Section 3
Test Case
Uri Segman, QA Automation Home exam.

When trying to assess the quality of a product such as the P2P of Mirotalk we must think of ourselves as the user, and try to locate the aspects that do not work that will most affect the user experience. Once we have defined these possible points of failure we need to devise a test and a method of scoring the test in order to record and measure the results of the test.

To begin with I will give an overview of the product of Mirocast as a first time user, I will then outline what I think are the most critical points in the user experience of the Mirotalk product, and then discuss possible test cases. Finally I will look into one test case and consider a method of solving it.

When looking at a product and attempting to test it effectively we must first understand and outline the most important pain points of the user. In the case of the video-conference product of MiroTalk, the most important aspects I would consider a user to care about fall into three categories: Ease of Use (is the products functionality accessible, readable, and easy to use), do the functions actually work (what are their limitations, where are their failure points), and timeliness, as no functionality no matter how good, is worth waiting for too long.

Generally speaking the MiroTalk product seems easy to understand. The home page is straight forward, and the journey from google search to video call is quick and seamless, and requires no sign-in. The video call from my short experience seems strong and of good quality. To start off, the MiroTalk video call product looks great. Once inside the video call the user has a variety of functions it can use, from chatting with chat-gpt on the conference chat, sending files, sharing screens, and even a whiteboard that can be used to write notes in for all to see in the call. In terms of general points to consider, the pain points I experienced when working with this product is long wait times for calls to begin and participants to join (timeliness), it took a long time for selenium to load things, and buttons could appear and disappear randomly. Nevertheless the Ease of use was good apart from the fact that the sidebar would disappear and reappear arbitrarily which can be annoying. In terms of functionality I noticed that the clickable icons at the top right of the screen did not work. Yet all in all the functionality and robustness of the product seemed very good and effective.

When considering the potential test to run on this product, in terms of QA automation, I would focus on two things: timeliness and functionality. In terms of ease of use, this can be better attributed to UX/UI and design, and require different forms of tests involving real users ie. a/b testing. In terms of the quality of the product, I would begin by looking at if each functionality works. Ie. Does the whiteboard work? Screen share works? When a participant raises their hand do all other participants see it? Does private messages work effectively?

Assuming the functions work in ‘comfortable’ settings I would look into each functionality and consider its performance under pressure, measuring performance metrics such as load times and response rates. For example In the case of adding a participant, I would look into how long it would take to add the 10th participant, the 100th. At what limit will the call fail? What would be the response times of other functions like the chat box or file_sharing in a call with 40 participants? Real testing should be done under pressure, as under pressure faults can be detected.

The method by which I would run a pressure test on the product, would follow this specific rule: record everything in a csv/text file. The main data points I would focus on collecting are load times (I do not know how I could do this with selenium, but ideally I would focus on this), and any expected output from a function should be recorded as a screenshot or text. In this specific case, assuming I want to check the functionality of the whiteboard under pressure.

To begin with I would need to run the code on a collection of computers therefore I would use a Lambda function that could be triggered that would add a participant. Each lambda would iteratively collect metrics on the white board sharing as a (timestamp, screenshot, participant_name, other_relevant_parameters) tuple, adding the recordings to a database. On the main participant (the one that is sharing the screen) I would perform arbitrary sets of acts on the whiteboard, at set time intervals. Each action would be recorded as a tuple of (timestamp, action_screenshot). Once built I would do this test with an incremental number of participants.

In terms of reading the data and scoring the performance of the product, this could be very challenging yet very possible. I would take the data points and group them into a time bracket of +x_time from a given action screenshot (the time is what we would consider a fair delay). Then each participant's screenshot would be compared to the screenshot of the action, and would be scored by the pixle_deltas between the two screenshots. This would represent the load time delays. These metrics would then be loaded into a CSV, and available to be analyzed relative to other parameters that could be relevant.

Potential pitfalls:

- Such a large scale QA project might not be worth it, but if done effectively it could be modular and used to test any new functionality added.
- Dealing with selenium exceptions (Selenium has many fail points, and often crashes as it depends on factors such as internet speed) making it very difficult to run selenium code consistently (from my experience).
- Measuring the deltas in pixels could be dependent upon screen size and window dimensions. A potential solution is finding a distance computation algorithm to calculate distance between two images of different dimensions.

In conclusion, the idea of making a large test would enable us to measure new features under pressure and truly get an understanding of their performance before entering production. The use of automation to perform QA tasks can truly be effective when done in scale. That being said, any QA test can only be effective when the target metrics are clearly defined and recorded in an easy to use manner that can be later analyzed and queried to measure improvement and locate failure points effectively.