

Contents

Introduction.....	2
Test Environment	2
Web Testing Summary (Swag Labs)	3
API Testing Summary.....	5
Automation Summary	14
Bug Summary.....	17
Conclusion	17

QA Assessment Summary

I. Introduction

The purpose of this QA assessment was to evaluate the functionality, reliability, and performance of the Swag Labs application and the JSONPlaceholder API. The assessment covered three main areas: web testing, API testing, and automation testing. Web testing focused on user interactions, checkout processes, and validation of form fields. API testing verified endpoints for retrieving, creating, and updating resources while handling both valid and invalid scenarios. Automation testing was performed to streamline repetitive test flows and ensure consistent results.

II. Test Environment

The testing was conducted on both web and API platforms to ensure proper functionality, responsiveness, and reliability across different scenarios.

Web Application: <https://www.saucedemo.com>

API Under Test: <https://jsonplaceholder.typicode.com>

Browser: Chrome (Windows 11)

Tools Used:

- Postman - API testing
- Playwright - Automation script
- Excel - Test case and bug logging

Github Repo Used:

Note: All web and API test cases, along with the bug reports, are compiled in a single Excel file named **QA_Assessment_TestCases_Bantucan.xlsx** for easy reference and organization.

III. Web Testing Summary (Swag Labs)

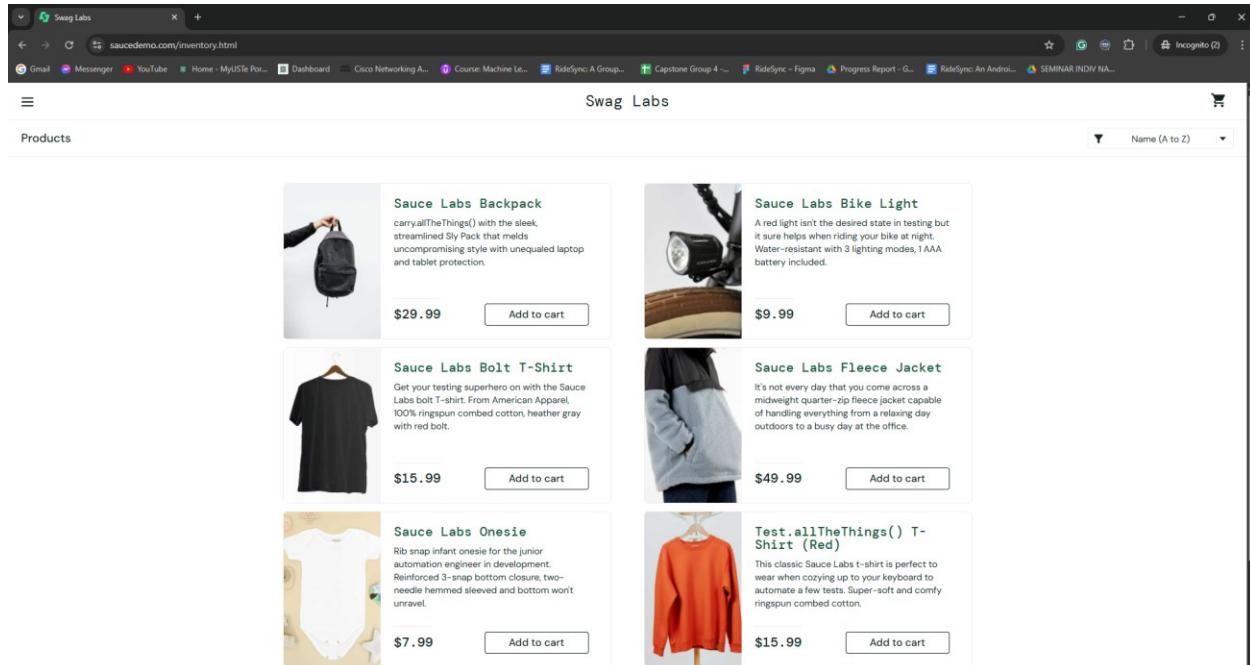
Total Test Cases: 21

Pass: 11

Fail: 10

A total of 21 test cases were executed for the checkout module. Out of these, 11 passed while 10 failed. The tests included scenarios such as proceeding through the checkout steps, validating required fields, verifying order totals, and handling URL manipulation. Several major bugs were identified, including the ability to bypass checkout steps through URL changes, completing checkout with an empty cart, and resubmitting orders after completion. Minor UI issues, such as oversized clickable areas, were also noted. Screenshots captured during testing illustrate successful workflows and areas where problems occurred.

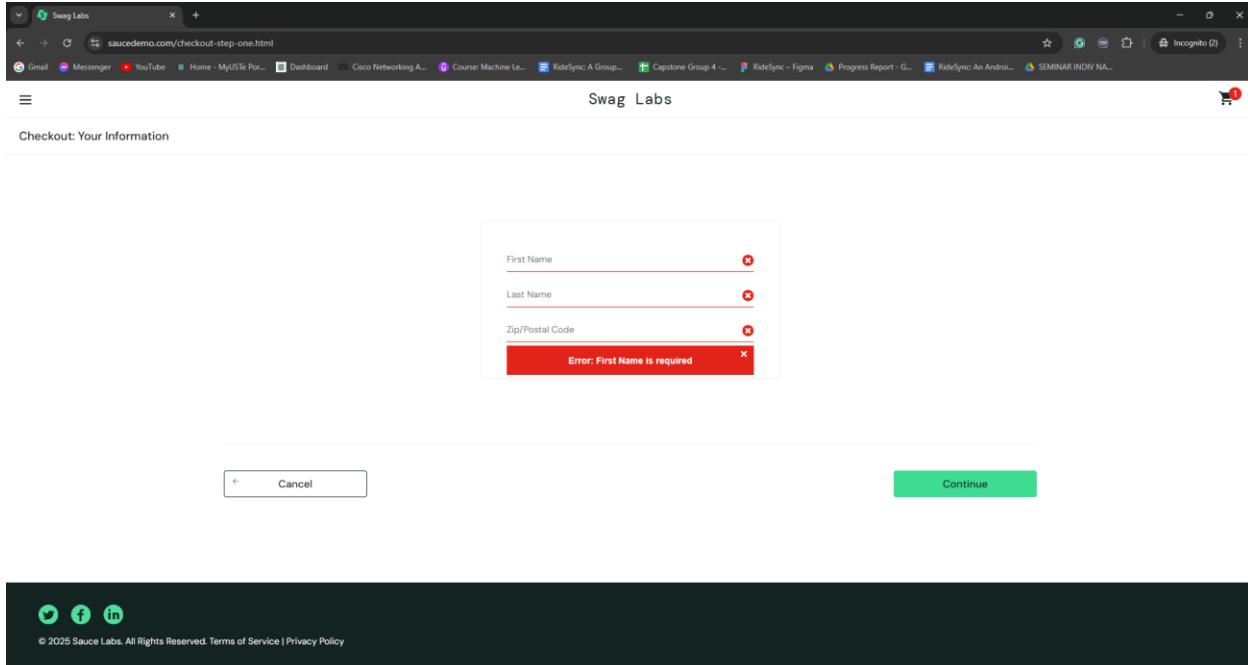
Screenshots:



Bantucan, Adrian S.

Project Name	Swag Labs (Practical Exam)											
Module Name	Add to Cart											
Created By	Adrian Bantucan											
Creation Date	December 05, 2025											
Test Scenario ID	Test Scenario Description	Test Case ID	Test Priority	Test Title	Test Case Description	Pre-Conditions	Test Steps	Test Data	Expected Result	Actual Result	Post-Conditions	Status
7	Verify user can add a product to the cart	TC_SL_CART_001	High	Add Single Product to Cart	Verify that the user can successfully add a single product to the cart and that the cart updates accordingly	The user must be logged in	1. Go to the "Products" page 2. Pick a product (Like: Sauce Labs Backpack) and click the "Add to cart" button 3. Observe the cart icon at the top right corner to see if the number has changed	The cart icon should update and it should show "1" and the product should be added to the cart	The cart icon shows "1" and it has been added	The product has been added to the cart and the user can continue shopping	Pass	
8	Verify that the user can add a product to the cart only once, and that the system prevents adding the same product multiple times	TC_SL_CART_002	High	Add Multiple Same Products to Cart	Validate that after adding a product to the cart, the "Add Product" button changes to "Remove" (or similar), preventing duplicate additions, and the cart accurately reflects the single instance of the product	The user must be logged in	1. Go to the "Products" page 2. Pick a product (Like: Sauce Labs Backpack) 3. Click the "Add to cart" button 4. Try to add the same product (Sauce Labs Backpack) 5. Observe the cart icon at the top right corner	The system should prevent users from adding multiple identical products to the cart. The "Add to cart" button should change to "Remove" button	The system prevented users from adding multiple identical products simultaneously. The "Add to cart" button does change to "Remove" button	The product has been added to the cart and there are no duplicate entries.	Pass	
9	Verify that the user can add multiple products to the cart	TC_SL_CART_003	High	Add Multiple Different Products to Cart	Verify that the user can add multiple different products to the cart and that each product appears as a separate line item	The user must be logged in	1. Go to the "Products" page 2. Pick multiple different products (Like: Sauce Labs Backpack, Sauce Labs Onesie, and Sauce Labs Bike Light) 3. Then click the "Add to cart" button 4. Observe the cart icon at the top right corner to see if the number has changed	The cart icon should update to show the number of added products, and the item should be added to the cart	The cart icon shows "3" and all products have been added	The products have been added to the cart and the user can continue shopping	Pass	
							1. Go to the "Products" page 2. Pick all six products by clicking the "Add to cart" button	The cart icon should update to show the number of added products	The cart icon	The cart icon should update to show the number of added products		

