statement)
Users can View All Comments On Individual Comic Books	Mher Eric Gyulumyan	Org.longbox.presentation.comicbook.ComicBookInfoPanel org.longbox.businesslogic.service.CommentService org.longbox.businesslogic.utils.MultiLineCellRenderer org.longbox.domainobjects.dto.CommentDto

		ComicBookInfoPanel :: displayComments()
Users Can Add/Remove Comic Books To/From A List Of Finished Comic Books (Ahan)	Ahan Bhargava	org.longbox.presentation.comicbook.ComicBookInfoPanel org.longbox.presentation.tablemodels.ReadingAndFinishedComicBookTableModel org.longbox.presentation.comicbook.ComicBookInfoPanel :: isComicInFinished org.longbox.presentation.comicbook.ComicBookInfoPanel :: finishedButtonStates org.longbox.businesslogic.controller.ComicBookInfoController org.longbox.persistence.dao.ComicBookFinishedListDaoImpl org.longbox.presentation.profile.ProfilePanel
Users Can Add/Remove Comic Books To/From A List Of Reading In Progress Comic Books (Ahan)	Ahan Bhargava	org.longbox.presentation.comicbook.ComicBookInfoPanel org.longbox.presentation.tablemodels.ReadingAndFinishedComicBookTableModel org.longbox.presentation.comicbook.ComicBookInfoPanel :: isComicInFinished org.longbox.presentation.comicbook.ComicBookInfoPanel :: finishedButtonStates org.longbox.businesslogic.controller.ComicBookInfoController org.longbox.persistence.dao.ComicBookReadingListDaoImpl org.longbox.presentation.profile.ProfilePanel
When adding a comic book to the repository a user can directly favorite it at the same time	Oscar Ye	Org.longbox.businesslogic.controller.homecontroller org.longbox.presentation.profile.FavoritesPanel org.longbox.persistence.dao.ComicBookFavouritesListDaoImpl org.longbox.presentation.profile.AddComicToRepoPanel
Users can remove a comic book from favorites	Oscar Ye	org.longbox.presentation.profile.FavoritesPanel org.longbox.persistence.dao.ComicBookFavouritesListDaoImpl org.longbox.businesslogic.controller.FavoritesController

Ahan Bhargava

1) User can edit and save “about me” text box in profile page

Relevant Artifacts:

`org.longbox.presentation.profile (about me boxes and buttons)`
`org.longbox.domainobjects.dto.UserDto (about me)`
`org.longbox.persistence.dao.UserDaoImpl (updateAboutMeString)`
`org.longbox.businesslogic.controller.ProfileController`

Review

- For the `setBounds`, some of the variables are repeated and can be set using a constant variable. (Magic Numbers)
- `UserDto` for this feature is not being used explicitly but in general is clean and has no unnecessary methods or constructors.
- `UserDaoImpl` method `updateAboutMeString()`, uses a deprecated method, this can be changed.
- The `ProfileController` class follows the pattern that is followed by the other controllers for this project.
    - The `actionPerformed()` method is big, all the business logic is inside the if cases, they should be handled by methods.
    - `Strings` are being compared for equality using `‘==’`, while it should use the `.equals()` method since `String` is an object (non-primitive type).
    - We are abstracting the database layer from the client code by using a service layer, but code is communicating directly with the database, dependency injection should be used.
    - The `UserDaoImpl` object is being created multiple times, it should be a global object and that instance should be used for one session, currently multiple objects are being created.
    - There are repeated method (code duplication) calls, this can be converted into a method in the GUI class and that method can be called. As well as method chaining is being done.
    - The `actionPerformed()` is using nested if cases, which can be confusing to read and debug at times.

2) Users Can Add A Comic Book To The System Repository

Relevant Artifacts:

`org.longbox.businesslogic.controller.HomeController`
`org.longbox.businesslogic.service.ComicBookService`
`org.longbox.domainobjects.dto.ComicBookDto`
`org.longbox.domainobjects.entity.ComicBook`
`org.longbox.persistence.dao.ComicBookDao`
`org.longbox.persistence.dao.ComicBookDaoImpl`
`org.longbox.persistence.dao.stubdatabase.ComicBookStubDb`
`org.longbox.presentation.profile.AddComicToRepoPanel`

Review

- The `AddComicToRepoPanel` has a big method called `initAddComicToRepoPage()`.
    - This can be broken up into several smaller methods based on functionalities, such as adding labels, adding fields, adding focus and document listeners, etc..
    - All labels are initialized as local variables and should be defined as global variables.
    - There are comments for the fields, but if they were labeled by the input parameter as compared to numbers it would be better to read the code.
    - All values for `setBounds` are hard coded, some common values can be clubbed together and can be initialized as constants.
- `HomeController` is a big class and should be broken down into separate controllers for each of the panels. The `actionPerformed` method for this class is big and can be shortened by removing the business logic and breaking them into separate methods.
    - This feature uses a method called `saveAddComicBookFromInput()`.
    - This method is big and there are multiple operations being performed in a single method.
    - The `ComicBookDto` object to be added is being constructed in the controller class itself, it can be moved to the GUI class and a method can be made to return the `ComicBookDto` object that must be returned. This would ensure that there is no long method changing done to extract text.
    - The way edge cases are handled is verbose and complex, also it is being handled when the object is being created, it should be done before creating the object.
    - The text resetting is done via a chain of method calls and can be confusing to read, this again can be converted to a method called `clearFields()` in the GUI class and a call can be made.
    - Multiple operations are being done at the same time, the favoriting of a comic book can have its separate method.
    - There is a method chaining done to get the state of Is Favorite checkbox, a getter can be used in its place.

