**Depop Login – Test Strategy/Plan/Report All in one**

**Requirements (Basis for testing)**

Depop website should have login functionality for users to be able to manage their own profile.

**Scope**

Scope of testing is limited to login functionality on login page of Depop website <https://www.depop.com/>.

Below are the key functions that are in scope for login functionality

* User is able to login with valid login credentials
* User is not able to login with invalid login details (can add extra numbers or special characters or blank spaces to make them invalid)
* User is shown error message when login details are invalid
* User is shown correct error message when username and password fields are left empty

**Not in Scope**

Every link other than related to login feature is out of scope from Depop website for this technical testing task. For example, below are out of scope for this test technical task

* Facebook button
* Sign up button
* Menu link or any other links that are visible

**Test Phases**

Below are the various test levels that will be performed for the Depop login feature technical task.

**Integration Testing**

* API testing to be done to verify the login system API responds correctly for various inputs.

**Feature Testing**

* When front end is designed, then login feature testing can be performed.
* Usability of the feature is tested on this occasion.

**Automation Testing**

Due to the limitations of the login attempts for login only 5 input validations are done for automation.

Feature file is pretty self-explanatory as in to understand what is being tested.

Parameterising has been done in order not to duplicate code.

Asserts can be found in Depop\_Login.java

**Automation Testing Report**

* Automation of the key features of this login application is completed (the automation work is uploaded in github).
* Below test methodologies are used in automation framework
  + BDD Agile methodologies, having feature files
  + Selenium Webdriver
  + Cucumber
  + Java
* Automation test principles
  + Each test has a concise verification (as atomic as possible)
  + README.md file has all the details including how to run the test.
  + CI/CD – Scheduling automation run every time a new application build is deployed will help to identify the issues early.
* Test Automation Report
  + Report can be found within the automation project. Instructions on how to open the report are given in project README.md.
  + Report will be updated with new results every time you run the tests.

**Exploratory Testing**

This is a type of testing where test cases are not created in advance but testers check system on the fly.

* Enter empty username, password or both and click login and check behaviour (application throws the appropriate the error)
* Verify what happens when page is refreshed after logging in (user is still logged in as expected)
* Click back button on browser and check that if login still persists (when clicked back button, it shows the login screen but eventually shows that user has logged in)

**Cross browser/Cross platform Testing**

* Desktop
* Device Android
* Device iOS
* Mac
* Windows
* Different browsers including Chrome, Firefox, Internet Explorer and etc

**Regression Testing**

* As this is the first iteration of testing, regression is not in the scope here. If any changes are made to this application due to bug fixes or functional enhancements, a set of tests need to be executed to ensure core existing functionality is not affected due to the changes introduced.

**End to End Testing**

* Usually this type of testing is performed after completing system testing where multiple systems with different interfaces are integrated. In this case, if API testing is performed initially to test the backend for login in large extent, then front end testing will have covered majority of the rest of the usability perspective.

**Security Testing**

* Session management: Once user is on logged in page, it is never expiring the session. This is just an observation.
* We are able to generate the below error as part of security testing
  + 400 invalid request
* We could not generate the below errors as part of security testing
  + 408 request time-out
  + 500 server error
* As there is not any access management, security testing in access to the application perspective is not done.
* Other security testing methods like brute force attack and SQL injection are not carried out as they are not applicable here.

**Performance Testing – Load and Stress**

* Load testing should be carried out to observe the application response times when it is under the expected load (For this task, as per login limitations couldn’t perform much)
* Stress testing is performed to verify the application behaviour under increased load (beyond usual expected load) to observe response times, how application can crash and recover, how failover and disaster recovery can be done. (Again, due to limitations of login attempts limit is required to be increased in order to perform this).

**Test Cases**

Should be in a very detailed format (If possible, get a test management tool like rally, testrail, etc) – Due to the time constraints, writing at high level.

However, a “Good Test Case” will have below:

* Preconditions => including test data creation, accessing environments, any required information to be able to execute the test
* Summary of test => Brief outline where you know what test is all about
* Steps for action => Detailed step wise action plan to execute the test case
* Expected Outcome => What is being expected of the action
* Any test data into the test case that any tester can pick up the test after say 6 months
* Any wireframe attachments that may be used for better understanding at the time of test