A	B	C	D	E	F	G	H	I	J	K	L	M
Project Name	Swag Labs (Practical Exam)											
Module Name	Checkout											
Created By	Adrian Bantucan											
Creation Date	December 05, 2025											
Test Scenario ID	Test Scenario Description	Test Case ID	Test Priority	Test Title	Test Case Description	Pre-Conditions	Test Steps	Test Data	Expected Result	Actual Result	Post-Conditions	Status
7	Verify that the user can proceed to the first step of checkout	TC_SL_CHECKOUT_001	High	First Step of Checking Out	Validate that clicking the checkout button leads the user to the "Checkout: Your Information" page	The user must be logged in and must have products in the cart	1. Add a product by clicking the "Add to cart"	The user should be directed to the "Checkout: Your Information" page and a form is displayed	The user is directed to the "Checkout: Your Information" page and a form is displayed	The user must be on the next page which is "Checkout: Your Information" page	Pass	
8	Verify that users can continue checkout when all required fields are correctly filled	TC_SL_CHECKOUT_002	High	Continuity of Checkout with Valid Information	Validate that entering valid first name, last name, and postal code allows progression to the checkout overview page	The user must be logged in and must be in the "Checkout: Your Information" page	1. Fill out the form with valid information 2. Click the "Continue" green button 3. Observe the next page	First Name: Adrian Last Name: Tester Zip/Postal Code: 490	The user should be directed to the "Checkout: Overview" page and the overview is indeed there	The user is directed to the "Checkout: Overview" page and the overview is indeed there	Pass	
9	Verify system behavior when the first name field is left blank	TC_SL_CHECKOUT_003	High	Checkout with a Missing First Name	Validate that leaving the first name empty prevents checkout from continuing and displays an appropriate error message	The user must be logged in and must be in the "Checkout: Your Information" page	1. Fill out the form with valid information, but leave the First Name field blank 2. Click the "Continue" green button 3. Observe the form	Last Name: Tester Zip/Postal Code: 490	The system will not let the user proceed to the next step and a validation will show saying "Error: First Name is Required"	The system did not let the user proceed with the next step and a validation indeed showed saying "Error: First Name is Required"	Pass	
							1. Fill out the form with valid information, but leave the Last Name field blank		The system will not let the user proceed to the next step and a validation will show saying "Error: Last Name is Required"	The system did not let the user proceed with the next step and a validation indeed showed	Pass	



IV. API Testing Summary

Endpoints Tested:

GET /users - retrieve all users
GET /users/1 - retrieve single user
GET /users/999 - non-existent user
POST /posts - create new post
PUT /posts/1 - update existing post

Status Codes Received:

200 OK - valid GET and PUT requests
201 Created - POST requests
404 Not Found - invalid endpoints or non-existent resources
500 Internal Server Error - updating a non-existing post

The API tests focused on the JSONPlaceholder endpoints for users and posts. Key endpoints tested included retrieving all users, retrieving a single user, creating new posts, and updating posts. Most tests returned the expected responses, including 200 OK and 201 Created status codes. Some edge cases were highlighted, such as the API allowing the creation of posts without required fields

and returning a 500 Internal Server Error when attempting to update non-existent posts. Sample responses were captured to validate data structure and content accuracy.

Screenshots:

