**1 – Test Case Exercise.**

Expect this exercise to take no more than 15-30 mins

Evernote, (https://evernote.com/), a month away from their “Go LIVE” date realized that although they had written some automated tests they were still unclear about the quality of their product, they decided to hire QA. The available functionalities at the time you joined were just the following:

Ability for a single user to share their notes across multiple different devices, BUT no document sharing between different users. Please consider the entire stack when answering.

1. write test cases for the application, no need to write out all similar type cases, just make note that there are more, eg repeat for each XYZ. Expecting a high level listing of areas to test not detailed Test Cases.
2. specify what you would do first to get an idea of how buggy the application might be

1)

1. Verify user1 can add note

2. Verify user1 can see previously created note

3. Verify user1 can edit previously created note

4. Verify user1 can see a note after closing browser and reopening

5. Verify user1 can see a note after log out / login

6. Verify user1 can see note after logging in on Mac, Windows, android browser, iPhone browser, android native app, iPhone native app

7. Verify user1 can see notes of different types (text, image, formatted text, html, etc)

8. Verify user1 can create multiple notes and see all on other device

9. Verify user2 can not see user1 note after user1 has logged out

10. Verify user1 can edit note on device 2 and changes are 'immediately' visible on same note on device1

- Verify that notes can be printed successfully

11. Verify user1 is required to enter a password

12. Verify if password is omitted, the user can not sign in

13. Verify that a dummy password is not accepted (e.g. "root")

14. Verify that the password and sign-in errors reveal no data

15. Create IDs with 10, 100, 1000, 10000 notes

- verify notes load for user on multiple devices

- verify notes can be edited for user on any device/session

- measure ux load time for high-note-volume users (socialize to determine whether or not the performance is acceptable)

16. Create notes with the following content, "select \* from users", "select \* from notes"

17. User1 load a note, observe the URL (presumably something like, https://www.evernote.com/user/12345/note/1

Change the URL to something like, https://www.evernote.com/user/1/note/1 - verify that notes from other users do not load

18. Scenario 1) Create 100 users, Scenario 2) create 1000 users, Scenario 3) create 10,000 users Scenario 4) create 100,000 users

- at each scenario, create a variety of users with 1, 10, 100 , 1000 notes

- measure performance in UX

- measure server CPU load and response times

- measure DB CPU load and response times

- measure 'web server' / access log for response times

- review app server & DB error logs

19. repeat test 18 scenarios with notes of different conten types (e.g. HTML, long notes, images, etc.)

20. a) Create 100 users, create 1, 10, 100, 1000 notes for some users

b) Create new notes for various users at a rate of 1 note per 5 minutes per user - measure performance and errors

c) Create new notes for various users at a rate of 1 note per minute per user - measure performance and errors

c) Continue with these tests until the infrastructure breaking point is observed

21. Verify 'Evernote' is spelled correctly everywhere in UI

- verify Evernote logo is displayed correctly on all devices, browsers and screen sizes

22. Verify all other text is spelled correctly in UI

23. Verify Evernote spellchecker catches spelling errors.

24. Verify browser based spellchecker works or is disabled so that Evernote spellchecker works

25. Verify all error alerts have valid, informative error messages

26. Shut down master DB, verify users can still load, create, view, delete notes

- bring the Master DB back up - verify all edits made during primary outage are still acesible by user

27. Shut down 1 app server, verify users can still load, create, view, delete notes (session should failover to other app server)

28. Find out what the CEO’s per-peeve bugs are – verify that they do not occur 😊

**Specify what you would do first to get an idea of how buggy the application might be**

1. Execute the testcases in response 1 in the order entered.
2. Have everyone else on team (HR, product, customer support) do adhoc testing