Test Cases Developed

User Manipulation Tests

1. **testFindById**: Tests the findById method of the UserService by mocking the UsersRepository to return a specific user when findFirstByUserId is called. It verifies that the expected user is returned.

2. **testFindByUsername**: Tests the findByUsername method of the UserService by mocking the UsersRepository to return a specific user when findFirstByUsername is called. It verifies that the expected user is returned.

3. **testFindByEmail**: Tests the findByEmail method of the UserService by mocking the UsersRepository to return a specific user when findFirstByEmail is called. It verifies that the expected user is returned.

4. **testGetProfilePicBytes**: Tests the getProfilePicBytes method of the UserService by mocking the DropboxService to return a byte array when downloadProfilePicture is called. It verifies that the expected byte array is returned.

5. **testFollow**: Tests the follow method of the UserService by mocking the required objects and verifies that the save method of the FollowRepository is called.

6. **testUnfollow**: Tests the unfollow method of the UserService by mocking the required objects and verifies that the delete method of the FollowRepository is called.

7. **testSave**: Tests the save method of the UserService by verifying that the save method of the UsersRepository is called.

8. **testFindAllByMatchingUsername**: Tests the findAllByMatchingUsername method of the UserService by mocking the UsersRepository to return a list of users when findAllByUsernameContainsIgnoreCase is called. It verifies that the expected list of users is returned.

9. **testChangeProfilePicture**: Tests the changeProfilePicture method of the UserService by mocking the required objects and verifies that the uploadProfilePicture method of the DropboxService is called.

Post Creation Tests

1. **postPointedByFile_WhenValidFileDataAndEnoughPoints_ShouldReturnPost**: Tests the postPointedByFile method of the MakePostService when valid file data and enough points are provided. It mocks the behavior of userService.getNumberOfFollowers and postService.createPost methods to return expected values. It verifies that the expected post is returned.

2. **postPointedByFile_WhenValidFileDataAndNotEnoughPoints_ShouldThrowMakePostException**: Tests the postPointedByFile method of the MakePostService when valid file data is provided but not enough points. It mocks the behavior of userService.getNumberOfFollowers method to return expected values. It verifies that a MakePostException is thrown and userService.save is not called.

3. **postPointedByFile_WhenInvalidFileData_ShouldThrowMakePostException**: Tests the postPointedByFile method of the MakePostService when invalid file data is provided. It checks that a MakePostException is thrown and userService.save isn't called.

4. **postNotPointedByFile_WhenValidFileData_ShouldReturnPost**: Tests the postNotPointedByFile method of the MakePostService when valid file data is provided. It mocks the behavior of postService.createPost method to return an expected post. It verifies that the expected post is returned.

5. **postNotPointedByFile_WhenInvalidFileData_ShouldThrowMakePostException**: Tests the postNotPointedByFile method of the MakePostService when invalid file data is provided. It verifies that a MakePostException is thrown.

6. **postPointedByLink_WhenValidLink_ShouldReturnPost**: Tests the postPointedByLink method of the MakePostService when a valid link is provided. It mocks the behavior of postService.createPost method to return an expected post. It verifies that the expected post is returned and userService.save is called.

7. **postPointedByLink_WhenEmptyLink_ShouldThrowMakePostException**: Tests the postPointedByLink method of the MakePostService when an empty link is provided. It verifies that a MakePostException is thrown and userService.save is not called.

8. **postPointedByLink_WhenInvalidLink_ShouldThrowMakePostException**: Tests the postPointedByLink method of the MakePostService when an invalid link is provided. It verifies that a MakePostException is thrown and userService.save isn't called.

9. **postNotPointedByLink_WhenValidLink_ShouldReturnPost**: Tests the postNotPointedByLink method of the MakePostService when a valid link is provided. It mocks the behavior of postService.createPost method to return an expected post. It verifies that the expected post is returned.

10. **postNotPointedByLink_WhenEmptyLink_ShouldThrowMakePostException**: Tests the postNotPointedByLink method of the MakePostService when an empty link is provided. It verifies that a MakePostException is thrown.

11. **postNotPointedByLink_WhenInvalidLink_ShouldThrowMakePostException**: Tests the postNotPointedByLink method of the MakePostService when an invalid link is provided. It verifies that a MakePostException is thrown.

12. **subtractPointsFromUser_ShouldUpdateUserPoints**: Tests the subtractPointsFromUser method of the MakePostService to ensure that user points are correctly updated. It verifies that the user points are updated and userService.save is called.

13. **checkEnoughPointsForPost_WhenEnoughPoints_ShouldReturnTrue**: Tests the checkEnoughPointsForPost method of the MakePostService when the user has enough points for a post. It mocks the behavior of getNumberOfFollowers method to return an expected value. It verifies that checkEnoughPointsForPost returns true.

14. **checkEnoughPointsForPost_WhenNotEnoughPoints_ShouldReturnFalse**: Tests the checkEnoughPointsForPost method of the MakePostService when the user does not have enough points for a post. It mocks the behavior of getNumberOfFollowers method to return an expected value. It verifies that checkEnoughPointsForPost returns false.

15. **validateFileContent_WhenValidFileData_ShouldNotThrowException**: Tests the validateFileContent method of the MakePostService when valid file data is provided. It verifies that no exception is thrown.

16. **validateFileContent_WhenInvalidFileData_ShouldThrowMakePostException**: Tests the validateFileContent method of the MakePostService when invalid file data is provided. It verifies that a MakePostException is thrown.

17. **manageImageURL_WhenValidLink_ShouldReturnInputStream**: Tests the manageImageURL method of the MakePostService when a valid link is provided. It verifies that the method returns an InputStream containing the expected content.

18. **manageImageURL_WhenEmptyLink_ShouldThrowMakePostException**: Tests the manageImageURL method of the MakePostService when an empty link is provided. It verifies that a MakePostException is thrown.

19. **manageImageURL_WhenInvalidLink_ShouldThrowMakePostException**: Tests the manageImageURL method of the MakePostService when an invalid link is provided. It verifies that a MakePostException is thrown.

20. **getPostCost_ShouldReturnCorrectCost**: Tests the getPostCost method of the MakePostService to ensure that the correct cost is calculated based on the number of followers. It mocks the behavior of userService.getNumberOfFollowers method to return an expected value. It verifies that the correct cost is returned.

Post Interaction Tests

1. **testGetAllUsersLiking**: Tests the getAllUsersLiking method of the InteractionService. It verifies that the method returns a list of users who have liked a post.

2. **testNewLike**: Tests the newLike method of the InteractionService. It verifies that a new like is created for a post and the corresponding user.

3. **testDislike**: Tests the dislike method of the InteractionService. It verifies that a like is removed from a post when the user dislikes it.

4. **testGetAllReposts**: Tests the getAllReposts method of the InteractionService. It verifies that the method returns the count of all reposts for a given post.

5. **testIsReposted**: Tests the isReposted method of the InteractionService. It checks if a post has been reposted by a user.

6. **testPointedRepost**: Tests the pointedRepost method of the InteractionService. It verifies the functionality of reposting with points, which involves updating the points of the repost and relevant users.

7. **testPointedRepost_NotEnoughPoints**: Tests the pointedRepost method of the InteractionService when the user doesn't have enough points to perform the operation. It checks if a PostException is thrown and the user's points remain unchanged.

8. **testNotPointedRepost**: Tests the notPointedRepost method of the InteractionService. It verifies that a not-pointed repost is saved for a post and user.

9. **testPointDistribution**: Tests the pointDistribution method of the InteractionService. It verifies the distribution of points among the reposts and users in a repost chain.