-GET Retrieve All Users

The screenshot shows the Postman application interface. The left sidebar displays a collection named "avalon harlow's Workspace" with a "Technical Assessment" folder containing several API endpoints. The "GET Retrieve All Users" endpoint is selected and highlighted in orange. The main workspace shows a GET request to the URL <https://jsonplaceholder.typicode.com/users>. The response status is 200 OK, and the response body is displayed as JSON. The JSON data represents two users, each with an ID, name, username, email, address, phone, website, and company information.

```

1 [
2   {
3     "id": 1,
4     "name": "Leanne Graham",
5     "username": "Bret",
6     "email": "Sincere@april.biz",
7     "address": {
8       "street": "Kulas Light",
9       "suite": "Apt. 556",
10      "city": "Gwenborough",
11      "zipcode": "92998-3674",
12      "geo": {
13        "lat": "-37.3159",
14        "lng": "81.1496"
15      }
16    },
17    "phone": "1-770-736-8031 x56442",
18    "website": "hildegard.org",
19    "company": {
20      "name": "Romaguera-Crona",
21      "catchPhrase": "Multi-layered client-server neural-net",
22      "bs": "harness real-time e-markets"
23    }
24  ],
25  [
26    {
27      "id": 2,
28      "name": "Ervin Howell",
29      "username": "Antonette",
30      "email": "Shanna@melissa.tv",
31      "address": {
32        "street": "Victor Plains",
33        "suite": "Suite 879",
34        ...
35      }
36    }
37  ]
38 ]
39
40 ]
41
42 ]
43
44 ]
45
46 ]
47
48 ]
49
50 ]
51
52 ]
53
54 ]
55
56 ]
57
58 ]
59
60 ]
61
62 ]
63
64 ]
65
66 ]
67
68 ]
69
70 ]
71
72 ]
73
74 ]
75
76 ]
77
78 ]
79
80 ]
81
82 ]
83
84 ]
85
86 ]
87
88 ]
89
90 ]
91
92 ]
93
94 ]
95
96 ]
97
98 ]
99
100 ]
101 ]
102 ]
103 ]
104 ]
105 ]
106 ]
107 ]
108 ]
109 ]
110 ]
111 ]
112 ]
113 ]
114 ]
115 ]
116 ]
117 ]
118 ]
119 ]
120 ]
121 ]
122 ]
123 ]
124 ]
125 ]
126 ]
127 ]
128 ]
129 ]
130 ]
131 ]
132 ]
133 ]
134 ]
135 ]
136 ]
137 ]
138 ]
139 ]
140 ]
141 ]
142 ]
143 ]
144 ]
145 ]
146 ]
147 ]
148 ]
149 ]
150 ]
151 ]
152 ]
153 ]
154 ]
155 ]
156 ]
157 ]
158 ]
159 ]
160 ]
161 ]
162 ]
163 ]
164 ]
165 ]
166 ]
167 ]
168 ]
169 ]
170 ]
171 ]
172 ]
173 ]
174 ]
175 ]
176 ]
177 ]
178 ]
179 ]
180 ]
181 ]
182 ]
183 ]
184 ]
185 ]
186 ]
187 ]
188 ]
189 ]
190 ]
191 ]
192 ]
193 ]
194 ]
195 ]
196 ]
197 ]
198 ]
199 ]
200 ]
201 ]
202 ]
203 ]
204 ]
205 ]
206 ]
207 ]
208 ]
209 ]
210 ]
211 ]
212 ]
213 ]
214 ]
215 ]
216 ]
217 ]
218 ]
219 ]
220 ]
221 ]
222 ]
223 ]
224 ]
225 ]
226 ]
227 ]
228 ]
229 ]
229 ]
230 ]
231 ]
232 ]
233 ]
234 ]
235 ]
236 ]
237 ]
238 ]
239 ]
239 ]
240 ]
241 ]
242 ]
243 ]
244 ]
245 ]
246 ]
247 ]
248 ]
249 ]
249 ]
250 ]
251 ]
252 ]
253 ]
254 ]
255 ]
256 ]
257 ]
258 ]
259 ]
259 ]
260 ]
261 ]
262 ]
263 ]
264 ]
265 ]
266 ]
267 ]
268 ]
269 ]
269 ]
270 ]
271 ]
272 ]
273 ]
274 ]
275 ]
276 ]
277 ]
278 ]
279 ]
279 ]
280 ]
281 ]
282 ]
283 ]
284 ]
285 ]
286 ]
287 ]
288 ]
289 ]
289 ]
290 ]
291 ]
292 ]
293 ]
294 ]
295 ]
296 ]
297 ]
298 ]
299 ]
299 ]
300 ]
301 ]
302 ]
303 ]
304 ]
305 ]
306 ]
307 ]
308 ]
309 ]
309 ]
310 ]
311 ]
312 ]
313 ]
314 ]
315 ]
316 ]
317 ]
318 ]
319 ]
319 ]
320 ]
321 ]
322 ]
323 ]
324 ]
325 ]
326 ]
327 ]
328 ]
329 ]
329 ]
330 ]
331 ]
332 ]
333 ]
334 ]
335 ]
336 ]
337 ]
338 ]
339 ]
339 ]
340 ]
341 ]
342 ]
343 ]
344 ]
345 ]
346 ]
347 ]
348 ]
349 ]
349 ]
350 ]
351 ]
352 ]
353 ]
354 ]
355 ]
356 ]
357 ]
358 ]
359 ]
359 ]
360 ]
361 ]
362 ]
363 ]
364 ]
365 ]
366 ]
367 ]
368 ]
369 ]
369 ]
370 ]
371 ]
372 ]
373 ]
374 ]
375 ]
376 ]
377 ]
378 ]
379 ]
379 ]
380 ]
381 ]
382 ]
383 ]
384 ]
385 ]
386 ]
387 ]
388 ]
389 ]
389 ]
390 ]
391 ]
392 ]
393 ]
394 ]
395 ]
396 ]
397 ]
398 ]
399 ]
399 ]
400 ]
401 ]
402 ]
403 ]
404 ]
405 ]
406 ]
407 ]
408 ]
409 ]
409 ]
410 ]
411 ]
412 ]
413 ]
414 ]
415 ]
416 ]
417 ]
418 ]
419 ]
419 ]
420 ]
421 ]
422 ]
423 ]
424 ]
425 ]
426 ]
427 ]
428 ]
429 ]
429 ]
430 ]
431 ]
432 ]
433 ]
434 ]
435 ]
436 ]
437 ]
438 ]
439 ]
439 ]
440 ]
441 ]
442 ]
443 ]
444 ]
445 ]
446 ]
447 ]
448 ]
449 ]
449 ]
450 ]
451 ]
452 ]
453 ]
454 ]
455 ]
456 ]
457 ]
458 ]
459 ]
459 ]
460 ]
461 ]
462 ]
463 ]
464 ]
465 ]
466 ]
467 ]
468 ]
469 ]
469 ]
470 ]
471 ]
472 ]
473 ]
474 ]
475 ]
476 ]
477 ]
478 ]
479 ]
479 ]
480 ]
481 ]
482 ]
483 ]
484 ]
485 ]
486 ]
487 ]
488 ]
489 ]
489 ]
490 ]
491 ]
492 ]
493 ]
494 ]
495 ]
496 ]
497 ]
498 ]
499 ]
499 ]
500 ]
501 ]
502 ]
503 ]
504 ]
505 ]
506 ]
507 ]
508 ]
509 ]
509 ]
510 ]
511 ]
512 ]
513 ]
514 ]
515 ]
516 ]
517 ]
518 ]
519 ]
519 ]
520 ]
521 ]
522 ]
523 ]
524 ]
525 ]
526 ]
527 ]
528 ]
529 ]
529 ]
530 ]
531 ]
532 ]
533 ]
534 ]
535 ]
536 ]
537 ]
538 ]
539 ]
539 ]
540 ]
541 ]
542 ]
543 ]
544 ]
545 ]
546 ]
547 ]
548 ]
549 ]
549 ]
550 ]
551 ]
552 ]
553 ]
554 ]
555 ]
556 ]
557 ]
558 ]
559 ]
559 ]
560 ]
561 ]
562 ]
563 ]
564 ]
565 ]
566 ]
567 ]
568 ]
569 ]
569 ]
570 ]
571 ]
572 ]
573 ]
574 ]
575 ]
576 ]
577 ]
578 ]
579 ]
579 ]
580 ]
581 ]
582 ]
583 ]
584 ]
585 ]
586 ]
587 ]
588 ]
589 ]
589 ]
590 ]
591 ]
592 ]
593 ]
594 ]
595 ]
596 ]
597 ]
598 ]
599 ]
599 ]
600 ]
601 ]
602 ]
603 ]
604 ]
605 ]
606 ]
607 ]
608 ]
609 ]
609 ]
610 ]
611 ]
612 ]
613 ]
614 ]
615 ]
616 ]
617 ]
618 ]
619 ]
619 ]
620 ]
621 ]
622 ]
623 ]
624 ]
625 ]
626 ]
627 ]
628 ]
629 ]
629 ]
630 ]
631 ]
632 ]
633 ]
634 ]
635 ]
636 ]
637 ]
638 ]
639 ]
639 ]
640 ]
641 ]
642 ]
643 ]
644 ]
645 ]
646 ]
647 ]
648 ]
649 ]
649 ]
650 ]
651 ]
652 ]
653 ]
654 ]
655 ]
656 ]
657 ]
658 ]
659 ]
659 ]
660 ]
661 ]
662 ]
663 ]
664 ]
665 ]
666 ]
667 ]
668 ]
669 ]
669 ]
670 ]
671 ]
672 ]
673 ]
674 ]
675 ]
676 ]
677 ]
678 ]
679 ]
679 ]
680 ]
681 ]
682 ]
683 ]
684 ]
685 ]
686 ]
687 ]
688 ]
689 ]
689 ]
690 ]
691 ]
692 ]
693 ]
694 ]
695 ]
696 ]
697 ]
698 ]
699 ]
699 ]
700 ]
701 ]
702 ]
703 ]
704 ]
705 ]
706 ]
707 ]
708 ]
709 ]
709 ]
710 ]
711 ]
712 ]
713 ]
714 ]
715 ]
716 ]
717 ]
718 ]
719 ]
719 ]
720 ]
721 ]
722 ]
723 ]
724 ]
725 ]
726 ]
727 ]
728 ]
729 ]
729 ]
730 ]
731 ]
732 ]
733 ]
734 ]
735 ]
736 ]
737 ]
738 ]
739 ]
739 ]
740 ]
741 ]
742 ]
743 ]
744 ]
745 ]
746 ]
747 ]
748 ]
749 ]
749 ]
750 ]
751 ]
752 ]
753 ]
754 ]
755 ]
756 ]
757 ]
758 ]
759 ]
759 ]
760 ]
761 ]
762 ]
763 ]
764 ]
765 ]
766 ]
767 ]
768 ]
769 ]
769 ]
770 ]
771 ]
772 ]
773 ]
774 ]
775 ]
776 ]
777 ]
778 ]
779 ]
779 ]
780 ]
781 ]
782 ]
783 ]
784 ]
785 ]
786 ]
787 ]
788 ]
789 ]
789 ]
790 ]
791 ]
792 ]
793 ]
794 ]
795 ]
796 ]
797 ]
798 ]
799 ]
799 ]
800 ]
801 ]
802 ]
803 ]
804 ]
805 ]
806 ]
807 ]
808 ]
809 ]
809 ]
810 ]
811 ]
812 ]
813 ]
814 ]
815 ]
816 ]
817 ]
818 ]
819 ]
819 ]
820 ]
821 ]
822 ]
823 ]
824 ]
825 ]
826 ]
827 ]
828 ]
829 ]
829 ]
830 ]
831 ]
832 ]
833 ]
834 ]
835 ]
836 ]
837 ]
838 ]
839 ]
839 ]
840 ]
841 ]
842 ]
843 ]
844 ]
845 ]
846 ]
847 ]
848 ]
849 ]
849 ]
850 ]
851 ]
852 ]
853 ]
854 ]
855 ]
856 ]
857 ]
858 ]
859 ]
859 ]
860 ]
861 ]
862 ]
863 ]
864 ]
865 ]
866 ]
867 ]
868 ]
869 ]
869 ]
870 ]
871 ]
872 ]
873 ]
874 ]
875 ]
876 ]
877 ]
878 ]
878 ]
879 ]
880 ]
881 ]
882 ]
883 ]
884 ]
885 ]
886 ]
887 ]
888 ]
889 ]
889 ]
890 ]
891 ]
892 ]
893 ]
894 ]
895 ]
896 ]
897 ]
898 ]
898 ]
899 ]
900 ]
901 ]
902 ]
903 ]
904 ]
905 ]
906 ]
907 ]
908 ]
909 ]
909 ]
910 ]
911 ]
912 ]
913 ]
914 ]
915 ]
916 ]
917 ]
918 ]
919 ]
919 ]
920 ]
921 ]
922 ]
923 ]
924 ]
925 ]
926 ]
927 ]
928 ]
929 ]
929 ]
930 ]
931 ]
932 ]
933 ]
934 ]
935 ]
936 ]
937 ]
938 ]
939 ]
939 ]
940 ]
941 ]
942 ]
943 ]
944 ]
945 ]
946 ]
947 ]
948 ]
949 ]
949 ]
950 ]
951 ]
952 ]
953 ]
954 ]
955 ]
956 ]
957 ]
958 ]
959 ]
959 ]
960 ]
961 ]
962 ]
963 ]
964 ]
965 ]
966 ]
967 ]
968 ]
969 ]
969 ]
970 ]
971 ]
972 ]
973 ]
974 ]
975 ]
976 ]
977 ]
978 ]
978 ]
979 ]
980 ]
981 ]
982 ]
983 ]
984 ]
985 ]
986 ]
987 ]
988 ]
989 ]
989 ]
990 ]
991 ]
992 ]
993 ]
994 ]
995 ]
996 ]
997 ]
998 ]
999 ]
999 ]
1000 ]
1001 ]
1002 ]
1003 ]
1004 ]
1005 ]
1006 ]
1007 ]
1008 ]
1009 ]
1009 ]
1010 ]
1011 ]
1012 ]
1013 ]
1014 ]
1015 ]
1016 ]
1017 ]
1018 ]
1019 ]
1019 ]
1020 ]
1021 ]
1022 ]
1023 ]
1024 ]
1025 ]
1026 ]
1027 ]
1028 ]
1029 ]
1029 ]
1030 ]
1031 ]
1032 ]
1033 ]
1034 ]
1035 ]
1036 ]
1037 ]
1038 ]
1039 ]
1039 ]
1040 ]
1041 ]
1042 ]
1043 ]
1044 ]
1045 ]
1046 ]
1047 ]
1048 ]
1049 ]
1049 ]
1050 ]
1051 ]
1052 ]
1053 ]
1054 ]
1055 ]
1056 ]
1057 ]
1058 ]
1059 ]
1059 ]
1060 ]
1061 ]
1062 ]
1063 ]
1064 ]
1065 ]
1066 ]
1067 ]
1068 ]
1069 ]
1069 ]
1070 ]
1071 ]
1072 ]
1073 ]
1074 ]
1075 ]
1076 ]
1077 ]
1078 ]
1078 ]
1079 ]
1080 ]
1081 ]
1082 ]
1083 ]
1084 ]
1085 ]
1086 ]
1087 ]
1088 ]
1088 ]
1089 ]
1090 ]
1091 ]
1092 ]
1093 ]
1094 ]
1095 ]
1096 ]
1097 ]
1098 ]
1098 ]
1099 ]
1099 ]
1100 ]
1101 ]
1102 ]
1103 ]
1104 ]
1105 ]
