Brief report summarising your findings, observations, and any challenges :

Problem 1)

Approach :
1.observed happy path for the given flow of test steps
2.identified 3 pages , planned to create POM model
3,intellij idea +JAVA + maven PROJECT created
4.created a framework to display wholesomeness in approach , added necessary dependencies from maven repository
5.created basic utils , browser factory, implemented exception as per the question and extent report classes
6.created pages for page object model , planned locator strategy and identified elements and methods and maintained abstraction
7.created a login test class to use all the three pages and made necessary assertion , implemented wait as per the assignment
8.implemented extent report on login test
9.executed test cases , documented extent report
10.staged repo and applied commits to the repo

Challenges :
1.selection of dependencies and setting up of test environment , in prod we use idk 11, selenium 3.6
2.took up challenge to use idk 20, recent version of selenium with less issues
3.some package named conventions

Resolutions :
1.executed maven rebuild to sync the dependencies
2.minimally Refractored the script based on debugging the flow

Findings :
1.all the test steps ran successfully and printed them to extent report
2.navigate page >executed steps > logged into secure area > logged out

Problem 2)
Approach 1:
1.created collection with get and post requests
2.created collection level variables and executed the test cases for json format validation and response codes
3.created a post request sent a json payload , the status code was successful , but tried to check if another get request would reflect the changes

Challenges :
1.the updated json payload through always returning with same id , which should not happen
2..variables were not set at environment level limited my approach to automate steps 1 and 2

Resolution :
1.ran collection for 5 iterations to see changes

Approach 2 :
1.set up an environment , created collection in it
2.declared environment variables
3.accessed the response of one request in another request
4.added a put request to confirm the identified issue , wrote tests accordingly
5.created a curl document

Findings :
1.the end point has 100 values in json , and when post request is created a new element is created with id 101 , however even after doing several modifications made still 101 is only reflected (attaching necessary documents regarding it)
2.automated step1 and step 2
3.for get and post tests ran successfully

Problem 3)
Approach :
1.installed jmeter
2.set thread count and condition for concurrent users
3.added a http request
4.added different listeners to match the necessary information regarding performance of the end point
5.generated summary report , aggregate report , response time graph , view results table

Challenges :
1.setting up test environment in macOS .

Resolution :
1.Modified homebrew path and reinstalled and set up necessary profile changes

Findings :

1.In prod usual accepted standard is around 400ms , many times the avg ,median were more than 500
2.the endpoint is slow to handle 100 concurrent users , the server needs to be ramped up .
From summary report
3.**Average response time is 1167ms**
4.minimum is 0ms , max response is 4790ms (high deviation)
5.the **error percentage is 0%**
6.the 90% line is 2321ms.
7.the median is at 1208 ms .

Recommendation :
to investigate for bottle necks in the instances causing inconsistencies in performance
From aggregate report
1.average response time is 1167 ms, with a median of 1208 ms. (depends on application but in normal business standards too high) , server needs to be ramped up.
2.The minimum response time is quite low at 178 ms, but the maximum response time spikes to 4790 ms.
Recommendation:
Investigate the cause for the high maximum response time and address it to improve overall performance consistency.