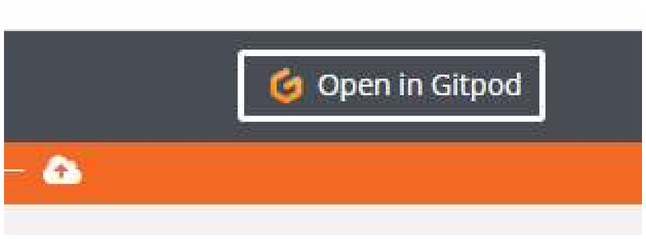
Project Requirements*

Instructions

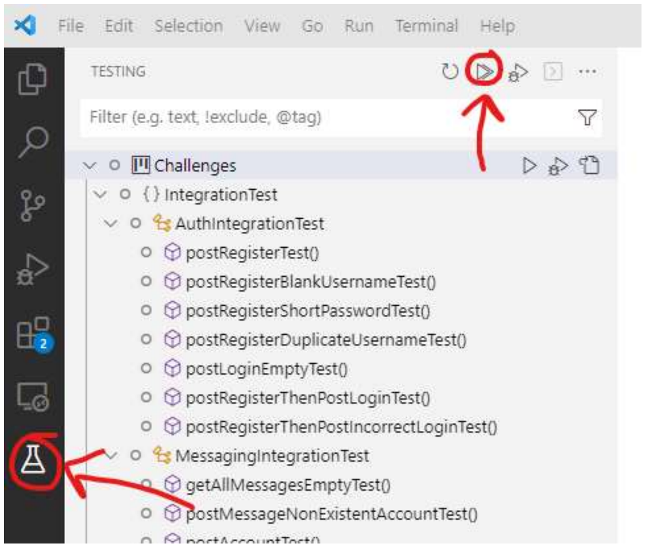
Opening Project

To open the project, please click the "Open in Gitpod" button above. This will automatically open gitpod with the correct project repo containing the project template.



Testing Project

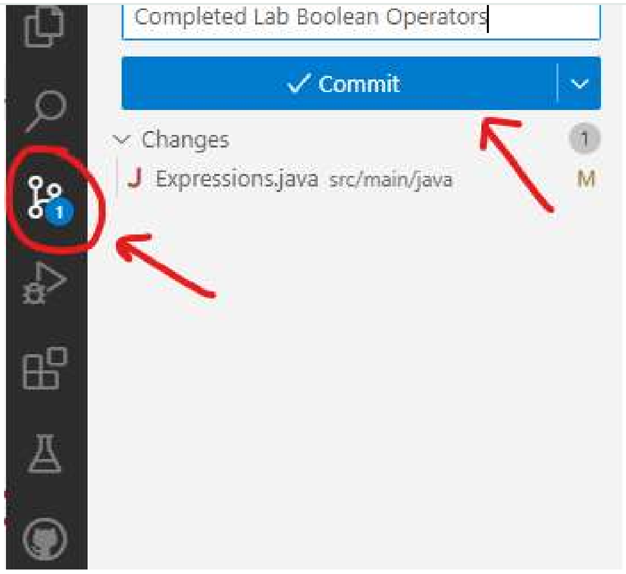
Once the project's workspace has been opened, you can test the project using the graphical test runner as shown below.


File Edit Selection View Go Run Terminal Help
TESTING
Filter (e.g. text, !exclude, @tag)

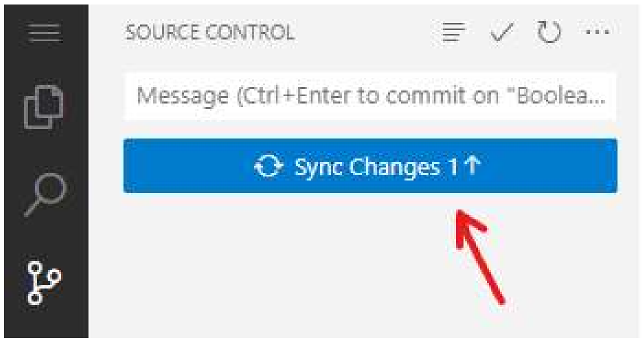
```
Challenges
  { } IntegrationTest
    AuthIntegrationTest
      postRegisterTest()
      postRegisterBlankUsernameTest()
      postRegisterShortPasswordTest()
      postRegisterDuplicateUsernameTest()
      postLoginEmptyTest()
      postRegisterThenPostLoginTest()
      postRegisterThenPostIncorrectLoginTest()
    MessagingIntegrationTest
      getAllMessagesEmptyTest()
      postMessageNonExistentAccountTest()
```

Pushing Changes to Github

When you are finished, you need to commit your changes. Click on the source control button on the left side of the IDE, type in your commit message and then click the blue commit button.

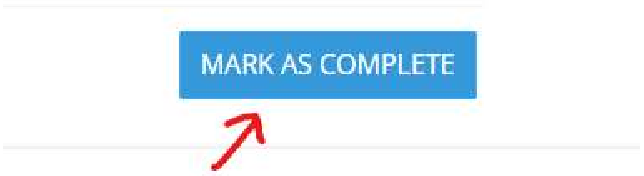

Completed Lab Boolean Operators
✓ Commit
Changes
Expressions.java src/main/java
M

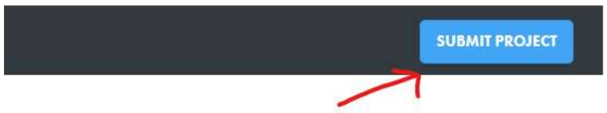
After clicking the blue "Commit" button, the button text will change to say "Sync Changes". Click "Sync Changes" to complete.