1106 ]
1107 ]
1108 ]
1109 ]
1109 ]
1110 ]
1111 ]
1112 ]
1113 ]
1114 ]
1115 ]
1116 ]
1117 ]
1118 ]
1119 ]
1119 ]
1120 ]
1121 ]
1122 ]
1123 ]
1124 ]
1125 ]
1126 ]
1127 ]
1128 ]
1129 ]
1129 ]
1130 ]
1131 ]
1132 ]
1133 ]
1134 ]
1135 ]
1136 ]
1137 ]
1138 ]
1139 ]
1139 ]
1140 ]
1141 ]
1142 ]
1143 ]
1144 ]
1145 ]
1146 ]
1147 ]
1148 ]
1149 ]
1149 ]
1150 ]
1151 ]
1152 ]
1153 ]
1154 ]
1155 ]
1156 ]
1157 ]
1158 ]
1159 ]
1159 ]
1160 ]
1161 ]
1162 ]
1163 ]
1164 ]
1165 ]
1166 ]
1167 ]
1168 ]
1169 ]
1169 ]
1170 ]
1171 ]
1172 ]
1173 ]
1174 ]
1175 ]
1176 ]
1177 ]
1178 ]
1178 ]
1179 ]
1180 ]
1181 ]
1182 ]
1183 ]
1184 ]
1185 ]
1186 ]
1187 ]
1188 ]
1188 ]
1189 ]
1190 ]
1191 ]
1192 ]
1193 ]
1194 ]
1195 ]
1196 ]
1197 ]
1197 ]
1198 ]
1199 ]
1199 ]
1200 ]
1201 ]
1202 ]
1203 ]
1204 ]
1205 ]
1206 ]
1207 ]
1208 ]
1209 ]
1209 ]
1210 ]
1211 ]
1212 ]
1213 ]
1214 ]
1215 ]
1216 ]
1217 ]
1218 ]
1219 ]
1219 ]
1220 ]
1221 ]
1222 ]
1223 ]
1224 ]
1225 ]
1226 ]
1227 ]
1228 ]
1229 ]
1229 ]
1230 ]
1231 ]
1232 ]
1233 ]
1234 ]
1235 ]
1236 ]
1237 ]
1238 ]
1239 ]
1239 ]
1240 ]
1241 ]
1242 ]
1243 ]
1244 ]
1245 ]
1246 ]
1247 ]
1248 ]
1249 ]
1249 ]
1250 ]
1251 ]
1252 ]
1253 ]
1254 ]
1255 ]
1256 ]
1257 ]
1258 ]
1259 ]
1259 ]
1260 ]
1261 ]
1262 ]
1263 ]
1264 ]
1265 ]
1266 ]
1267 ]
1268 ]
1269 ]
1269 ]
1270 ]
1271 ]
1272 ]
1273 ]
1274 ]
1275 ]
1276 ]
1277 ]
1278 ]
1278 ]
1279 ]
1280 ]
1281 ]
1282 ]
1283 ]
1284 ]
1285 ]
1286 ]
1287 ]
1288 ]
1288 ]
1289 ]
1290 ]
1291 ]
1292 ]
1293 ]
1294 ]
1295 ]
1296 ]
1297 ]
1297 ]
1298 ]
1299 ]
1299 ]
1300 ]
1301 ]
1302 ]
1303 ]
1304 ]
1305 ]
1306 ]
1307 ]
1308 ]
1309 ]
1309 ]
1310 ]
1311 ]
1312 ]
1313 ]
1314 ]
1315 ]
1316 ]
1317 ]
1318 ]
1319 ]
1319 ]
1320 ]
1321 ]
1322 ]
1323 ]
1324 ]
1325 ]
1326 ]
1327 ]
1328 ]
1329 ]
1329 ]
1330 ]
1331 ]
1332 ]
1333 ]
1334 ]
1335 ]
1336 ]
1337 ]
1338 ]
1339 ]
1339 ]
1340 ]
1341 ]
1342 ]
1343 ]
1344 ]
1345 ]
1346 ]
1347 ]
1348 ]
1349 ]
1349 ]
1350 ]
1351 ]
1352 ]
1353 ]
1354 ]
1355 ]
1356 ]
1357 ]
1358 ]
1359 ]
1359 ]
1360 ]
1361 ]
1362 ]
1363 ]
1364 ]
1365 ]
1366 ]
1367 ]
1368 ]
1369 ]
1369 ]
1370 ]
1371 ]
1372 ]
1373 ]
1374 ]
1375 ]
1376 ]
1377 ]
1378 ]
1378 ]
1379 ]
1380 ]
1381 ]
1382 ]
1383 ]
1384 ]
1385 ]
1386 ]
1387 ]
1388 ]
1388 ]
1389 ]
1390 ]
1391 ]
1392 ]
1393 ]
1394 ]
1395 ]
1396 ]
1397 ]
1398 ]
1398 ]
1399 ]
1399 ]
1400 ]
1401 ]
1402 ]
1403 ]
1404 ]
1405 ]
1406 ]
1407 ]
1408 ]
1409 ]
1409 ]
1410 ]
1411 ]
1412 ]
1413 ]
1414 ]
1415 ]
1416 ]
1417 ]
1418 ]
1419 ]
1419 ]
1420 ]
1421 ]
1422 ]
1423 ]
1424 ]
1425 ]
1426 ]
1427 ]
1428 ]
1429 ]
1429 ]
1430 ]
1431 ]
1432 ]
1433 ]
1434 ]
1435 ]
1436 ]
1437 ]
1438 ]
1439 ]
1439 ]
1440 ]
1441 ]
1442 ]
1443 ]
1444 ]
1445 ]
1446 ]
1447 ]
1448 ]
1449 ]
1449 ]
1450 ]
1451 ]
1452 ]
1453 ]
1454 ]
1455 ]
1456 ]
1457 ]
1458 ]
1459 ]
1459 ]
1460 ]
1461 ]
1462 ]
1463 ]
1464 ]
1465 ]
1466 ]
1467 ]
1468 ]
1469 ]
1469 ]
1470 ]
1471 ]
1472 ]
1473 ]
1474 ]
1475 ]
1476 ]
1477 ]
1478 ]
1478 ]
1479 ]
1480 ]
1481 ]
1482 ]
1483 ]
1484 ]
1485 ]
1486 ]
1487 ]
1488 ]
1488 ]
1489 ]
1490 ]
1491 ]
1492 ]
1493 ]
1494 ]
1495 ]
1496 ]
1497 ]
1497 ]
1498 ]
1499 ]
1499 ]
1500 ]
1501 ]
1502 ]
1503 ]
1504 ]
1505 ]
1506 ]
1507 ]
1508 ]
1509 ]
1509 ]
1510 ]
1511 ]
1512 ]
1513 ]
1514 ]
1515 ]
1516 ]
1517 ]
1518 ]
1519 ]
1519 ]
1520 ]
1521 ]
1522 ]
1523 ]
1524 ]
1525 ]
1526 ]
1527 ]
1528 ]
1529 ]
1529 ]
1530 ]
1531 ]
1532 ]
1533 ]
1534 ]
1535 ]
1536 ]
1537 ]
1538 ]
1539 ]
1539 ]
1540 ]
1541 ]
1542 ]
1543 ]
1544 ]
1545 ]
1546 ]
1547 ]
1548 ]
1549 ]
1549 ]
1550 ]
1551 ]
1552 ]
1553 ]
1554 ]
1555 ]
1556 ]
1557 ]
1558 ]
1559 ]
1559 ]
1560 ]
1561 ]
1562 ]
1563 ]
1564 ]
1565 ]
1566 ]
1567 ]
1568 ]
1569 ]
1569 ]
1570 ]
1571 ]
1572 ]
1573 ]
1574 ]
1575 ]
1576 ]
1577 ]
1578 ]
1578 ]
1579 ]
1580 ]
1581 ]
1582 ]
1583 ]
1584 ]
1585 ]
1586 ]
1587 ]
1588 ]
1588 ]
1589 ]
1590 ]
1591 ]
1592 ]
1593 ]
1594 ]
1595 ]
1596 ]
1597 ]
1597 ]
1598 ]
1599 ]
1599 ]
1600 ]
1601 ]
1602 ]
1603 ]
1604 ]
1605 ]
1606 ]
1607 ]
1608 ]
1609 ]
1609 ]
1610 ]
1611 ]
1612 ]
1613 ]
1614 ]
1615 ]
1616 ]
1617 ]
1618 ]
1619 ]
1619 ]
1620 ]
1621 ]
1622 ]
1623 ]
1624 ]
1625 ]
1626 ]
1627 ]
1628 ]
1629 ]
1629 ]
1630 ]
1631 ]
1632 ]
1633 ]
1634 ]
1635 ]
1636 ]
1637 ]
1638 ]
1639 ]
1639 ]
1640 ]
1641 ]
1642 ]
1643 ]
1644 ]
1645 ]
1646 ]
1647 ]
1648 ]
1649 ]
1649 ]
1650 ]
1651 ]
1652 ]
1653 ]
1654 ]
1655 ]
1656 ]
1657 ]
1658 ]
1659 ]
1659 ]
1660 ]
1661 ]
1662 ]
1663 ]
1664 ]
1665 ]
1666 ]
1667 ]
1668 ]
1669 ]
1669 ]
1670 ]
1671 ]
1672 ]
1673 ]
1674 ]
1675 ]
1676 ]
1677 ]
1678 ]
1678 ]
1679 ]
1680 ]
1681 ]
1682 ]
1683 ]
1684 ]
1685 ]
1686 ]
1687 ]
1688 ]
1688 ]
1689 ]
1690 ]
1691 ]
1692 ]
1693 ]
1694 ]
1695 ]
1696 ]
1697 ]
1697 ]
1698 ]
1699 ]
1699 ]
1700 ]
1701 ]
1702 ]
1703 ]
1704 ]
1705 ]
1706 ]
1707 ]
1708 ]
1709 ]
1709 ]
1710 ]
1711 ]
1712 ]
1713 ]
1714 ]
1715 ]
1716 ]
1717 ]
1718 ]
1719 ]
1719 ]
1720 ]
1721 ]
1722 ]
1723 ]
1724 ]
1725 ]
1726 ]
1727 ]
1728 ]
1729 ]
1729 ]
1730 ]
1731 ]
1732 ]
1733 ]
1734 ]
1735 ]
1736 ]
1737 ]
1738 ]
1739 ]
1739 ]
1740 ]
1741 ]
1742 ]
1743 ]
1744 ]
1745 ]
1746 ]
1747 ]
1748 ]
1749 ]
1749 ]
1750 ]
1751 ]
1752 ]
1753 ]
1754 ]
1755 ]
1756 ]
1757 ]
1758 ]
1759 ]
1759 ]
1760 ]
1761 ]
1762 ]
1763 ]
1764 ]
1765 ]
1766 ]
1767 ]
1768 ]
1769 ]
1769 ]
1770 ]
1771 ]
1772 ]
1773 ]
1774 ]
1775 ]
1776 ]
1777 ]
1778 ]
1778 ]
1779 ]
1780 ]
1781 ]
1782 ]
1783 ]
1784 ]
1785 ]
1786 ]
1787 ]
1788 ]
1788 ]
1789 ]
1790 ]
1791 ]
1792 ]
1793 ]
1794 ]
1795 ]
1796 ]
1797 ]
1797 ]
1798 ]
1799 ]
1799 ]
1800 ]
1801 ]
1802 ]
1803 ]
1804 ]
1805 ]
1806 ]
1807 ]
1808 ]
1809 ]
1809 ]
1810 ]
1811 ]
1812 ]
1813 ]
1814 ]
1815 ]
1816 ]
1817 ]
1818 ]
1819 ]
1819 ]
1820 ]
1821 ]
1822 ]
1823 ]
1824 ]
1825 ]
1826 ]
1827 ]
1828 ]
1829 ]
1829 ]
1830 ]
1831 ]
1832 ]
1833 ]
1834 ]
1835 ]
1836 ]
1837 ]
1838 ]
1839 ]
1839 ]
1840 ]
1841 ]
1842 ]
1843 ]
1844 ]
1845 ]
1846 ]
1847 ]
1848 ]
1849 ]
1849 ]
1850 ]
1851 ]
1852 ]
1853 ]
1854 ]
1855 ]
1856 ]
1857 ]
1858 ]
1859 ]
1859 ]
1860 ]
1861 ]
1862 ]
1863 ]
1864 ]
1865 ]
1866 ]
1867 ]
1868 ]
1869 ]
1869 ]
1870 ]
1871 ]
1872 ]
1873 ]
1874 ]
1875 ]
1876 ]
1877 ]
1878 ]
1878 ]
1879 ]
1880 ]
1881 ]
1882 ]
1883 ]
1884 ]
1885 ]
1886 ]
1887 ]
1888 ]
1888 ]
1889 ]
1890 ]
1891 ]
1892 ]
1893 ]
1894 ]
1895 ]
1896 ]
1897 ]
1898 ]
1898 ]
1899 ]
1900 ]
1901 ]
1902 ]
1903 ]
1904 ]
1905 ]
1906 ]
1907 ]
1908 ]
1909 ]
1909 ]
1910 ]
1911 ]
1912 ]
1913 ]
1914 ]
1915 ]
1916 ]
1917 ]
1918 ]
1919 ]
1919 ]
1920 ]
1921 ]
1922 ]
1923 ]
1924 ]
1925 ]
1926 ]
1927 ]
1928 ]
1929 ]
1929 ]
1930 ]
1931 ]
1932 ]
1933 ]
1934 ]
1935 ]
1936 ]
1937 ]
1938 ]
1939 ]
1939 ]
1940 ]
1941 ]
1942 ]
1943 ]
1944 ]
1945 ]
1946 ]
1947 ]
1948 ]
1949 ]
1949 ]
1950 ]
1951 ]
1952 ]
1953 ]
1954 ]
1955 ]
1956 ]
1957 ]
1958 ]
1959 ]
1959 ]
1960 ]
1961 ]
1962 ]
1963 ]
1964 ]
1965 ]
1966 ]
1967 ]
1968 ]
1969 ]
1969 ]
1970 ]
1971 ]
1972 ]
1973 ]
1974 ]
1975 ]
1976 ]
1977 ]
1978 ]
1978 ]
1979 ]
1980 ]
1981 ]
1982 ]
1983 ]
1984 ]
1985 ]
1986 ]
1987 ]
1988 ]
1988 ]
1989 ]
1990 ]
1991 ]
1992 ]
1993 ]
1994 ]
1995 ]
1996 ]
1997 ]
1998 ]
1999 ]
1999 ]
2000 ]
2001 ]
2002 ]
2003 ]
2004 ]
2005 ]
2006 ]
2007 ]
2008 ]
2009 ]
2009 ]
2010 ]
2011 ]
2012 ]
2013 ]
2014 ]
2015 ]
2016 ]
2017 ]
2018 ]
2019 ]
2019 ]
2020 ]
2021 ]
2022 ]
2023 ]
2024 ]
2025 ]
2026 ]
2027 ]
2028 ]
2029 ]
2029 ]
2030 ]
2031 ]
2032 ]
2033 ]
2034 ]
2035 ]
2036 ]
2037 ]
2038 ]
2039 ]
2039 ]
2040 ]
2041 ]
2042 ]
2043 ]
2044 ]
2045 ]
2046 ]
2047 ]
2048 ]
2049 ]
2049 ]
2050 ]
2051 ]
2052 ]
2053 ]
2054 ]
2055 ]
2056 ]
2057 ]
2058 ]
2059 ]
2059 ]
2060 ]
2061 ]
2062 ]
2063 ]
2064 ]
2065 ]
2066 ]
2067 ]
2068 ]
2069 ]
2069 ]
2070 ]
2071 ]
2072 ]
2073 ]
2074 ]
2075 ]
2076 ]
2077 ]
2078 ]
2078 ]
2079 ]
2080 ]
2081 ]
2082 ]
2083 ]
2084 ]
2085 ]
2086 ]
2087 ]
2088 ]
2088 ]
2089 ]
2090 ]
2091 ]
2092 ]
2093 ]
2094 ]
2095 ]
2096 ]
2097 ]
2098 ]
2098 ]
2099 ]
2099 ]
2100 ]
2101 ]
2102 ]
2103 ]
2104 ]
2105 ]
2106 ]
2107 ]
2108 ]
2109 ]
2109 ]
2110 ]
2111 ]
2112 ]
2113 ]
2114 ]
2115 ]
2116 ]
2117 ]
2118 ]
2119 ]
2119 ]
2120 ]
2121 ]
2122 ]
2123 ]
2124 ]
2125 ]
2126 ]
2127 ]
2128 ]
2129 ]
2129 ]
2130 ]
2131 ]
2132 ]
2133 ]
2134 ]
2135 ]
2136 ]
2137 ]
2138 ]
2139 ]
2139 ]
2140 ]
2141 ]
2142 ]
2143 ]
2144 ]
2145 ]
2146 ]
2147 ]
2148 ]
2149 ]
2149 ]
2150 ]
2151 ]
2152 ]
2153 ]
2154 ]
2155 ]
2156 ]
2157 ]
2158 ]
2159 ]
2159 ]
2160 ]
2161 ]
2162 ]
2163 ]
2164 ]
2165 ]
2166 ]
2167 ]
2168 ]
2169 ]
2169 ]
2170 ]
2171 ]
2172 ]
2173 ]
2174 ]
2175 ]
2176 ]
2177 ]
2178 ]
2178 ]
2179 ]
2180 ]
2181 ]
2182 ]
2183 ]
2184 ]
2185 ]
2186 ]
2187 ]
2188 ]
2188
```