    - Improper exception handling is done, the `UserIDDoesNotExistException` is being thrown instead it should be handled and appropriate error message should be shown to the user.
    - `JOptionPane,` a UI component is initialized inside the controller and business logic, it should be moved to the GUI class.
- For `ComicBookDaoImpl,` error handling is not proper, a generic error stack is being printed.
- For `ComicBookDto` and `ComicBook,` multiple constructors are being used, this can be changed, and Java Beans can be used.

3) Users Can Add/Remove Comic Books To/From A List Of Finished In Progress Comic Books

4) Users Can Add/Remove Comic Books To/From A List Of Reading In Progress Comic Books

Stories 3 and 4 follow similar code structure and have the same classes, so clubbed the review.

Relevant Artifacts

`org.longbox.presentation.comicbook.ComicBookInfoPanel`
`org.longbox.presentation.tablemodels.ReadingAndFinishedComicBookTableModel`
`org.longbox.presentation.comicbook.ComicBookInfoPanel :: isComicInFinished`
`org.longbox.presentation.comicbook.ComicBookInfoPanel :: finishedButtonStates`
`org.longbox.businesslogic.controller.ComicBookInfoController`
`org.longbox.persistence.dao.ComicBookFinishedListDaoImpl`
`org.longbox.persistence.dao.ComicBookReadingListDaoImpl`
`org.longbox.presentation.profile.ProfilePanel`

Review

- The `ComicBookInfoPanel` is a big class with some business logic (`displayAvgRating`), this can be moved out of the class and perhaps a util class can be made with a static method to calculate the ratings.
    - There are constants being used but not throughout, all the strings for labels should be converted to static strings and should be final. (Magic Strings)
    - The values for `setBounds` are currently hard coded, common ones should be converted to constants.
    - There is some unused code that should be removed for better readability and maintenance.
    - Buttons are dynamic in Swing, and that should be used, currently the GUI is cluttered as it has many buttons.
- `ReadingAndFinishedTableModel`, everything is being done inside the constructor, another method should be added, it would be better for readability and maintenance.
- The `ComicBookInfoController`
    - Has a big action performed method, as most of the business logic is being implemented within the if statements, separate methods should be used.
    - Abstraction of client code and database not done as the favorites button, reading button and finished button still directly communicate with Dao instead of via a service layer.
    - `ComicBookFavouritesListDaoImpl`, `ComicBookReadingListDao` and `ComicBookFinishedListDao` objects are being created multiple times, it should be a global object and that instance should be used for one session, currently multiple objects are being created.

    - There is a lot of code duplication inside the action performed method.
    - Improper exception handling as a runtime error is being thrown instead of it being handled.
    - There is a lot of method chaining going on, instead there should be getters for such objects, it leads to better readability and maintainability of the code.
- The `ProfilePanel`, `reloadTables()` method
    - It is responsible for loading and reloading the data into the tables, and it is a large method.
    - `UserDao`, `ComicBookReadingListDao` and `ComicBookFinishedListDao` objects are being created multiple times, it should be a global object and that instance should be used for one session, currently multiple objects are being created.
    - Abstraction of client code and database not done as `Dao's` are being used directly instead of through a service layer.
    - Similar names for `readingList` and `readList`, it was confusing to read the code, `readingList` can be renamed to `finishedList`.
    - Currently there is controller code inside the GUI code (`mouseListener`), this should be moved to the controller class.
    - Improper handling for `UserIDDoesNotExistException`.
    - Code duplication, the lists are being converted from entity type to `Dto` type in two steps, this can be done using the `ComicBookMapper` class, and code duplication can be removed.
    - Values for `setBounds` are hard coded, similar values can be clubbed together and converted to constants.
    - Long parameter lists, variables of the same type can be clubbed together and be denoted by commas.
- `ComicBookFinishedListDao` and `ComicBookReadingListDao` have two methods each which are currently not being used. (Speculative Generality)

Conclusion

For this assignment, based on the sources provides, I planned to check the following code smells:

- **Bloaters**
  - **Long Classes**
    - Found in GUI classes such as `ProfilePanel` and `ComicInfoPanel`.
  - **Long Method**
    - Found in `actionPerformed()` of controllers and `saveAddComicBookFromInput()`, these methods were big and had multiple operations being performed within them.
  - **Primitive Obsession**
    - Found in GUI classes where we could see magic numbers as well as magic strings.
  - **Long Parameter List**
    - Found in GUI classes where the same type of variable was defined individually rather than in a comma separated list.
- **Dispensable**
  - **Duplicate Code**
    - Found in the `reloadTables()` method of `ProfilePanel.`
  - **Dead Code**
    - Found in `ComicBookInfoPanel.`
  - **Speculative Generality**
    - `ComicBookFinishedListDao` and `ComicBookReadingListDao` have two methods each which are currently not being used, either those methods should be used or removed.
- **Couplers**
  - **Feature Envy, Inappropriate Intimacy and Message Chaining**
    - This is found in controller classes, where there is a lot of method chaining, this can be avoided by moving the chain calls as methods inside the parent class.
- **Miscellaneous**
  - Comparison of Strings using `'=='` instead of `.equals()`.
  - Improper use of dependency injection, for most of the code the client code was directly interacting with the database instead of going through the service layer.
  - Similar names, creating confusion while reading the code.
  - Improper exception handling, there were errors being thrown instead of being handled.
  - Code in incorrect places, there was business logic inside GUI classes and Controller logic inside GUI classes.

Ali Sina

1) Users Can Login Using Email Address and Password

Relevant Artifacts:

org.longbox.businesslogic.controller.AuthenticationController
org.longbox.businesslogic.service.UserService
org.longbox.domainobjects.dto.UserDto
org.longbox.domainobjects.entity.User
org.longbox.domainobjects.mapper.UserMapper
org.longbox.persistence.dao.UserDao
org.longbox.persistence.dao.UserDaoImpl
org.longbox.persistence.stubdatabase.UserStubDb
org.longbox.presentation.authentication.AuthenticationFrame
org.longbox.presentation.authentication.LoginPanel
org.longbox.presentation.authentication.RegistrationPanel
org.longbox.presentation.profile.HomeFrame

Review

- In the `AuthenticationController` the controller has a dependency on `'UserDaoImpl'` directly. Consider initializing dao inside the method to be independent externally.
- In the `AuthenticationController` the name for `decryptedPassword` might be misleading if the password is not actually decrypted in this context.

2) Users Can Sort A Comic Table Based on Every Attribute that a Comic Has

Relevant Artifacts:

`org.longbox.presentation.profile.ComicRepositoryPanel`
`org.longbox.presentation.tablemodels.ComicBookTableModel`
`org.longbox.domainobjects.dto.ComicBookDto`
`org.longbox.domainobjects.entity.ComicBook`

Review

- In the `ComicRepositoryPanel` the method `initComicCollectionPage()`is a long method, consider breaking it down to multiple self explanatory methods.
- In the `ComicRepositoryPanel` the `currentItem` variable is declared but not used anywhere
- In the `ComicRepositoryPanel` ComicBookDaoImpl and ComicBookFavouritesListDaoImpl are directly instantiated within the class. Consider using dependency injection using service layer to provide these dependencies externally, which promotes loose coupling and easier unit testing.

3) User can Search for Comic by Title, Genre, Publisher, and Year

Relevant Artifacts:

`org.longbox.presentation.profile.ComicRepositoryPanel`
`org.longbox.businesslogic.controller.ComicRepositoryController`
`org.longbox.businesslogic.controller.ComicBookSearchResultsController`
`org.longbox.presentation.comicbook.CoicBookSearchResultsFrame`
`org.longbox.businesslogic.utils.ComicBookSearchUtils`
`org.longbox.businesslogic.controller.ComicBookInfoController`
`org.longbox.presentation.comicbook.ComicBookInfoPanel`
`org.longbox.presentation.comicbook.ComicBookFrame`

Review

- In `ComicRepositoryController` the search logic is intertwined with UI interaction in the actionPerformed() method which can lead to a violation of the Single Responsibility Principle (SRP). Consider extracting the search logic into a separate method.

- In `ComicRepositoryController` the search functionality interacts directly with the `ComicBookDaoImpl` to perform the search operation. Consider using dependency injection using service layer to provide these dependencies externally, which promotes loose coupling and easier unit testing.

- In `ComicRepositoryController` there is a lack of error handling within the search functionality, if there was an error such as database connection error, invalid input, the code should handle it.

4) User Can Navigate on a Comic Book Page by Clicking its Title if it Exists in a Results Table and Comic Collection

Relevant Artifacts:

`org.longbox.businesslogic.controller.ComicRepositoryController`
`org.longbox.businesslogic.utils.ComicBookSearchUtils`
`org.longbox.presentation.tablemodels.ComicBookTableModel`
`org.longbox.businesslogic.controller.ComicBookInfoController`
`org.longbox.presentation.comicbook.ComicBookInfoPanel`
`org.longbox.presentation.comicbook.ComicBookFrame`

Review

- In `ComicBookSearchUtils` for search criteria methods (title, author, artist, genre, publisher, year), similar code structures are repeated. Consider refactoring these methods to a single, more generic search method that accepts search criteria as parameters.
- In `ComicBookSearchUtils` in the `searchComicBook` method, the variable comicBook is initialized but never used, try to delete the code and return null in case the search result is empty.
- In `ComicBookSearchUtils` in the `loadComicBookPage` there is inconsistency in terms of `setVisible(true)` to either be in the frame constructor or in the method frame being initialized.

Hashir Jamil Code Review

In this section the various code smells that have been checked are listed and justified. After this the four relevant user stories have been divided up to show which exact Java files are relevant to them. In each stories section the general overview and suggestions are provided. To finish off the code review all detected code smells across the four stories are listed and grouped. The smells are grouped together because files in the four stories have substantial overlap.

