Backend Testing

Load/Stress Test

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
Login	100	55	48	88	104	114	26	123	0.00%	5.4/sec	2.09	1.23
GetUploa...	100	33	35	49	54	69	8	70	0.00%	5.4/sec	192.50	0.90
GetUploa...	100	32	30	50	61	74	8	87	0.00%	5.4/sec	198.02	0.86
FileUploa...	100	172	181	235	249	256	38	292	0.00%	5.4/sec	1.31	2104.68
FileUploa...	100	16610	16625	16934	16989	17027	15949	17045	0.00%	2.9/sec	0.70	143994.91
ProcessA...	100	44	37	75	89	107	21	108	0.00%	5.3/sec	1.22	0.89
ProcessFI...	100	41	38	60	75	88	18	96	0.00%	5.3/sec	1.22	0.85
PredictAP...	100	41	38	63	64	97	18	97	0.00%	5.3/sec	1.22	1.11
PredictFI...	100	39	37	57	68	80	16	108	0.00%	5.3/sec	1.22	0.98
Downloa...	100	22	21	32	36	73	6	75	0.00%	5.3/sec	78.72	0.99
Downloa...	100	43	45	71	82	104	7	106	0.00%	5.3/sec	365.03	1.04
TOTAL	1100	1558	40	235	16593	16929	6	17045	0.00%	31.3/sec	449.19	143989.62

Figure 1: Load test of 50 users on main functionality

The load test was done with *fifty mocked users* that went through the main functionality of the application twice (Login, upload, process, predict, and download). The results show that the application was able to handle up to fifty users at once without any problems. The expected number of users at a given time is only about 5 users, which is well within the limits of the tests. To note, it took file upload the longest time with an average of 16610ms or 16s because the files used to test it were large compared to the other files. This is because file upload was the area of largest concern.

The file sizes used for uploading were 2.5MB zip files. The goal was to test the file upload with file sizes up to 50MB, so to mock the 50MB files, 20 of the zip files were uploaded for each request. For processing, predicting, and downloading, the files were 0.25MB.

Summary table of file sizes and request completion times:

Request	File Size (MB)	Average time (s)
Upload	50MB	16.60s
Process	0.25MB	0.04s
Predict	0.25MB	0.04s
Download	0.25MB	0.04s

The oversight from using this type of load testing is that it tests the server response from the API endpoints which does not fully test the functionality of sending the mocked files through the provided algorithms from our stakeholder. This is what caused the process, prediction, and download to have completion times of 0.04 seconds. We were not able to mock the process, predict and download files with proper sized files since the example files we were provided were on the smaller end. It is not possible to test the API endpoints with random files just to fit the size requirement because there are other specific requirements for each file for the server to process the request successfully. This prevented us from testing the upper limits of the backend.

What was confirmed through this load testing is that even with fifty users, the server's response time was not hindered by the large amounts of users. Even with many requests being made at once, the server could handle and respond swiftly and with 0% error rate. Furthermore, the uploading of many large files did not overload the server and was handled gracefully.