Bantucan, Adrian S.

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Home, Workspaces, API Network, Collections, Environments, History, Flows, and Documentation. The main workspace is titled "avalon harlow's Workspace" and contains a "Technical Assessment" collection. Within this collection, there is a "Retrieve All Users" endpoint. The request URL is <https://sonplaceholder.typicode.com/users>. The response status is 200 OK, and the response body is a JSON object representing multiple user profiles. One user profile includes the following details:

```
        "id": 1,
        "name": "Romaguera-Jacobson",
        "username": "Ramiel",
        "email": "Julianne.Conner@kory.org",
        "address": {
            "street": "Hoeger Mall",
            "suite": "Apt. 602",
            "city": "South Elvins",
            "zipcode": "63959-4287",
            "geo": [
                {
                    "lat": "99.0987",
                    "lng": "-164.2996"
                }
            ],
            "phone": "493-178-9623 x156",
            "website": "Kale.dil"
        },
        "company": {
            "name": "Rehail-Corley",
            "catchPhrase": "Multi-tiered zero tolerance productivity"
        }
```

The screenshot shows the Postman application interface. The left sidebar displays 'avalon harlow's Workspace' with several collections: 'My Collection' and 'Technical Assessment'. Under 'Technical Assessment', there are four items: 'GET Retrieve All Users' (selected), 'GET Retrieve All Invalid Users', 'GET Retrieve Single User', and 'GET Retrieve Non-Existing User'. The main workspace shows a 'Technical Assessment / Retrieve All Users' collection. A 'GET' request is selected with the URL <https://sonplaceholder.typicode.com/users>. The response status is '200 OK' with a duration of '355 ms' and a size of '2.94 KB'. The response body is a JSON array of user objects:

```
85 [
86   {
87     "id": 1,
88     "name": "Leanne Graham",
89     "username": "Bret",
90     "email": "Sincere@april.biz",
91     "address": {
92       "street": "Kulas Light",
93       "suite": "Apt. 550",
94       "city": "Levenger",
95       "zipcode": "92902-1234",
96       "geo": {
97         "lat": "41.8834",
98         "lng": "-73.9352"
99       }
100     },
101     "phone": "(091)982-3206",
102     "website": "hildegard.org",
103     "company": {
104       "name": "Romaguera-Crona",
105       "catchPhrase": "Multi-tiered zero tolerance productivity",
106       "bs": "transition cutting-edge web services"
107     }
108   },
109   {
110     "id": 2,
111     "name": "Chelsey Dietrich",
112     "username": "Kamren",
113     "email": "Lucio_Nattinger@annie.ca",
114     "address": {
115       "street": "Skiles Walks",
116       "suite": "Suite 351",
117       "city": "Roscoeview",
118       "zipcode": "33263",
119       "geo": {
120         "lat": "29.4572",
121         "lng": "-98.3125"
122       }
123     },
124     "phone": "(254)994-1289",
125     "website": "demarco.info",
126     "company": {
127       "name": "Keebler LLC",
128       "catchPhrase": "User-centric fault-tolerant solution",
129       "bs": "revolutionize end-to-end systems"
130     }
131   },
132   {
133     "id": 3,
134     "name": "Kori Lakin",
135     "username": "Elwyn.Skiles",
136     "email": "Sincere@april.biz",
137     "address": {
138       "street": "Kulas Light",
139       "suite": "Apt. 550",
140       "city": "Levenger",
141       "zipcode": "92902-1234",
142       "geo": {
143         "lat": "41.8834",
144         "lng": "-73.9352"
145       }
146     },
147     "phone": "(091)982-3206",
148     "website": "hildegard.org",
149     "company": {
150       "name": "Romaguera-Crona",
151       "catchPhrase": "Multi-tiered zero tolerance productivity",
152       "bs": "transition cutting-edge web services"
153     }
154   }
155 ]
```

Bantucan, Adrian S.

avalon harlow's Workspace

New Import

GET Retrieve Single User POST Create New Post GET Retrieve All Users

POST Create Post with Misspelled Title PUT Update Existing Post PUT Update Non-Existing Post

Technical Assessment / Retrieve All Users

GET https://jsonplaceholder.typicode.com/users

Body Cookies Headers (26) Test Results

JSON Preview Visualize

200 OK · 355 ms · 2.94 KB · Save Response

```
[{"id": 1, "name": "Leanne Graham", "username": "Bret", "email": "Sincere@april.biz", "address": {"street": "Kulas Light", "suite": "Suite 123", "city": "Rathbury", "zipcode": "90660", "geo": {"lat": "42.4129", "lng": "71.0580"}}, "phone": "1-770-736-8031 x/5467", "website": "hildegard.org"}, {"id": 2, "name": "Ervin Howell", "username": "Antonette", "email": "Shanna@melissa.tv", "address": {"street": "Kulas Light", "suite": "Suite 123", "city": "Rathbury", "zipcode": "90660", "geo": {"lat": "42.4129", "lng": "71.0580"}}, "phone": "1-770-736-8031 x/5467", "website": "hildegard.org"}]
```

Bantucan, Adrian S.

The screenshot shows the Postman application interface. On the left, the sidebar displays the workspace structure: 'avalon harlow's Workspace' with a 'New' and 'Import' button, followed by a tree view of collections including 'My Collection' and 'Technical Assessment'. Under 'Technical Assessment', there are four requests: 'GET Retrieve All Users', 'GET Retrieve All Invalid Users', 'GET Retrieve Single User', and 'GET Retrieve Non-Existing User'. Below these are three POST requests: 'POST Create New Post', 'POST Create Post with Missing Title', and 'PUT Update Existing Post'. At the bottom of the sidebar, there are buttons for 'Online' and 'Console'.