Code Smells That Have Been Checked:

Categories 1 to 5 are general categories derived from the following wiki sources:
Source Making - Refactoring Bad code smells
Refactoring Guru - Code Smells
Pragmatic Ways - 31 Code Smells all software engineers must watch out for
Category 6 is additional smells created ad-hoc to help better describe an artifact under scrutiny that may not fit nicely into any of the general refactoring patterns listed in the above sources.

1. Bloaters
    i. Long Method
    ii. Large Class
    iii. Primitive Obsession
    iv. Long Parameter List
    v. Data Clumps
2. Object-Orientation Abusers
    i. Switch Statements
    ii. Temporary Field
3. Change Preventers
    i. Divergent Change
    ii. Shotgun Surgery
4. Dispensables
    i. Duplicate Code
    ii. Lazy Class
    iii. Dead Code
    iv. Speculative Generality
5. Couplers
    i. Feature Envy
    ii. Inappropriate Intimacy
    iii. Message Chains
    iv. Middle Man
6. Miscellaneous (Some are ad-hoc and others are found from internet sources)
    i. Naming & Scope Pitfall
        ➢ Class, variable, method, constant, parameter names are not meaningful and their scopes are not correct.
        ➢ This category helped to denote cases where constants were not used properly and magic strings/absolute positions were set in UI components.
    ii. Incomplete, Multifaceted, Unused and Repeated Abstraction
        ➢ The level of abstraction inside a class or package is not consistent.
    iii. Lazy Exception Handling
        ➢ Exceptions are handled directly or hard coded.
    iv. Dependency Injection Pitfall
        ➢ This smell concerns not using dependency injection properly leaving a class tightly coupled.
    v. Insufficient Logging
        ➢ This smell is indicative of not creating meaningful stack traces for exceptions and not using direct logging mechanisms.

Story 1: Users can register for LongBox

Relevant Artifacts

`org.longbox.businesslogic.controller.AuthenticationController`
`org.longbox.businesslogic.controller.HomeController`
`org.longbox.businesslogic.UserSession`
`org.longbox.businesslogic.exception`
`org.longbox.businesslogic.service.UserService`
`org.longbox.businesslogic.utils.RegistrationUtils`
`org.longbox.domainobjects.dto.UserDto`
`org.longbox.domainobjects.entity.User`
`org.longbox.domainobjects.mapper.UserMapper`
`org.longbox.persistence.dao.UserDao`
`org.longbox.persistence.dao.UserDaoImpl`
`org.longbox.persistence.stubdatabase.UserStubDb`
`org.longbox.presentation.authentication.AuthenticationFrame`
`org.longbox.presentation.authentication.RegistrationPanel`

Review & Suggestions

1. There is a lack of use of constants, especially for GUI components.
    - ➢ User constants to represent as many of the strings and positions used in GUI components. There are many string written in line when components are made
2. Many variables are declared where they are used when they should be declared at the class level. This is a problem in the GUI class `RegistrationPanel`.
    - ➢ Many of the JLabel objects in the form are declared in the constructor. They should probably be scoped one level up to the actual class level fields.
3. The `RegistrationPanel` has very long methods and the class ends up being very long in general. This may be unavoidable but it's likely the `updateButtonState` method can be broken up and made simpler.
4. Good job on the exceptions. Almost all imaginable cases are handled.
5. The `UserDto` and `User` classes have complex mapping logic that is getting messy and may cause errors when extended or built upon. There are too many constructors and we have already seen problems happening when we try to work around these classes.
    - ➢ Since these two classes are technically “Java Beans” they should only have a NoArgsConstructor and fields should be set by mutators. Mapping should be performed via the `UserMapper` class only.
6. The `UserService` should be fully utilized for any database interaction. Right now the abstraction created around the service layer is not consistently used.
    - ➢ All communications between the database and the GUI/controllers should be relayed through the service layer to reduce coupling. The `AuthenticationController` class still has coupling with the `UserDaoImpl` class. The major issue here is that it doesn’t allow mocking of databases and it

also doesn't abstract away data access. The service layer should be the only point of communication here.

7. Dependency injection and dependencies are poorly handled. This is done by instantiating objects at the class level scope and not in the constructor. This occurs in the `AuthenticationController` class. The following two lines are declared fields that get instantiated right away:

```
UserDaoImpl userDaoImpl = new UserDaoImpl(HibernateUtils.getSessionFactory());
UserService userService = new UserService(userDaoImpl);
```

   - ➢ Only `UserService` should be here and it should be instantiated in the constructor.
8. Error logging can be improved in the `UserDaoImpl` class. Right now it is only printing generic stack traces
9. The error logging in the `AuthenticationController` class is much better and problem 8. Can be done as such.

