| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| name: “Jane Doe”, email: “email@mun.ca”, username: “JD2”, password: “password”, student\_id: “1”, gender: “Female”, campus: “stjohns” | A user object in the database with all the respective values for each corresponding variable. | Input restrictions apply. |
|
|

| User | |
| --- | --- |
| Test Case Number | Test Case 1 |
| Test Case Name | Create User WB |
| Test Type (White Box or Black Box) | White box test |
| Test Description | We are creating a user object in the UI and verifying its existence by attempting to access that user’s profile page and attempting to log in as that user. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Register user with all necessary values |
| Step 4 | Login as Username: JD2, Password: password |
| Step 5 | Click “My Profile” |
| Step 6 | Refresh MongoDB Compass |
| Step 7 | Verify values are correct in database |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| A user object in the database with all the respective values for each corresponding variable, with the additional ‘\_\_v = 0’ field that mongodb generates | Pass |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| Load webpage localhost:3000/profile/{id of John Doe} | The profile page for John Doe, reading “John Doe’s Profile” and “John Doe’s Friends” with “No friends to show.” | When logged in as Jane Doe, “+Add Friend” button should also be present. |
|
|

| User | |
| --- | --- |
| Test Case Number | Test Case 2 |
| Test Case Name | View Another Profile |
| Test Type (White Box or Black Box) | Black box test |
| Test Description | We are accessing the profile of John Doe while logged out, and then again while logged in as Jane Doe. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000/profile/{id of John Doe} |
| Step 3 | Click “Login” |
| Step 4 | Login as Username: JD2, Password: password |
| Step 5 | Load webpage localhost:3000/profile/{id of John Doe} |
| Step 6 | Verify changes to page in both circumstances (logged in as different user, logged out |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| The profile page for John Doe, reading “John Doe’s Profile” and “John Doe’s Friends” with “No friends to show.” | Pass |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| Click “+Add Friend”, Logout, Login as John Doe, click “My Profile”, accept friend request | After requesting, both users should now have an attribute “friends”, with one object, whose id is identical to the id of the user they are meant to be friends with. | After requesting, status should be “requested” for John Doe in Jane Doe’s friends and “pending” for vice versa. After accepting, it should be “accepted” for both. |
|
|

| User | |
| --- | --- |
| Test Case Number | Test Case 3 |
| Test Case Name | Request Friend/Add Friend |
| Test Type (White Box or Black Box) | White Box test |
| Test Description | We are logging in as Jane Doe and attempting to add John Doe as a friend. Before accepting the friend request while logged in as John Doe, we are verifying the objects in the database. Finally, we will accept the friend request as John Doe and again verify the object in the database. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Login as Username: JD2, Password: password |
| Step 2 | Load webpage localhost:3000/profile/{id of John Doe} |
| Step 3 | Click “+Add Friend” |
| Step 4 | Verify values are correct in database |
| Step 5 | Click “My Profile” |
| Step 6 | Refresh MongoDB Compass. |
| Step 7 | Verify values are correct in database |
| Step 8 | Logout. |
| Step 9 | Login as Username: JD1, Password: password |
| Step 10 | Click “My Profile” |
| Step 11 | Click “+Add Friend” under “My Friend Requests” |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| Both users now have an attribute “friends”, with one object, whose id is identical to the id of the user they are meant to be friends with. | Pass |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| userId: “a string”, author: “John Smith”, text: “This is a post.”, date: “This is the date.”, image: “0”, visible: “0” | A post object is created in the database with all the respective values for each corresponding variable. | No input restrictions apply. |
|
|

| Post | |
| --- | --- |
| Test Case Number | Test Case 4 |
| Test Case Name | Create Post WB |
| Test Type (White Box or Black Box) | White box test |
| Test Description | We are creating a post in the database and verifying its existence by logging in as John Doe and viewing it on the dashboard. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Open MongoDB Compass |
| Step 2 | Access database “4770TeamProject” |
| Step 3 | Access schema “posts” |
| Step 4 | Click “INSERT DOCUMENT” |
| Step 5 | Insert all necessary variable names and values |
| Step 6 | Click “INSERT” |
| Step 7 | Console node app.js |
| Step 8 | Load webpage localhost:3000 |
| Step 9 | Login as Username: JD1, Password: password |
| Step 10 | View post and verify its values |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| A post object with ‘John Smith’ as the author, with date ‘This is the date’ and text ‘This is a post’ | Pass |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| text: “This is a post.” | A post object in the database with all the respective values for each corresponding variable. | All other inputs from Test Case 5 are automatic. userId will be John Doe’s user id, the date will be the current date, etc. |
|
|

| Post | |
| --- | --- |
| Test Case Number | Test Case 5 |
| Test Case Name | Create Post WB |
| Test Type (White Box or Black Box) | White box test |
| Test Description | We are creating a post object in the UI, while logged in as John Doe, and verifying its existence by viewing it on the dashboard. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Create Post” |
| Step 5 | Enter text and click “Post” |
| Step 6 | Verify values are correct in database |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| A post object created in the database with the text “This is a post.” | Pass |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| text: “This is an edited post.” | The post will be changed. When logged in as Jane Doe, we will be unable to edit John Doe’s post. |  |
|
|