The main area shows the details for the 'GET Retrieve All Users' request. The URL is <https://sonplaceholder.typicode.com/users>. The response status is 200 OK, with a duration of 355 ms and a size of 2.94 KB. The response body is a JSON array of user objects, each with properties like id, name, username, email, address, phone, website, company, catchPhrase, and bs. One example object is:

```
1  [
2    {
3      "id": 9,
4      "name": "Mallena Deloshant",
5      "username": "Deloshine",
6      "email": "Chaim_MoDermitt@dana.io",
7      "address": {
8        "street": "Dayna Park",
9        "suite": "Suite 449",
10       "city": "Burtholomewbury",
11       "zipcode": "76486-1269",
12       "geo": {
13         "lat": "24.4646",
14         "lng": "-168.8889"
15       }
16     },
17     "phone": "(776)976-6794 x41206",
18     "website": "conrad.com",
19     "company": {
20       "name": "Yost and Sons",
21       "catchPhrase": "Switchable contextually-based project",
22       "bs": "aggregate real-time technologies"
23     }
24   ],
25   {
26     "id": 10,
27     "name": "Loreen Bahringer",
28     "username": "BahringerLoreen",
29     "email": "Kaleena_Bahringer@dana.io",
30     "address": {
31       "street": "Kathleen Park",
32       "suite": "Suite 449",
33       "city": "Burtholomewbury",
34       "zipcode": "76486-1269",
35       "geo": {
36         "lat": "24.4646",
37         "lng": "-168.8889"
38       }
39     },
40     "phone": "(776)976-6794 x41206",
41     "website": "conrad.com",
42     "company": {
43       "name": "Yost and Sons",
44       "catchPhrase": "Switchable contextually-based project",
45       "bs": "aggregate real-time technologies"
46     }
47   ],
48   {
49     "id": 11,
50     "name": "Cecilia Klocko",
51     "username": "KlockoCecilia",
52     "email": "Kaleena_Klocko@dana.io",
53     "address": {
54       "street": "Kathleen Park",
55       "suite": "Suite 449",
56       "city": "Burtholomewbury",
57       "zipcode": "76486-1269",
58       "geo": {
59         "lat": "24.4646",
60         "lng": "-168.8889"
61       }
62     },
63     "phone": "(776)976-6794 x41206",
64     "website": "conrad.com",
65     "company": {
66       "name": "Yost and Sons",
67       "catchPhrase": "Switchable contextually-based project",
68       "bs": "aggregate real-time technologies"
69     }
70   ],
71   {
72     "id": 12,
73     "name": "Loreen Bahringer",
74     "username": "BahringerLoreen",
75     "email": "Kaleena_Bahringer@dana.io",
76     "address": {
77       "street": "Kathleen Park",
78       "suite": "Suite 449",
79       "city": "Burtholomewbury",
80       "zipcode": "76486-1269",
81       "geo": {
82         "lat": "24.4646",
83         "lng": "-168.8889"
84       }
85     },
86     "phone": "(776)976-6794 x41206",
87     "website": "conrad.com",
88     "company": {
89       "name": "Yost and Sons",
90       "catchPhrase": "Switchable contextually-based project",
91       "bs": "aggregate real-time technologies"
92     }
93   ],
94   {
95     "id": 13,
96     "name": "Loreen Bahringer",
97     "username": "BahringerLoreen",
98     "email": "Kaleena_Bahringer@dana.io",
99     "address": {
100      "street": "Kathleen Park",
101      "suite": "Suite 449",
102      "city": "Burtholomewbury",
103      "zipcode": "76486-1269",
104      "geo": {
105        "lat": "24.4646",
106        "lng": "-168.8889"
107      }
108    },
109    "phone": "(776)976-6794 x41206",
110    "website": "conrad.com",
111    "company": {
112      "name": "Yost and Sons",
113      "catchPhrase": "Switchable contextually-based project",
114      "bs": "aggregate real-time technologies"
115    }
116  ],
117  {
118    "id": 14,
119    "name": "Loreen Bahringer",
120    "username": "BahringerLoreen",
121    "email": "Kaleena_Bahringer@dana.io",
122    "address": {
123      "street": "Kathleen Park",
124      "suite": "Suite 449",
125      "city": "Burtholomewbury",
126      "zipcode": "76486-1269",
127      "geo": {
128        "lat": "24.4646",
129        "lng": "-168.8889"
130      }
131    },
132    "phone": "(776)976-6794 x41206",
133    "website": "conrad.com",
134    "company": {
135      "name": "Yost and Sons",
136      "catchPhrase": "Switchable contextually-based project",
137      "bs": "aggregate real-time technologies"
138    }
139  ],
140  {
141    "id": 15,
142    "name": "Loreen Bahringer",
143    "username": "BahringerLoreen",
144    "email": "Kaleena_Bahringer@dana.io",
145    "address": {
146      "street": "Kathleen Park",
147      "suite": "Suite 449",
148      "city": "Burtholomewbury",
149      "zipcode": "76486-1269",
150      "geo": {
151        "lat": "24.4646",
152        "lng": "-168.8889"
153      }
154    },
155    "phone": "(776)976-6794 x41206",
156    "website": "conrad.com",
157    "company": {
158      "name": "Yost and Sons",
159      "catchPhrase": "Switchable contextually-based project",
160      "bs": "aggregate real-time technologies"
161    }
162  ],
163  {
164    "id": 16,
165    "name": "Loreen Bahringer",
166    "username": "BahringerLoreen",
167    "email": "Kaleena_Bahringer@dana.io",
168    "address": {
169      "street": "Kathleen Park",
170      "suite": "Suite 449",
171      "city": "Burtholomewbury",
172      "zipcode": "76486-1269",
173      "geo": {
174        "lat": "24.4646",
175        "lng": "-168.8889"
176      }
177    },
178    "phone": "(866)493-6943 x148",
179    "website": "jacynthe.com",
180    "company": {
181      "name": "Abcronymy Group",
182      "catchPhrase": "Implemented secondary concept",
183      "bs": "e-enable extensible e-tailers"
184    }
185  },
186  {
187    "id": 17,
188    "name": "Mallena Deloshant",
189    "username": "Deloshine",
190    "email": "Chaim_MoDermitt@dana.io",
191    "address": {
192      "street": "Dayna Park",
193      "suite": "Suite 449",
194      "city": "Burtholomewbury",
195      "zipcode": "76486-1269",
196      "geo": {
197        "lat": "24.4646",
198        "lng": "-168.8889"
199      }
200    },
201    "phone": "(776)976-6794 x41206",
202    "website": "conrad.com",
203    "company": {
204      "name": "Yost and Sons",
205      "catchPhrase": "Switchable contextually-based project",
206      "bs": "aggregate real-time technologies"
207    }
208  },
209  {
210    "id": 18,
211    "name": "Loreen Bahringer",
212    "username": "BahringerLoreen",
213    "email": "Kaleena_Bahringer@dana.io",
214    "address": {
215      "street": "Kathleen Park",
216      "suite": "Suite 449",
217      "city": "Burtholomewbury",
218      "zipcode": "76486-1269",
219      "geo": {
220        "lat": "24.4646",
221        "lng": "-168.8889"
222      }
223    },
224    "phone": "(776)976-6794 x41206",
225    "website": "conrad.com",
226    "company": {
227      "name": "Yost and Sons",
228      "catchPhrase": "Switchable contextually-based project",
229      "bs": "aggregate real-time technologies"
230    }
231  },
232  {
233    "id": 19,
234    "name": "Loreen Bahringer",
235    "username": "BahringerLoreen",
236    "email": "Kaleena_Bahringer@dana.io",
237    "address": {
238      "street": "Kathleen Park",
239      "suite": "Suite 449",
240      "city": "Burtholomewbury",
241      "zipcode": "76486-1269",
242      "geo": {
243        "lat": "24.4646",
244        "lng": "-168.8889"
245      }
246    },
247    "phone": "(776)976-6794 x41206",
248    "website": "conrad.com",
249    "company": {
250      "name": "Yost and Sons",
251      "catchPhrase": "Switchable contextually-based project",
252      "bs": "aggregate real-time technologies"
253    }
254  },
255  {
256    "id": 20,
257    "name": "Loreen Bahringer",
258    "username": "BahringerLoreen",
259    "email": "Kaleena_Bahringer@dana.io",
260    "address": {
261      "street": "Kathleen Park",
262      "suite": "Suite 449",
263      "city": "Burtholomewbury",
264      "zipcode": "76486-1269",
265      "geo": {
266        "lat": "24.4646",
267        "lng": "-168.8889"
268      }
269    },
270    "phone": "(776)976-6794 x41206",
271    "website": "conrad.com",
272    "company": {
273      "name": "Yost and Sons",
274      "catchPhrase": "Switchable contextually-based project",
275      "bs": "aggregate real-time technologies"
276    }
277  },
278  {
279    "id": 21,
280    "name": "Loreen Bahringer",
281    "username": "BahringerLoreen",
282    "email": "Kaleena_Bahringer@dana.io",
283    "address": {
284      "street": "Kathleen Park",
285      "suite": "Suite 449",
286      "city": "Burtholomewbury",
287      "zipcode": "76486-1269",
288      "geo": {
289        "lat": "24.4646",
290        "lng": "-168.8889"
291      }
292    },
293    "phone": "(776)976-6794 x41206",
294    "website": "conrad.com",
295    "company": {
296      "name": "Yost and Sons",
297      "catchPhrase": "Switchable contextually-based project",
298      "bs": "aggregate real-time technologies"
299    }
300  },
301  {
302    "id": 30,
303    "name": "Loreen Bahringer",
304    "username": "BahringerLoreen",
305    "email": "Kaleena_Bahringer@dana.io",
306    "address": {
307      "street": "Kathleen Park",
308      "suite": "Suite 449",
309      "city": "Burtholomewbury",
310      "zipcode": "76486-1269",
311      "geo": {
312        "lat": "24.4646",
313        "lng": "-168.8889"
314      }
315    },
316    "phone": "(776)976-6794 x41206",
317    "website": "conrad.com",
318    "company": {
319      "name": "Yost and Sons",
320      "catchPhrase": "Switchable contextually-based project",
321      "bs": "aggregate real-time technologies"
322    }
323  },
324  {
325    "id": 31,
326    "name": "Loreen Bahringer",
327    "username": "BahringerLoreen",
328    "email": "Kaleena_Bahringer@dana.io",
329    "address": {
330      "street": "Kathleen Park",
331      "suite": "Suite 449",
332      "city": "Burtholomewbury",
333      "zipcode": "76486-1269",
334      "geo": {
335        "lat": "24.4646",
336        "lng": "-168.8889"
337      }
338    },
339    "phone": "(776)976-6794 x41206",
340    "website": "conrad.com",
341    "company": {
342      "name": "Yost and Sons",
343      "catchPhrase": "Switchable contextually-based project",
344      "bs": "aggregate real-time technologies"
345    }
346  },
347  {
348    "id": 32,
349    "name": "Loreen Bahringer",
350    "username": "BahringerLoreen",
351    "email": "Kaleena_Bahringer@dana.io",
352    "address": {
353      "street": "Kathleen Park",
354      "suite": "Suite 449",
355      "city": "Burtholomewbury",
356      "zipcode": "76486-1269",
357      "geo": {
358        "lat": "24.4646",
359        "lng": "-168.8889"
360      }
361    },
362    "phone": "(776)976-6794 x41206",
363    "website": "conrad.com",
364    "company": {
365      "name": "Yost and Sons",
366      "catchPhrase": "Switchable contextually-based project",
367      "bs": "aggregate real-time technologies"
368    }
369  },
370  {
371    "id": 33,
372    "name": "Loreen Bahringer",
373    "username": "BahringerLoreen",
374    "email": "Kaleena_Bahringer@dana.io",
375    "address": {
376      "street": "Kathleen Park",
377      "suite": "Suite 449",
378      "city": "Burtholomewbury",
379      "zipcode": "76486-1269",
380      "geo": {
381        "lat": "24.4646",
382        "lng": "-168.8889"
383      }
384    },
385    "phone": "(776)976-6794 x41206",
386    "website": "conrad.com",
387    "company": {
388      "name": "Yost and Sons",
389      "catchPhrase": "Switchable contextually-based project",
390      "bs": "aggregate real-time technologies"
391    }
392  },
393  {
394    "id": 34,
395    "name": "Loreen Bahringer",
396    "username": "BahringerLoreen",
397    "email": "Kaleena_Bahringer@dana.io",
398    "address": {
399      "street": "Kathleen Park",
400      "suite": "Suite 449",
401      "city": "Burtholomewbury",
402      "zipcode": "76486-1269",
403      "geo": {
404        "lat": "24.4646",
405        "lng": "-168.8889"
406      }
407    },
408    "phone": "(776)976-6794 x41206",
409    "website": "conrad.com",
410    "company": {
411      "name": "Yost and Sons",
412      "catchPhrase": "Switchable contextually-based project",
413      "bs": "aggregate real-time technologies"
414    }
415  },
416  {
417    "id": 35,
418    "name": "Loreen Bahringer",
419    "username": "BahringerLoreen",
420    "email": "Kaleena_Bahringer@dana.io",
421    "address": {
422      "street": "Kathleen Park",
423      "suite": "Suite 449",
424      "city": "Burtholomewbury",
425      "zipcode": "76486-1269",
426      "geo": {
427        "lat": "24.4646",
428        "lng": "-168.8889"
429      }
430    },
431    "phone": "(776)976-6794 x41206",
432    "website": "conrad.com",
433    "company": {
434      "name": "Yost and Sons",
435      "catchPhrase": "Switchable contextually-based project",
436      "bs": "aggregate real-time technologies"
437    }
438  },
439  {
440    "id": 36,
441    "name": "Loreen Bahringer",
442    "username": "BahringerLoreen",
443    "email": "Kaleena_Bahringer@dana.io",
444    "address": {
445      "street": "Kathleen Park",
446      "suite": "Suite 449",
447      "city": "Burtholomewbury",
448      "zipcode": "76486-1269",
449      "geo": {
450        "lat": "24.4646",
451        "lng": "-168.8889"
452      }
453    },
454    "phone": "(776)976-6794 x41206",
455    "website": "conrad.com",
456    "company": {
457      "name": "Yost and Sons",
458      "catchPhrase": "Switchable contextually-based project",
459      "bs": "aggregate real-time technologies"
460    }
461  },
462  {
463    "id": 37,
464    "name": "Loreen Bahringer",
465    "username": "BahringerLoreen",
466    "email": "Kaleena_Bahringer@dana.io",
467    "address": {
468      "street": "Kathleen Park",
469      "suite": "Suite 449",
470      "city": "Burtholomewbury",
471      "zipcode": "76486-1269",
472      "geo": {
473        "lat": "24.4646",
474        "lng": "-168.8889"
475      }
476    },
477    "phone": "(776)976-6794 x41206",
478    "website": "conrad.com",
479    "company": {
480      "name": "Yost and Sons",
481      "catchPhrase": "Switchable contextually-based project",
482      "bs": "aggregate real-time technologies"
483    }
484  },
485  {
486    "id": 38,
487    "name": "Loreen Bahringer",
488    "username": "BahringerLoreen",
