|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Ref | Requirement being tested | Test Content | Input | Output | Pass Criteria |
|  | FR1 | Check that a user can register. | Valid data for the registration form, email, user, and password, such as; [jau1@aber.ac.uk](mailto:jau1@aber.ac.uk), James and Pass1 | User should be redirected to the main page, and the new user should be created in the database. | New user is created in DB, and user is redirected to main page. The information in the database is the same as that entered in the form |
|  | FR1 | If the password given in the confirmation is different do not create a new user. | Valid data for email and user. But the password and password confirmation field should be different. | The user should be shown an error saying that the two password are not the same. | The user is shown an error and no new user is created in the DB. |
|  | FR1 | See if email is registered already | A valid user and password, but the email should be registered before. Such as reusing the email [jau1@aber.ac.uk](mailto:jau1@aber.ac.uk) form test [test ref here] | The user should be shown an error saying that email is in use, and link to recover a password. | Error is shown, and no new user is created in the DB. |
|  | FR1 | See if user can login with valid login information | Valid username and password | User should get a session and be redirected to mainpage. | The user is given a valid session and is redirected to the mainpage. |
|  | FR1 | Invalid login information should be rejected. | A bad username a valid password. Then a valid username with bad password. | The user should see an error saying that his login information is incorrect. | The user is shown an error and no valid session is created for the user |
|  | FR1 | Check that a user can register. | Valid data for the registration form, email, user, and password | User should be redirected to the main page, and the new user should be created in the DB. | New user is created in DB, and user is redirected to main page. |
|  | FR2 | User should see list of their friends upon login. | Log in details.  lwv@aber.ac.uk as the username.  1234pass as the password. Also accounts for the friends are entered into the DB. | List of users friends:  John  Paul. | Shows List of friends. |
|  | FR2 | User should be able to send a friend request by entering an email address. | Enter email: [yap@aber.ac.uk](mailto:yap@aber.ac.uk), where there is a an account registered for this email. | Tell the User that a friend request was sent successfully. | Message saying request sent. |
|  | FR2 | Check that user had entered a vald email address. | Enter a invalid email:  lwv@@dg.cffriu. | Error message warns the user they must enter a valid email. | Warning of invalid email. |
|  | FR2 | Check the user has entered a users email that exists on a Monster Mash Server. | Enter a email not presently used by a user:  llion@me.com. | Error message saying that no such user exists. | Warning that no such user exists. |
|  | FR3 | Check that new window will appear when user hovers mouse over monsters in the monster list. | Hover mouse over single monster from the monster list. | Small window with all the monster’s attributes appears. | All the monster attributes are visible to the user. |
|  | FR3 | When user creates new account, it receives a random monster and small amount of money. | User creates new account. | New account with basic monster and small amount of money is created. | New user has basic monster and small pot of virtual money. |
|  | FR3 | Check that there is some probability that monster can be injured or ill. | Leave server running with some random account for x time. | After x time it is possible that monster is ill or injured. This will be reflected in the monsters attributes. | Monster being ill is added to the attributes and shown to a user. |
|  | FR3 | Check that monster is getting older while server is running. | Leave server running with some random account for x time. | After x time monster is older. | Monster is older and the age attributes is increased and shown in the attributes box. |
|  | FR3 | Check that very old or ill monster can die. | Leave server running for a long time with random account. | After x time monster will die, because it was very old or ill. | Monster dies and is removed from the list, an update message is sent to the user. |
|  | FR3 | Check that there is some probability that monster gets better after illness. | Leave server running with some random account for x time. | After x time monster gets better. | Monster gets better, illness is no-longer displayed as part of its attributes. |
|  | FR4 | To test that a user has the ability to “monster mash” (battle other users with their monsters. | Clicking on a friend and then choosing to battle them. | Some data about the battle should be given out. | That correct data about the battle is sent to both players |
|  | FR4 | To test whether the system accurately assigns the monetary value to monsters and that this is distributed correctly after battle. | Getting two monsters of known value to fight. | The distribution of money to the winning player. | The correct amount of money is added to the winners total. |
|  | FR4 | To test whether a user can send battle requests. | Clicking on another player and choosing to send them a request. | The request received by the other player. | The second player receives the notification and correct data about being challenged. |
|  | FR4 | To test that our ‘battle algorithm’ is working and produces the expected results. | Conducting multiple battles with known attributes, so that the likely outcome is known. | The battle logs. | That although there is an element of chance involved the algorithm is carried out. |
|  | FR4 | To test whether an element of chance is included in the algorithm. | Conducting many battles with monsters of known attributes. | The battle logs. | That in one of the battles a monster with lower attributes that should have lost does in fact win. |