Story 2: Users Can Login to LongBox

Relevant Artifacts

`org.longbox.businesslogic.controller.AuthenticationController`
`org.longbox.businesslogic.controller.HomeController`
`org.longbox.businesslogic.UserSession`
`org.longbox.businesslogic.exception`
`org.longbox.businesslogic.service.UserService`
`org.longbox.domainobjects.dto.UserDto`
`org.longbox.domainobjects.entity.User`
`org.longbox.domainobjects.mapper.UserMapper`
`org.longbox.persistence.dao.UserDao`
`org.longbox.persistence.dao.UserDaoImpl`
`org.longbox.persistence.stubdatabase.UserStubDb`
`org.longbox.presentation.authentication.AuthenticationFrame`
`org.longbox.presentation.authentication.LoginPanel`

Review & Suggestions

Several of the points to note are similar to Story 2 as there are common files between the two stories.

1. There is a lack of use of constants, especially for GUI components.
    - ➢ User constants to represent as many of the strings and positions used in GUI components. There are many strings written in line when components are made in the `AuthenticationController` class.
2. Many variables are declared where they are used when they should be declared at the class level. This is a problem in the GUI class `LoginPanel`.
    - ➢ Many of the JLabel objects in the form are declared in the constructor. They should probably be scoped one level up to the actual class level fields.
3. The `UserDto` and `User` classes have to be mapped by the `UserMapper` class just as mentioned in the previous section.
4. Delegate tasks through the service layer.
5. In the `AuthenticationController` class the `validateLogin` method is long and has many complex branches.
    - ➢ Extracting this method into smaller pieces of code may be beneficial.
6. Don’t interact directly with data access objects, use the service layer instead.

Story 3: Once Users Login, Users Remain Logged in and Maintain this State.

Relevant Artifacts

`org.longbox.businesslogic.controller.AuthenticationController`
`org.longbox.businesslogic.controller.HomeController`
`org.longbox.businesslogic.UserSession`
`org.longbox.businesslogic.service.UserService`
`org.longbox.presentation.profile.HomeFrame`

Review & Suggestions

1. The `UserSession` class is implementing the singleton creational pattern with the lazy initialization method. However, the self reference to an object of UserSession type is initialized as null. This is bad practice.
    - ➢ There should not be any null initialization. It should be only performed in the constructor.
2. Aside from this, session management and persistence is done very well.
3. One further improvement can be to have database level states for being logged in or logged out. This could help build better features to provide more user value in the future.

Story 4: Once Logged in, Users Can Choose to Logout at Any Time.

Relevant Artifacts

`org.longbox.businesslogic.controller.HomeController`
`org.longbox.businesslogic.UserSession`
`org.longbox.businesslogic.service.UserService`
`org.longbox.presentation.profile.HomeFrame`

Review & Suggestions

1. There is a lack of use of constants, especially for GUI components.
    - ➢ User constants to represent as many of the strings and positions used in GUI components. There are many string written in line when components are made
2. The `UserService` should be fully utilized for ending a session and logging out.
    - ➢ The `UserSession` class's `clearSession` method should be connected to the service layer by working through `UserService`.

Code Smells Detected Across All Stories:

In the following multilevel list level 1 denotes the code smell detected, level 2 denotes the artifact(s) where the smell was detected and level 3 denotes the justification as well as potential improvements to the code.

1. Large Class
    i. `org.longbox.businesslogic.controller.AuthenticationController`
        - ➢ seems to handle both UI interactions and business logic related to authentication and user registration. This violates the single responsibility, as it should ideally have separate classes for UI handling and business logic.

    ii. `org.longbox.businesslogic.controller.HomeController`
        - ➢ combines responsibilities related to UI interaction, business logic (such as saving comic book details), and session management (such as logging out users).
        - ➢ Potential violation of single responsibility leading to hard long term maintenance

    iii. `org.longbox.persistence.stubdatabase.UserStubDb`
        - ➢ The class has multiple responsibilities, including acting as a DAO (Data Access Object) and handling JSON serialization/deserialization. It is implementing two interfaces. It should be split into two classes.

2. Long Method
    i. `org.longbox.businesslogic.controller.AuthenticationController`

    ii. `org.longbox.businesslogic.controller.HomeController`
        - ➢ The `actionPerformed` method is quite large in both classes and handles multiple responsibilities. It should be broken down into smaller, more focused methods.

    iii. `org.longbox.domainobjects.dto.UserDto`

    iv. `org.longbox.domainobjects.entity.User`
        - ➢ Both of these classes have some very long and convoluted constructors
        - ➢ This will lead to errors in the long run.

3. Feature Envy
    i. `org.longbox.domainobjects.dto.UserDto`

    ii. `org.longbox.domainobjects.entity.User`
        - ➢ Both of these classes seem to rely on each other's methods to instantiate objects leading to tight coupling between the two objects.
        - ➢ To fix this rely solely on the UserMapper class to convert and remove the constructors.

    iii. `org.longbox.businesslogic.controller.AuthenticationController`
        - ➢ Business logic such as user validation and registration is directly implemented in the controller (`validateLogin` and `registerUser` methods). This violates the Separation of Concerns principle

    iv. `org.longbox.businesslogic.controller.HomeController`
        - ➢ The controller is tightly coupled to specific implementations such as `ComicRepositoryController`, `TrendingController`, `ProfileController`, and `ComicBookService`. This tight coupling makes the controller less flexible and harder to maintain.