489    "email": "Kaleena_Bahringer@dana.io",
490    "address": {
491      "street": "Kathleen Park",
492      "suite": "Suite 449",
493      "city": "Burtholomewbury",
494      "zipcode": "76486-1269",
495      "geo": {
496        "lat": "24.4646",
497        "lng": "-168.8889"
498      }
499    },
500    "phone": "(776)976-6794 x41206",
501    "website": "conrad.com",
502    "company": {
503      "name": "Yost and Sons",
504      "catchPhrase": "Switchable contextually-based project",
505      "bs": "aggregate real-time technologies"
506    }
507  },
508  {
509    "id": 39,
510    "name": "Loreen Bahringer",
511    "username": "BahringerLoreen",
512    "email": "Kaleena_Bahringer@dana.io",
513    "address": {
514      "street": "Kathleen Park",
515      "suite": "Suite 449",
516      "city": "Burtholomewbury",
517      "zipcode": "76486-1269",
518      "geo": {
519        "lat": "24.4646",
520        "lng": "-168.8889"
521      }
522    },
523    "phone": "(776)976-6794 x41206",
524    "website": "conrad.com",
525    "company": {
526      "name": "Yost and Sons",
527      "catchPhrase": "Switchable contextually-based project",
528      "bs": "aggregate real-time technologies"
529    }
530  },
531  {
532    "id": 40,
533    "name": "Loreen Bahringer",
534    "username": "BahringerLoreen",
535    "email": "Kaleena_Bahringer@dana.io",
536    "address": {
537      "street": "Kathleen Park",
538      "suite": "Suite 449",
539      "city": "Burtholomewbury",
540      "zipcode": "76486-1269",
541      "geo": {
542        "lat": "24.4646",
543        "lng": "-168.8889"
544      }
545    },
546    "phone": "(776)976-6794 x41206",
547    "website": "conrad.com",
548    "company": {
549      "name": "Yost and Sons",
550      "catchPhrase": "Switchable contextually-based project",
551      "bs": "aggregate real-time technologies"
552    }
553  },
554  {
555    "id": 41,
556    "name": "Loreen Bahringer",
557    "username": "BahringerLoreen",
558    "email": "Kaleena_Bahringer@dana.io",
559    "address": {
560      "street": "Kathleen Park",
561      "suite": "Suite 449",
562      "city": "Burtholomewbury",
563      "zipcode": "76486-1269",
564      "geo": {
565        "lat": "24.4646",
566        "lng": "-168.8889"
567      }
568    },
569    "phone": "(776)976-6794 x41206",
570    "website": "conrad.com",
571    "company": {
572      "name": "Yost and Sons",
573      "catchPhrase": "Switchable contextually-based project",
574      "bs": "aggregate real-time technologies"
575    }
576  },
577  {
578    "id": 42,
579    "name": "Loreen Bahringer",
580    "username": "BahringerLoreen",
581    "email": "Kaleena_Bahringer@dana.io",
582    "address": {
583      "street": "Kathleen Park",
584      "suite": "Suite 449",
585      "city": "Burtholomewbury",
586      "zipcode": "76486-1269",
587      "geo": {
588        "lat": "24.4646",
589        "lng": "-168.8889"
590      }
591    },
592    "phone": "(776)976-6794 x41206",
593    "website": "conrad.com",
594    "company": {
595      "name": "Yost and Sons",
596      "catchPhrase": "Switchable contextually-based project",
597      "bs": "aggregate real-time technologies"
598    }
599  },
599  {
600    "id": 60,
601    "name": "Loreen Bahringer",
602    "username": "BahringerLoreen",
603    "email": "Kaleena_Bahringer@dana.io",
604    "address": {
605      "street": "Kathleen Park",
606      "suite": "Suite 449",
607      "city": "Burtholomewbury",
608      "zipcode": "76486-1269",
609      "geo": {
610        "lat": "24.4646",
611        "lng": "-168.8889"
612      }
613    },
614    "phone": "(776)976-6794 x41206",
615    "website": "conrad.com",
616    "company": {
617      "name": "Yost and Sons",
618      "catchPhrase": "Switchable contextually-based project",
619      "bs": "aggregate real-time technologies"
620    }
621  },
621  {
622    "id": 61,
623    "name": "Loreen Bahringer",
624    "username": "BahringerLoreen",
625    "email": "Kaleena_Bahringer@dana.io",
626    "address": {
627      "street": "Kathleen Park",
628      "suite": "Suite 449",
629      "city": "Burtholomewbury",
630      "zipcode": "76486-1269",
631      "geo": {
632        "lat": "24.4646",
633        "lng": "-168.8889"
634      }
635    },
636    "phone": "(776)976-6794 x41206",
637    "website": "conrad.com",
638    "company": {
639      "name": "Yost and Sons",
640      "catchPhrase": "Switchable contextually-based project",
641      "bs": "aggregate real-time technologies"
642    }
642  },
642  {
643    "id": 62,
644    "name": "Loreen Bahringer",
645    "username": "BahringerLoreen",
646    "email": "Kaleena_Bahringer@dana.io",
647    "address": {
648      "street": "Kathleen Park",
649      "suite": "Suite 449",
650      "city": "Burtholomewbury",
651      "zipcode": "76486-1269",
652      "geo": {
653        "lat": "24.4646",
654        "lng": "-168.8889"
655      }
656    },
657    "phone": "(776)976-6794 x41206",
658    "website": "conrad.com",
659    "company": {
660      "name": "Yost and Sons",
661      "catchPhrase": "Switchable contextually-based project",
662      "bs": "aggregate real-time technologies"
663    }
663  },
663  {
664    "id": 63,
665    "name": "Loreen Bahringer",
666    "username": "BahringerLoreen",
667    "email": "Kaleena_Bahringer@dana.io",
668    "address": {
669      "street": "Kathleen Park",
670      "suite": "Suite 449",
671      "city": "Burtholomewbury",
672      "zipcode": "76486-1269",
673      "geo": {
674        "lat": "24.4646",
675        "lng": "-168.8889"
676      }
677    },
678    "phone": "(776)976-6794 x41206",
679    "website": "conrad.com",
680    "company": {
681      "name": "Yost and Sons",
682      "catchPhrase": "Switchable contextually-based project",
683      "bs": "aggregate real-time technologies"
684    }
684  },
684  {
685    "id": 64,
686    "name": "Loreen Bahringer",
687    "username": "BahringerLoreen",
688    "email": "Kaleena_Bahringer@dana.io",
689    "address": {
690      "street": "Kathleen Park",
691      "suite": "Suite 449",
692      "city": "Burtholomewbury",
693      "zipcode": "76486-1269",
694      "geo": {
695        "lat": "24.4646",
696        "lng": "-168.8889"
697      }
698    },
699    "phone": "(776)976-6794 x41206",
700    "website": "conrad.com",
701    "company": {
702      "name": "Yost and Sons",
703      "catchPhrase": "Switchable contextually-based project",
704      "bs": "aggregate real-time technologies"
705    }
705  },
705  {
706    "id": 65,
707    "name": "Loreen Bahringer",
708    "username": "BahringerLoreen",
709    "email": "Kaleena_Bahringer@dana.io",
710    "address": {
711      "street": "Kathleen Park",
712      "suite": "Suite 449",
713      "city": "Burtholomewbury",
714      "zipcode": "76486-1269",
715      "geo": {
716        "lat": "24.4646",
717        "lng": "-168.8889"
718      }
719    },
720    "phone": "(776)976-6794 x41206",
721    "website": "conrad.com",
722    "company": {
723      "name": "Yost and Sons",
724      "catchPhrase": "Switchable contextually-based project",
725      "bs": "aggregate real-time technologies"
726    }
726  },
726  {
727    "id": 66,
728    "name": "Loreen Bahringer",
729    "username": "BahringerLoreen",
730    "email": "Kaleena_Bahringer@dana.io",
731    "address": {
732      "street": "Kathleen Park",
733      "suite": "Suite 449",
734      "city": "Burtholomewbury",
735      "zipcode": "76486-1269",
736      "geo": {
737        "lat": "24.4646",
738        "lng": "-168.8889"
739      }
740    },
741    "phone": "(776)976-6794 x41206",
742    "website": "conrad.com",
743    "company": {
744      "name": "Yost and Sons",
745      "catchPhrase": "Switchable contextually-based project",
746      "bs": "aggregate real-time technologies"
747    }
747  },
747  {
748    "id": 67,
749    "name": "Loreen Bahringer",
750    "username": "BahringerLoreen",
751    "email": "Kaleena_Bahringer@dana.io",
752    "address": {
753      "street": "Kathleen Park",
754      "suite": "Suite 449",
755      "city": "Burtholomewbury",
756      "zipcode": "76486-1269",
757      "geo": {
758        "lat": "24.4646",
759        "lng": "-168.8889"
760      }
761    },
762    "phone": "(776)976-6794 x41206",
763    "website": "conrad.com",
764    "company": {
765      "name": "Yost and Sons",
766      "catchPhrase": "Switchable contextually-based project",
767      "bs": "aggregate real-time technologies"
768    }
768  },
768  {
769    "id": 68,
770    "name": "Loreen Bahringer",
771    "username": "BahringerLoreen",
772    "email": "Kaleena_Bahringer@dana.io",
773    "address": {
774      "street": "Kathleen Park",
775      "suite": "Suite 449",
776      "city": "Burtholomewbury",
777      "zipcode": "76486-1269",
778      "geo": {
779        "lat": "24.4646",
780        "lng": "-168.8889"
781      }
782    },
783    "phone": "(776)976-6794 x41206",
784    "website": "conrad.com",
785    "company": {
786      "name": "Yost and Sons",
787      "catchPhrase": "Switchable contextually-based project",
788      "bs": "aggregate real-time technologies"
789    }
789  },
789  {
790    "id": 69,
791    "name": "Loreen Bahringer",
792    "username": "BahringerLoreen",
793    "email": "Kaleena_Bahringer@dana.io",
794    "address": {
795      "street": "Kathleen Park",
796      "suite": "Suite 449",
797      "city": "Burtholomewbury",
798      "zipcode": "76486-1269",
799      "geo": {
800        "lat": "24.4646",
801        "lng": "-168.8889"
802      }
803    },
804    "phone": "(776)976-6794 x41206",
805    "website": "conrad.com",
806    "company": {
807      "name": "Yost and Sons",
808      "catchPhrase": "Switchable contextually-based project",
809      "bs": "aggregate real-time technologies"
810    }
810  },
810  {
811    "id": 70,
812    "name": "Loreen Bahringer",
813    "username": "BahringerLoreen",
814    "email": "Kaleena_Bahringer@dana.io",
815    "address": {
816      "street": "Kathleen Park",
817      "suite": "Suite 449",
818      "city": "Burtholomewbury",
819      "zipcode": "76486-1269",
820      "geo": {
821        "lat": "24.4646",
822        "lng": "-168.8889"
823      }
824    },
825    "phone": "(776)976-6794 x41206",
826    "website": "conrad.com",
827    "company": {
828      "name": "Yost and Sons",
829      "catchPhrase": "Switchable contextually-based project",
830      "bs": "aggregate real-time technologies"
831    }
831  },
831  {
832    "id": 71,
833    "name": "Loreen Bahringer",
834    "username": "BahringerLoreen",
835    "email": "Kaleena_Bahringer@dana.io",
836    "address": {
837      "street": "Kathleen Park",
838      "suite": "Suite 449",
839      "city": "Burtholomewbury",
840      "zipcode": "76486-1269",
841      "geo": {
842        "lat": "24.4646",
843        "lng": "-168.8889"
844      }
845    },
846    "phone": "(776)976-6794 x41206",
847    "website": "conrad.com",
848    "company": {
849      "name": "Yost and Sons",
850      "catchPhrase": "Switchable contextually-based project",
851      "bs": "aggregate real-time technologies"
852    }
852  },
852  {
853    "id": 72,
854    "name": "Loreen Bahringer",
855    "username": "BahringerLoreen",
856    "email": "Kaleena_Bahringer@dana.io",
857    "address": {
858      "street": "Kathleen Park",
859      "suite": "Suite 449",
860      "city": "Burtholomewbury",
861      "zipcode": "76486-1269",
862      "geo": {
863        "lat": "24.4646",
864        "lng": "-168.8889"
865      }
866    },
867    "phone": "(776)976-6794 x41206",
868    "website": "conrad.com",
869    "company": {
870      "name": "Yost and Sons",
871      "catchPhrase": "Switchable contextually-based project",
872      "bs": "aggregate real-time technologies"
873    }
873  },
873  {
874    "id": 73,
875    "name": "Loreen Bahringer",
876    "username": "BahringerLoreen",
877    "email": "Kaleena_Bahringer@dana.io",
878    "address": {
879      "street": "Kathleen Park",
880      "suite": "Suite 449",
881      "city": "Burtholomewbury",
882      "zipcode": "76486-1269",
883      "geo": {
884        "lat": "24.4646",
885        "lng": "-168.8889"
886      }
887    },
888    "phone": "(776)976-6794 x41206",
889    "website": "conrad.com",
890    "company": {
891      "name": "Yost and Sons",
892      "catchPhrase": "Switchable contextually-based project",
893      "bs": "aggregate real-time technologies"
894    }
894  },
894  {
895    "id": 74,
896    "name": "Loreen Bahringer",
897    "username": "BahringerLoreen",
898    "email": "Kaleena_Bahringer@dana.io",
899    "address": {
900      "street": "Kathleen Park",
901      "suite": "Suite 449",
902      "city": "Burtholomewbury",
903      "zipcode": "76486-1269",
904      "geo": {
905        "lat": "24.4646",
906        "lng": "-168.8889"
907      }
908    },
909    "phone": "(776)976-6794 x41206",
910    "website": "conrad.com",
911    "company": {
912      "name": "Yost and Sons",
913      "catchPhrase": "Switchable contextually-based project",
914      "bs": "aggregate real-time technologies"
915    }
915  },
915  {
916    "id": 75,
917    "name": "Loreen Bahringer",
918    "username": "BahringerLoreen",
919    "email": "Kaleena_Bahringer@dana.io",
920    "address": {
921      "street": "Kathleen Park",
922      "suite": "Suite 449",
923      "city": "Burtholomewbury",
924      "zipcode": "76486-1269",
925      "geo": {
926        "lat": "24.4646",
927        "lng": "-168.8889"
928      }
929    },
930    "phone": "(776)976-6794 x41206",
931    "website": "conrad.com",
932    "company": {
933      "name": "Yost and Sons",
934      "catchPhrase": "Switchable contextually-based project",
935      "bs": "aggregate real-time technologies"
936    }
936  },
936  {
937    "id": 76,
938    "name": "Loreen Bahringer",
939    "username": "BahringerLoreen",
940    "email": "Kaleena_Bahringer@dana.io",
941    "address": {
942      "street": "Kathleen Park",
943      "suite": "Suite 449",
944      "city": "Burtholomewbury",
945      "zipcode": "76486-1269",
946      "geo": {
947        "lat": "24.4646",
948        "lng": "-168.8889"
949      }
950    },
951    "phone": "(776)976-6794 x41206",
952    "website": "conrad.com",
953    "company": {
954      "name": "Yost and Sons",
9
```