| Post | |
| --- | --- |
| Test Case Number | Test Case 6 |
| Test Case Name | Edit Post |
| Test Type (White Box or Black Box) | Black box test |
| Test Description | We are editing the text of a post we already created as John Doe. We will then login as Jane Doe and verify that we cannot edit that post. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Edit Post” on post from Test Case 6 |
| Step 5 | Enter text and click “Post” |
| Step 6 |  |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| The post is updated with the new text. | Pass |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| text: “This is another comment.” | A comment object in the database with all the respective values for each corresponding variable. | This comment should also be editable by John Doe after it has been posted. Again, unlike Test Case 8, all inputs other than text are automatic. |
|
|

| Comment | |
| --- | --- |
| Test Case Number | Test Case 7 |
| Test Case Name | Create Comment WB |
| Test Type (White Box or Black Box) | White box test |
| Test Description | We are creating a comment object in the UI, while logged in as John Doe, on the post created in Test Case 6, and verifying its existence by viewing it on the dashboard. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Add a Comment” |
| Step 5 | Enter text and click “Post” |
| Step 6 | Verify values are correct in database |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| A comment object is created and added to the database. | Pass |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| text: “This is an edited comment” | The comment will be changed. |  |
|
|

| Comment | |
| --- | --- |
| Test Case Number | Test Case 8 |
| Test Case Name | Edit Comment |
| Test Type (White Box or Black Box) | Black box test |
| Test Description | We are editing the text of a comment we already created as John Doe. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Edit Comment” on comment from Test Case 9 |
| Step 5 | Enter text and click “Post” |
| Step 6 |  |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| The comment text is changed to the entered text. | Pass |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| text: “This is a temporary comment.” | A comment object is created in the database with all the respective values for each corresponding variable, and then subsequently removed upon deletion. |  |
|
|

| Comment | |
| --- | --- |
| Test Case Number | Test Case 9 |
| Test Case Name | Delete Comment |
| Test Type (White Box or Black Box) | White Box test |
| Test Description | We are creating a new comment while logged in as John Doe, on the same post created in Test Case 6, and then subsequently deleting it. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Add a Comment” |
| Step 5 | Enter text and click “Post” |
| Step 6 | Verify values are correct in database |
| Step 7 | Click “Delete Comment” |
| Step 8 | Verify values are correct in database |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| The comment is deleted and removed from the database. | Pass. |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| Course name: “Math 1000”, Monday box checked, Monday time: “2-3pm”, Course slot: “2” | A schedule object is created in the database with all the respective values for each corresponding variable. |  |
|
|

| Schedule | |
| --- | --- |
| Test Case Number | Test Case 10 |
| Test Case Name | Create Course |
| Test Type (White Box or Black Box) | White Box test |
| Test Description | We are creating a new course while logged in as John Doe. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Schedule” from the navigation menu |
| Step 5 | Enter the variables and click “Submit” |
| Step 6 | Verify values are correct in database |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| The schedule is created and added to the database. | Pass. |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| N/A | The schedule object is deleted from the database. |  |
|
|

| Schedule | |
| --- | --- |
| Test Case Number | Test Case 11 |
| Test Case Name | Delete Course |
| Test Type (White Box or Black Box) | White Box test |
| Test Description | We are deleting the course we created in the last test, while logged in as John Doe. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Schedule” from the navigation menu |
| Step 5 | Click “Delete” on the course to delete. |
| Step 6 | Verify the schedule object has been removed from the database. |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| The schedule is deleted and removed from the database. | Pass. |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| Course Name: “example” | A poll object is added to the database. |  |
|
|

| Poll | |
| --- | --- |
| Test Case Number | Test Case 12 |
| Test Case Name | Create Poll |
| Test Type (White Box or Black Box) | White Box test |
| Test Description | We are creating a new poll while logged in as John Doe. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Poll” from the navigation menu |
| Step 5 | Enter the course name and click “Submit“ |
| Step 6 | Verify the poll object has been added to the database |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| A poll object with the enter course name is created and added to the database. | Pass. |
|
|

| Specifications | | |
| --- | --- | --- |
| Input | Expected Output | Notes |
| “3 Stars” radio button is selected. | A vote is added to the array of the poll object in the database. |  |
|
|

| Poll | |
| --- | --- |
| Test Case Number | Test Case 13 |
| Test Case Name | Vote on a Poll |
| Test Type (White Box or Black Box) | Black Box test |
| Test Description | We are voting on the poll created in the last use case. |
|
|

| Procedural Steps | |
| --- | --- |
| Step 1 | Console node app.js |
| Step 2 | Load webpage localhost:3000 |
| Step 3 | Login as Username: JD1, Password: password |
| Step 4 | Click “Poll” from the navigation menu |
| Step 5 | Select the “3 Stars” button and click “Vote“ |
| Step 6 | Verify the poll object has been added to the database |
| Step 7 |  |
| Step 8 |  |
| Step 9 |  |
| Step 10 |  |

|  |  |
| --- | --- |
| Actual Output | Pass/Fail |
| The poll object’s vote array is updated in the database to include the new vote. | Pass. |
|
|