Tests below has a summary and are laid out as

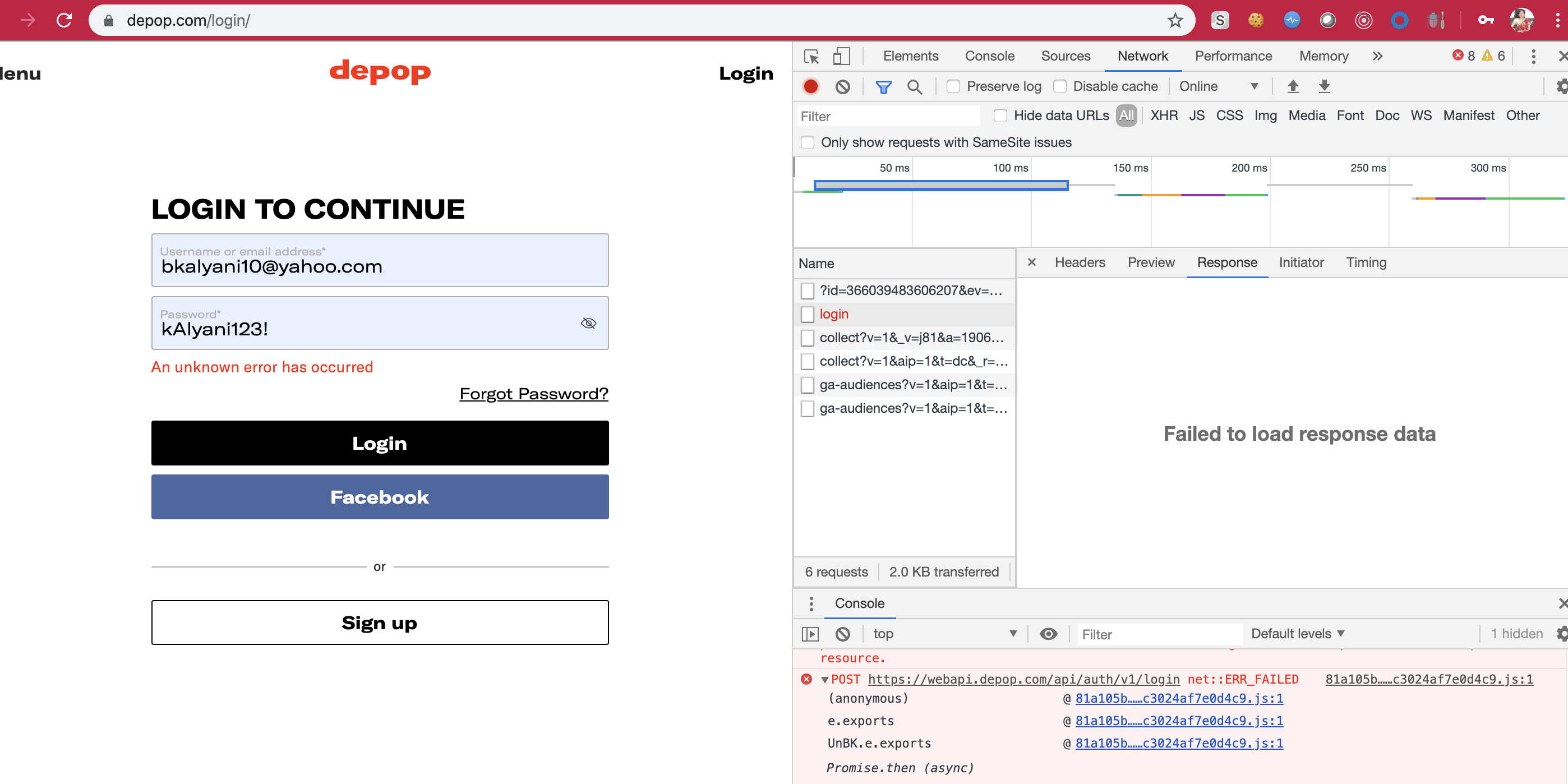
<test case/scenario> - <an expected outcome> (Defect: if found)

Automated Test Scenarios;

1. Logging with correct username and password text fields – user logs in
2. Logging with correct email and password text fields – user logs in
3. Logging with incorrect email and password text fields – error message “invalid username and password” shown for user
4. Logging with incorrect username and password fields – error message “invalid username and password” shown for user
5. Click login with empty username – error message “username is required” is shown for user
6. Click login with empty password and correct username – error message “password is required” is shown for user

Exploratory Test Scenarios

1. After user is logged in refresh the page - logged in session still persists
2. After user is logged in click back button on browser – logged in session still persists
3. Enter just numbers and click Login – error message “Invalid username and password” and sometimes “unknown error has occured” like in below screeshot
4. Enter just special characters and click Login – “unknown error has occured” like in below screeshot
5. Enter correct username and wrong password and click login a few times – Error message appears “An unknown error has occured” (Defect: User is left confused of what is happening. It just locks options even after user enters correct password) Screen shot below. There is a situation where user is thrown error message “unfortunately user is locked for xx hours” but errors are not handled properly I believe on this occasion.



And many more tests can be designed depending on the requirements or user story acceptance criteria.

**Defects**

Ideally bugs to be raised in tools such as jira, Bugzilla, etc where user stories exist.

However, a good defect should consist of the below:

* Summary of defect => a brief description as to understand what it is about
* Steps to Reproduce => A detailed step by step reproducible steps for a developer or any other team member to understand and reproduce
* Expected outcome => What was expected
* Actual Outcome => What is the outcome and why you think it is not correct
* Screenshots => Any screenshot that may help for more understanding of the defect
* Data => Any data that can be used to reproduce the defect

**Acceptance criteria**

Entry criteria

* Login backend implementation is done.
* Unit tests are covered.
* Login front end website page is designed.
* Environments are available to deploy the feature to be tested.

Exit criteria

* Login functionality works as per the requirements.
* No high or medium priority defects are outstanding.
* Defect triage is done and little low priority defects are allowed to have signed off.

**Assumptions and limitations**

* In the absence of exact requirement use cases, business requirements and functional requirements I assumed the generic application behaviour and written the high-level test cases.
* The expected results of the above test cases are based on generic assumption of how login feature behaviour.
* It’s assumed most of load, stress and non-functional testing is not in the scope of this assignment.
* The severity and priority of defects is not defined and it’s assumed that these defects will be triaged later.
* Due to the limitation of the login feature of only able to be attempt 10 logins each hour, not been able to add many automation scenarios.
* Due to the limitation of login errors, throwing different errors, when continuous invalid login attempts are made, not been able to test scenarios as the login capability is pushed to hours to not been able to login upto 17 hours.
* Due to time constraints and lack of formal requirements, the test strategy is not fully comprehensive.

**Residual Risks**

* As there are no load testing requirements, and limitations of the login attempts login implementation may need to be tweaked to be able to perform load testing.

**Recommendations**

* If the defects are found, they will need to be triaged with business before agreeing whether this application is fit for business purpose.