Submittal Process

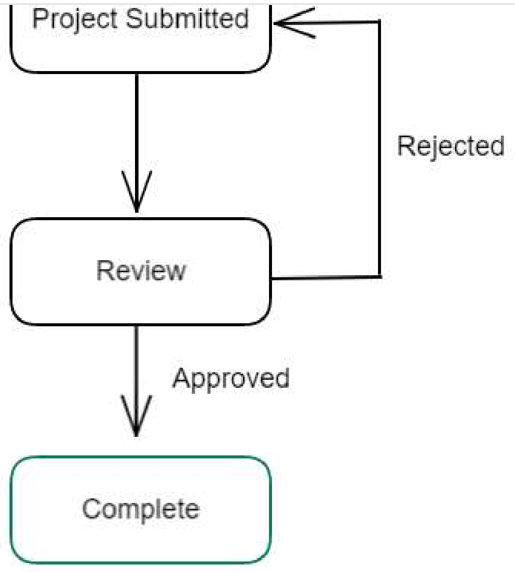
Once you complete the project, you need to submit it for review. To submit the project, you need to click the "MARK AS COMPLETE" button at the bottom of the page and then click the "SUBMIT PROJECT" button (As shown below)





Project Approval Process

Once you open the project in gitpod, your project status will change to "Started". When you start working on the project and running test cases, the status will change to "Incomplete". Once you complete all the user stories and pass all of the test cases, you can click the submit button to change the status to "Submitted". Your project will then be manually reviewed by internal employees.



Coda

See coda link if you would like additional information regarding the project: https://tinyurl.com/final-project-student-guide

Description

Only projects submitted by the due date will be reviewed. Make sure you commit, push and submit your code before then.

Below are the users stories that must be completed:

User Registration

As a user, I should be able to create a new Account on the endpoint POST localhost:8080/register. The body will contain a representation of a JSON Account, but will not contain an account_id.

- The registration will be successful if and only if the username is not blank, the password is at least 4 characters long, and an Account with that username does not already exist. If all these conditions are met, the response body should contain a JSON of the Account, including its account_id. The response status should be 200 OK, which is the default. The new account should be persisted to the database.
- If the registration is not successful due to a duplicate username, the response status should be 409. (Conflict)
- If the registration is not successful for some other reason, the response status should be 400. (Client error)

representation of an Account, not containing an account_id. In the future, this action may generate a Session token to allow the user to securely use the site. We will not worry about this for now.

- The login will be successful if and only if the username and password provided in the request body JSON match a real account existing on the database. If successful, the response body should contain a JSON of the account in the response body, including its account_id. The response status should be 200 OK, which is the default.
- If the login is not successful, the response status should be 401. (Unauthorized)

Create New Message

As a user, I should be able to submit a new post on the endpoint POST localhost:8080/messages. The request body will contain a JSON representation of a message, which should be persisted to the database, but will not contain a message_id.

- The creation of the message will be successful if and only if the message_text is not blank, is under 255 characters, and posted_by refers to a real, existing user. If successful, the response body should contain a JSON of the message, including its message_id. The response status should be 200, which is the default. The new message should be persisted to the database.
- If the creation of the message is not successful, the response status should be 400. (Client error)

Get All Messages

As a user, I should be able to submit a GET request on the endpoint GET localhost:8080/messages.

- The response body should contain a JSON representation of a list containing all messages retrieved from the database. It is expected for the list to simply be empty if there are no messages. The response status should always be 200, which is the default.

Get One Message Given Message Id

As a user, I should be able to submit a GET request on the endpoint GET localhost:8080/messages/{message_id}.

- The response body should contain a JSON representation of the message identified by the message_id. It is expected for the response body to simply be empty if there is no such message. The response status should always be 200, which is the default.

Delete a Message Given Message Id

As a User, I should be able to submit a DELETE request on the endpoint DELETE localhost:8080/messages/{message_id}.

- The deletion of an existing message should remove an existing message from the database. If the message existed, the response body should contain the number of rows updated (1). The response status should be 200, which is the default.
- If the message did not exist, the response status should be 200, but the response body should be empty. This is because the DELETE verb is intended to be idempotent, ie, multiple calls to the DELETE endpoint should respond with the same type of response.

As a user, I should be able to submit a PATCH request on the endpoint PATCH localhost:8080/messages/{message_id}. The request body should contain a new message_text values to replace the message identified by message_id. The request body can not be guaranteed to contain any other information.

- The update of a message should be successful if and only if the message id already exists and the new message_text is not blank and is not over 255 characters. If the update is successful, the response body should contain the number of rows updated (1), and the response status should be 200, which is the default. The message existing on the database should have the updated message_text.
- If the update of the message is not successful for any reason, the response status should be 400. (Client error)

Get All Messages From User Given Account Id

As a user, I should be able to submit a GET request on the endpoint GET localhost:8080/accounts/{account_id}/messages.

- The response body should contain a JSON representation of a list containing all messages posted by a particular user, which is retrieved from the database. It is expected for the list to simply be empty if there are no messages. The response status should always be 200, which is the default

Project leverages Spring Boot Framework

As a developer, I see that the project uses the Spring framework to inject dependencies and autowire functionality using Spring annotations.

MARK AS COMPLETE