First Part

Application Chosen: Linkedin

Test Cases:

- **Registration:**

---

1. A positive scenario by inserting a valid e-mail and password of 6 characters (restricted by Linkedin for first time password).
2. A negative scenario by inserting a valid e-mail and a password of less than 6 characters
3. A negative scenario by inserting a invalid e-mail (eg. 123@h0tmail.com)and a valid password of 6 characters.
4. Verify that an e-mail can not be used to register for more than one account.

- **Login:**

---

1. A positive scenario by using a valid created e-mail and its correct password.
2. A negative scenario by using a valid created e-mail and trying a wrong password.
3. Verify that Uppercase characters are not treated the same as Lowercase ones.
4. Verify that passwords copy and paste is not allowed.
5. Verify that an e-mail should be sent to the user when his account is logged in using an unusual device.
6. Verify that after 4 or 5 wrong password attempts, an e-mail should be sent to the user.
7. Verify that after 4 or 5 wrong password attempts, login should be prevented for this account for a specific time.

- **Messaging:**

---

1. A positive scenario by trying to send a message to more than one receiver, a body text, insertion of emojis, attached documents, insertion of photos, insertion of GIF and sending the message.
2. A positive scenario by trying to sending a voice note to one ore more receiver.
3. Verify that it is not possible to send an empty message to a another contact on Linkedin.
4. Verify that it is not possible to send a text message to an empty recipient.

- **Posts:**

---

1. Verify that all types of posts: photo, document, poll, celebration ....etc are working.
2. Verify that it is not possible to post an empty post.

- **Log out:**

---

1. Verify that logging out from all devices can be done.

Search:

---

1. Verify that when getting the search box, recent searches appears and are those of that account only.
2. Verify that Clear button clears all recent searches even after log out and log in again.
3. Verify that search results are relevant to search keyword.
4. Verify that total number of search results is displayed somewhere in the page.

Prioritization:

The test cases should be arranged according to its business impact as follows:

1. Security tests related to Login and Logout should come first.
2. Messaging tests as they ensure communication between users is working efficiently.
3. Posts and Search tests are as users need for their business and sharing experience or asking for help so they should be working well.
4. Registration tests as they ensure easy joining to Linkedin.

Second Part

Bugs Found

1. On attempting to login with wrong password for many times (eg. 5 or more times), Linkedin doesn't send an e-mail to the user to warn him against a possible attack. **(High Priority, Critical Severity)**
2. On attempting to login with wrong password for many times (eg. 5 or more times), login trials are not stopped for the account. **(High Priority, Critical Severity)**
3. Copy and Paste for passwords from text box is allowed.**(Medium Priority, Major Severity)**
4. When using the search engine by using symbols "{¥", some results appeared which are not related to the keyword inserted. **(Medium Priority, Minor Severity)**
5. When using the search engine, no number for found results appears on the page. **(Low Priority, Cosmetic Severity)**
6. There is no option to log out of the same account form all other devices.**(Medium Priority, Moderate Severity)**

All these bugs were found on the application on both iOS (Iphone X, IPad) and Android (Samsung S4, Samsung A5) platforms.

Fourth Part

Test Cases for Best-Buy API

For Every one of the objects (Products, Stores, Services, and Categories), the following test cases are to be done:

1. POST a new One Object --> Expect Positive Response.
2. GET the new Object with its ID --> Expect Positive Response.
3. PATCH the new Object by changing one of its parameters --> Expect Positive Response.
4. DELETE the new Object with its ID --> Expect Positive Response.
5. GET the deleted object with its ID --> Expect Negative Response.
6. Delete the deleted object with its ID --> Expect Negative Response.
7. PATCH the deleted object with its ID --> Expect Negative Response.
8. POST a new Object with missing variables --> Expect Negative Response.
9. GET the Objects with limit and skip --> Expect Positive Response.

For the utilities (version, and healthcheck), a test case is to be done:

1. GET for each utility --> Expect Positive Response.

These test cases are created using POSTMAN Collection written using JSON.