4. Inappropriate Intimacy
    i. `org.longbox.businesslogic.controller.AuthenticationController`
        - ➢ The class directly interacts with UI components (`AuthenticationFrame`, `RegistrationPanel`) to retrieve data and display messages. This tight coupling makes the class less modular and harder to test in isolation.
5. Duplicate Code
    i. `org.longbox.businesslogic.controller.AuthenticationController`
        - ➢ There are repetitive patterns in event handling (`actionPerformed()`), such as checking the event source and performing actions based on it. This could lead to code duplication if similar patterns exist elsewhere in the application.
    ii. `org.longbox.domainobjects.dto.UserDto`
        - ➢ The same initialization code for default values (`joinDate`, `comicsReading`, `comicsFinished`) is repeated across multiple constructors.
6. Incomplete, Multifaceted, Unused and Repeated Abstraction
    i. `org.longbox.businesslogic.controller.AuthenticationController`
        - ➢ The class contains both high-level actions (such as `registerUser()`) and low-level UI manipulation (such as changing colors of error labels). This mixing of concerns can make the code harder to understand and maintain.
7. Lazy Exception Handling
    i. `org.longbox.businesslogic.controller.AuthenticationController`
        - ➢ While the class does handle exceptions, the error messages are directly set on UI components (`getErrorLabel()` and `getMessageLabel()`). This could be improved by separating the exception handling logic from the UI presentation, potentially using a dedicated error-handling mechanism or logging framework.
    ii. `org.longbox.businesslogic.controller.HomeController`
        - ➢ The exception handling is not optimal, with some exceptions being caught and rethrown as `RuntimeExceptions`. It's better to handle exceptions more gracefully and provide meaningful error messages to users.
    iii. `org.longbox.persistence.stubdatabase.UserStubDb`
        - ➢ The exception handling in methods like `serializeStubData()` and `deserializeStubData()` is not ideal. Instead of throwing a generic `RuntimeException`, it's better to handle specific exceptions or propagate them to the caller.
8. Naming & Scope Pitfall
    i. `org.longbox.businesslogic.controller.AuthenticationController`
        - ➢ String literals such as "Registration Successful", "Username or Email Exists! Please choose a unique username or Email.", etc., are directly set on UI components. It's generally better to use constants or resource bundles for such strings to improve maintainability and localization support.
    ii. `org.longbox.presentation.authentication.AuthenticationFrame`
        - ➢ String literals such as "registration" are used to identify the panels in the

card layout. There should be constants instead of string set in methods directly, especially if the strings get repeated

iii. `org.longbox.domainobjects.entity.User`
   - ➢ The ordering of parameters in the constructors is not consistent, which can make it confusing for developers and increase the likelihood of errors.

iv. `org.longbox.persistence.dao.UserDaoImpl`
   - ➢ Hibernate queries are written in line. This may lead to errors. It is better to have them as stored strings.

9. Dependency Injection Pitfall
   i. `org.longbox.businesslogic.controller.AuthenticationController`
      - ➢ The `AuthenticationController` class directly creates instances of `UserDaoImpl` and `UserService`. This makes the class tightly coupled to these concrete implementations, making it harder to swap implementations or mock dependencies for testing.

   ii. `org.longbox.presentation.authentication.AuthenticationFrame`
      - ➢ The `loginPanel` and `registrationPanel` are directly initialized within the class. This makes the class less flexible and harder to extend if different authentication panels need to be used.

10. Shotgun Surgery
   i. `org.longbox.presentation.authentication.AuthenticationFrame`
      - ➢ The class is designed specifically for managing login and registration panels. If additional authentication flows or panels are needed in the future, the class may need significant modifications.

11. Insufficient Logging
   i. `org.longbox.persistence.dao.UserDaoImpl`

   ii. `org.longbox.persistence.stubdatabase.UserStubDb`
      - ➢ Error logging for all methods is print stack trace for a general error. This can be improved substantially using better and more comprehensive stack traces and by using the logger class

12. Speculative Generality
   i. `org.longbox.persistence.dao.UserDaoImpl`

   ii. `org.longbox.persistence.stubdatabase.UserStubDb`
      - ➢ The `deleteUser()` and `modifyUser()` methods are declared but not implemented. Either implement these methods or remove them if they're not needed.

Mher Eric Gyulumyan

1) Users Have Access To A Menu Of Links To All Aspects of the Home Dashboard/Profile

Relevant Artifacts:

`org.longbox.businesslogic.controller.AuthenticationController`
`org.longbox.businesslogic.controller.HomeController`
`org.longbox.presentation.profile.HomeFrame`
`org.longbox.presentation.profile.SearchPanel`
`org.longbox.presentation.profile.TrendingPanel`
`org.longbox.presentation.profile.FavoritesPanel`
`org.longbox.presentation.profile.ComicRepositoryPanel`
`org.longbox.presentation.profile.AddComicToRepoPanel`