|  | FR5 | User enters an email not registered on any servers an error should be shown. | An email that is not registered. | A message saying the user could not be found. | A message saying that the user is not found and no request is stored. |
|  | FR5 | Whether a user can buy a monster listed on another server. | User buys a monster stored on another server. | Monster has been bought message. | Message that monster has been bought and monster is now stored under the user locally, and users founds are changed. |
|  | FR5 | User cannot buy monsters without the right amount of money. | User buys a monster stored on another server | Error message saying the user has insufficient funds is shown | The buy request is not sent and user is shown an error message. |
|  | FR5 | User can sell monster to friends on other servers. | Friends on a different server buys a monster from our test user | A notification that the monster is sold, to whom and for what. | A notification is shown to the user and the monster is transferred to the other server and removing the local entry. |
|  | FR5 | User’s monster is not sold when the request has insufficient funds. | A friend on a different server buys a monster from our test user while having insufficient funds. | Response to the server sending the request with an exception. | The users monster is not sold and exception is sent to the server sending the request. |
|  | FR5 | Users from other servers can request user data from us by passing a valid email. | Remote server requests the data for a user with a valid email. | Response containing the user data is sent. | Response containing the user data is sent. |
|  | FR5 | A user can breed their monster with users on other servers. | User views the mating list of friends and accepts a mating offer with a user on another server. | The local user should receive some funds and a message saying the “breeding” was successful. | Users receives funds, is shown a notification and the results are stored. |
|  | FR5 | User can list their monsters for breeding so users on other servers can breed with them | User lists a monster for breeding, and a user on another server accepts the offer. | Local user should get a notification with the attributes of the child | User receives notification and the child is stored in users monster list. |
|  | FR5 | User can send fight request to friends on other servers. | User selects a friend and sends a fight request. | The user gets a notification that the request is sent. | Request is sent to remote server, and the user gets a notification. |
|  | FR5 | Remote request is accepted and the monsters fight. | A request has been sent to a remote server. | The output should be a message saying if the user has won/lost with new attributes for the user’s monster. | User should be shown notification, and the result should be stored correctly in the DB. |
|  | FR5 | User remote fight request is cancelled. | A fight request is sent to another user, and then the pending request is cancelled. | No battle occurs, and notification is sent to the user. | A notification detailing the challenge and the subsequent withdrawal is sent to both users |
|  | FR5 | User can accept remote fight requests | User clicks the accept button of a remote fight request. | The output should be a message saying if the user has won/lost with new attributes for the user’s monster. | User should be shown notification, and the result should be stored correctly in the DB. |
|  | FR5 | User can decline remote fight requests. | User clicks the decline button. | Notification should be removed. | No battle is held and the notifications of the users updated. |
|  | FR9 | User should be able to accept a friend request and the friend should be added to their friend list. | The user clicks accept on the friend request. | The friend request should disappear and the friend should appear on the list. | Request disappears and friend appears, and is added to the list of friends in the DB. |
|  | FR9 | When a user’s friend request is accepted the friend should be added to the user’s friend list. | Friend request will be sent and needs to be accepted by the friend. | Friend should appear on the users friend list. | Friend appears on users friend list. |
|  | FR9 | User should be able to decline a friend request and the request should be removed. | The user clicks decline on the friend request. | Friend request should disappear and the friend should not appear on the list. | Request disappears and friend is added. |
|  | FR9 | When a users friend request is declined the rejected request should disappear. | Friend request will be sent and needs to be declined by the friend. | Friend shouldn’t appear on the users friend list and the request should disappear. | Request disappears and no friend is added. |
|  | FR10 | Whether a friend’s monster that was killed, has been removed from their Monster list. | Win a battle against a friend and kill their monster. | Monster should now be removed from the list following the battle. | The monster list for that user is updated in the DB and no longer appears there or is shown to the user. |
|  | FR10 | That a user’s monster was killed and has been removed from their monster list. | Lose a battle, and have the monster killed. | The user’s monster should now be removed from the users monster list. | N/A |
|  | FR10 | Winning a battle will cause the user to gain prize money, and have it added to their account | Win a battle, with prize money. | The user’s money should have increased by the prize money amount. Adding this to their previous total. | N/A |
|  | FR10 | Update the user’s monster after a battle. | Win a battle and receive an injury | Monster should be damaged in some way, following the battle with another monster. | N/A |
|  | FR11 | User should be able to see a list of their friends (Including themselves) with the wealth of each, ordered by wealth. | User should click on the leaderboard. | List of friends and their wealth should appear in order of wealth. | List of friends in order of wealth. |