Review

- The HomeController file is overall well made with a few minor code smells
    - Hard coded strings: many of these are present here, especially whe dialog boxes are present. Most of these messages should be stored in variables on the class level, such that they can be easily accessed and changed later in the future id required.
    - Large constructor: the constructor is initializing a lot of the other controllers, if possible, all or some should be initialized at the class level
    - Unused imports: these should be removed since they are not being used
    - Long method: the actionPerformed method is very long. It is understandable that it handles all the interactions, but some of them should be set into functions and called if they are lengthy, like for the profile
    - The favouritesController is declared and instantiated when the favorites button is pressed. A lot of the other controller variables are declared at class level and instantiated at the constructor. There should be more consistency between how controllers are declared and instantiated
- Overall, AuthenticationController is also well made, some small smells persist but nothing that break functionality
    - Unused imports: simply remove anything that is unused
    - Hard coded string: Theses should be stored in variables at the class level for easy access and future modifications
- The file HomeFrame contains a bug, that is that the label userNameLabel is being covered by the button logOutButton. This can be easily fixed by slightly moving the userNameLabel to the left
    - Another smell is unused imports and hard coded strings. Hard coded strings exist for the button names, these strings should be declared as private final strings at the class level so they can be easily accessed and modified in the future

2) Users Can View Their Personal Profile Information

Relevant Artifacts:

`org.longbox.businesslogic.controller.HomeController`
`org.longbox.domainobjects.dto.UserDto`
`org.longbox.domainobjects.entity.User`
`org.longbox.domainobjects.mapper.UserMapper`
`org.longbox.persistence.dao.UserDao`
`org.longbox.persistence.dao.UserDaoImpl`
`org.longbox.persistence.stubdatabase.UserStubDb`
`org.longbox.presentation.profile.ProfilePanel`

Review

- The file ProfilePanel has some major smells
    - Logic in display panel: in this class, two instances of the MouseAdapter class being called as anonymous classes to get the mouseClicked function exist. All logic should be removed from the display panels and be present in their appropriate controller classes
    - Logic in display panel: the reloadTable method is constantly declaring and instantiating a new userDaoImpl variable, as well as readingListDaoImpl and readListDaoImpl. This reload function should not exist within the display panel class, it should be within the controller. Also, these variables can be declared at class level and re-instantiated whenever a reload is required. This goes for all the other variables that get declared in the reload function, since it is reloading values, it does not need to declare them
    - Hard coded string: the buttons with the text “Edit” and “Cancel” should have their titles stored in private final string variables for easier access and future modifications
    - Commenting: the reloadTable method can seem very complex, appropriate commenting to explain what does what is necessary to debug and maintain the code for future iterations
    - Duplicate code: the anonymous classes for MouseAdapter are identical, the logic for them should be stored in a single function that both will access to reduce code duplication
- SInce HomeController is reviewed before, then the same concerns apply
- UserDaoImpl has some slight smells
    - Unused imports: these can be quickly removed
    - Deprecated code: createQuery method is deprecated

3) Users Can Write Comments On Individual Comic Books

Relevant Artifacts:

`org.longbox.presentation.comicbook.ComicBookInfoPanel`

```
org.longbox.businesslogic.service.CommentService
org.longbox.businesslogic.utils.MultiLineCellRenderer
org.longbox.domainobjects.dto.CommentDto
org.longbox.businesslogic.controller.ComicBookInfoController
org.longbox.persistence.dao.CommentDaoImpl
ComicBookInfoController :: ActionPerformed (first if statement)
```

Review

- There are major code smells within ComicBookInfoPanel
    - Logic in GUI panel: many instances of buttons, comments, and star rating logic exist on function within the panel code. All of these should be within their respective controller classes. This class should not be 400 lines of code
    - Hard coded strings: there are some small instances of hardcoded strings, which appear to mainly be on labels. These should be in private final string variables
    - There are some instances of Swing variables being declared and instantiated within the initComicBookPageInfo method, while others are declared on the class level. This inconsistency should be resolved, and have them declared at the class level and instantiated within the initComicBookPageInfo method
- While there are no significant smells in ComicBookInfoController, a lot of the logic that exists in ComicBookInfoPanel should be moved to here and/or its respective service class. Besides this change, all the new code will require proper commenting to entail what each method and complex line does exactly
    - The method ComicBookInforController has a very long method titled actionPerformed. This method can instead have some of it's code transferred to functions, which actionPerformed will call, in order to make it more readable
    - Some variables declared and instantiated within the actionPerformed method can be declared at the class level and then instantiated within the actionPerformed when the desired action is performed

4) Users can View All Comments On Individual Comic Books

Relevant Artifacts:

```
Org.longbox.presentation.comicbook.ComicBookInfoPanel
org.longbox.businesslogic.service.CommentService
org.longbox.businesslogic.utils.MultiLineCellRenderer
org.longbox.domainobjects.dto.CommentDto
ComicBookInfoPanel :: displayComments()
```

Review

- There are major code smells within ComicBookInfoPanel which have been discussed above. All of them apply here as well.
    - For the method displayComments, there should be an error handling for the getCommentsByComic function in case comicBookDTO.getId() is null. A

reattempt of error message in the GUI should be displayed in the event such a thing happens

- The file MultiLineCellRenderer has a few small smells
    - Magic numbers: the number 495 can be private and final and declared and instantiated outside of the function calculateNumLines, as well as for the function setPreferredSize
    - No base case: the function formatDate has no null value check on the input variable ‘date’. This must be accounted for to avoid potential errors
- For the file CommentDto, in the equals method, the method should also compare the commentDate as well. This should be compared because it too is an important factor of a comment, and can be useful during debugging since it is also mention in the toString method as well

Oscar Ye

Code Smells That Have Been Checked:

The code smells that have been checked are from this source:
https://pragmaticways.com/31-code-smells-you-must-know/#Dispensables

1. Dispensables
    a. Unnecessary or unclear comments
    b. Duplicate code
    c. Lazy classes
    d. Dead code
    e. Speculative generality
    f. Oddball solution
2. Bloaters
    a. Large Class
    b. Long Method
    c. Long Parameter List
    d. Primitive Obsession
    e. Data Clumps
3. Abusers
    a. Switch Statements
    b. Temporary Field
    c. Refused Bequest
    d. Alternative Classes with Different Interfaces
    e. Combinatorial Explosion
    f. Conditional Complexity
4. Couplers
    a. Inappropriate Intimacy
    b. Indecent Exposure
    c. Feature Envy
    d. Message Chains
    e. Middle Man
5. Preventers
    a. Divergent Change
    b. Shotgun Surgery
    c. Parallel Inheritance Hierarchies
    d. Solution Sprawl
6. Miscellaneous smells
    a. Inconsistent naming
    b. Uncommunicative Name
    c. Type Embedded in Name
    d. Magic Numbers
    e. Incomplete Library Class

Story 1: When adding a comic book to the repository a user can directly favorite it at the same time

Relevant Artifacts

`Org.longbox.businesslogic.controller.homecontroller`
`org.longbox.presentation.profile.FavoritesPanel`
`org.longbox.persistence.dao.ComicBookFavouritesListDaoImpl`
`org.longbox.presentation.profile.AddComicToRepoPanel`
`org.longbox.businesslogic.controller.FavoritesController`

Review & Suggestions

1. Inconsistent spelling of “favorite” and “favorite”
    a. Suggestion: change all instances of “favorite” to “favorite”
2. HomeController
    a. Large Constructor:
        i. The constructor of HomeController has several lines of code initializing controllers and adding listeners. This might indicate that the class is doing too much.
        ii. Consider breaking down the constructor into smaller methods or delegating the initialization to helper methods.
    b. Repeated Code:
        i. In actionPerformed(ActionEvent e), there are multiple blocks of code that handle button clicks similarly (showing panels, updating tables, etc.).
        ii. This repetition could be extracted into separate methods to reduce duplication.
3. FavouritesPanel
    a. Lack of Encapsulation:
        i. Direct access to fields such as comicBookTable, textField, etc., might violate encapsulation.
        ii. Consider using private fields with getter and setter methods to control access and ensure proper encapsulation.
4. ComicBookFavouritesListDaoImpl
    a. Exception Handling:
        i. There are catch blocks with printStackTrace(), which is not ideal for handling exceptions.
        ii. Consider logging exceptions or throwing custom exceptions with meaningful messages.
    b. Session Management:
        i. Session management (opening, closing sessions) is done manually in methods like saveToFavorites, which can lead to resource leaks.
        ii. Consider using try-with-resources or closing sessions in a finally block to ensure proper session handling.
5. FavouritesController

a. High Coupling:
    i. The controller directly interacts with the FavoritesPanel and UserComicBookCollectionService, which could lead to high coupling.
    ii. High coupling makes the code less flexible and harder to test.
b. Data Flow:
    i. In actionPerformed(ActionEvent e), the logic for handling different button clicks is mixed together.
    ii. Consider separating the logic for each button into separate methods for better clarity.

Story 2: Users can remove a comic book from favorites

Relevant Artifacts

`Org.longbox.businesslogic.controller.homecontroller`
`org.longbox.presentation.profile.FavoritesPanel`
`org.longbox.persistence.dao.ComicBookFavouritesListDaoImpl`
`org.longbox.presentation.profile.AddComicToRepoPanel`
`org.longbox.businesslogic.controller.FavoritesController`

Review & Suggestions

1. There are 3 different ways to favorite a comic and 2 different ways to unfavorite one. It might be too much and convoluted.
    - ➢ Consider reducing the amount of buttons everywhere for a cleaner feel and look. Perhaps only allow favoriting from the specific comic's page and unfavoriting from the favorites page
2. Same code smells in the same classes from the story before

Story 3: Users Can Browse The Full Catalog of Comic Books In The System's Repository

Relevant Artifacts

`org.longbox.businesslogic.controller.ComicRepositoryController`
`org.longbox.presentation.tablemodels.ComicBookTableModel`
`org.longbox.presentation.profile.ComicRepositoryPanel`

Review & Suggestions

1. ComicRepositoryController
    a. UserService is unused bloat code
2. ComicRepositoryPanel
    a. Private string currentItem is declared but unused
3. Otherwise very well made classes as expected from Hashir

Story 4 : Users Can Search For Individual Comic Books By Series Title

Relevant Artifacts

`org.longbox.presentation.profile.ComicRepositoryPanel`
`org.longbox.businesslogic.controller.ComicRepositoryController`
`org.longbox.presentation.tablemodels.ComicBookTableModel`

Review & Suggestions

1. Same code smells in the same classes from the story before