The screenshot shows the Postman application interface. The left sidebar displays 'avalon harlow's Workspace' with various collections and environments. The main workspace shows a collection named 'Technical Assessment' with several API endpoints listed under 'My Collection'. One endpoint, 'GET Retrieve All Users', is selected and expanded. The request details pane shows a GET request to 'https://ponyplaceholder.typicode.com/users'. The response pane displays a JSON array of user objects, each with properties like id, name, email, address, and company information. The status bar at the bottom indicates a 200 OK response with 355 ms duration and 2.94 KB size.

```
201 {
  "id": 1,
  "name": "Clementina DuBouef",
  "email": "Clementina.Dubouef@example.com",
  "address": {
    "street": "Kattie Turnpike",
    "suite": "Suite 198",
    "city": "Lebsackbury",
    "zipcode": "31498-2261",
    "geo": {
      "lat": "-38.2396",
      "lng": "87.2233"
    }
  },
  "phone": "924-648-3884",
  "website": "ambizze.net",
  "company": {
    "name": "Hoeges LLC",
    "catchPhrase": "Centralized empowering task-force",
    "bs": "target end-to-end models"
  }
}
```

-GET Retrieve All Invalid Users

The screenshot shows the Postman interface with a collection named "Technical Assessment". A GET request is made to `https://jsonplaceholder.typicode.com/userzzz`. The response status is 404 Not Found, indicating the resource was not found.

-GET Retrieve Single User

The screenshot shows the Postman interface with a collection named "Technical Assessment". A GET request is made to `https://jsonplaceholder.typicode.com/users/1`. The response status is 200 OK, and the JSON response body is displayed, containing detailed information about a user named Leanne Graham.

```

1 [
2   {
3     "id": 1,
4     "name": "Leanne Graham",
5     "username": "Bret",
6     "email": "Sincere@april.biz",
7     "address": {
8       "street": "Kulas Light",
9       "suite": "Apt. 550",
10      "city": "Gwenborough",
11      "zipcode": "92998-3874",
12      "geo": {
13        "lat": "-37.3159",
14        "lng": "81.1496"
15      }
16    },
17    "phone": "1-770-736-8031 x56442",
18    "website": "hildegard.org",
19    "company": {
20      "name": "Romaguera-Crona",
21      "catchPhrase": "Multi-layered client-server neural-net",
22      "bs": "harness real-time e-markets"
23    }
24  ]
  
```

The detailed user information is as follows:

id	name	username	email	address	phone	website	company
1	Leanne Graham	Bret	Sincere@april.biz	Kulas Light, Apt. 550, Gwenborough, 92998-3874	1-770-736-8031 x56442	hildegard.org	Romaguera-Crona Multi-layered client-server neural-net harness real-time e-markets

-GET Retrieve Non-Existing User

The screenshot shows the Postman interface with a collection named "Technical Assessment". A specific request titled "GET Retrieve Non-Existing User" is selected. The URL is set to `https://jsonplaceholder.typicode.com/users/999`. The response status is 404 Not Found, indicating the user does not exist.

-POST Create New Post

The screenshot shows the Postman interface with a collection named "Technical Assessment". A specific request titled "POST Create New Post" is selected. The URL is set to `https://jsonplaceholder.typicode.com/posts`. The request body contains JSON data for a new post:

```

1 {
2   "title": "Hello world! This is my new post",
3   "body": "Post content",
4   "userId": 1
5 }
6
7
  
```

The response status is 201 Created, indicating the post was successfully created.

-POST Create Post with Missing Link

The screenshot shows the Postman interface with a collection named "Technical Assessment". A POST request titled "Create Post with Missing Title" is selected. The request URL is <https://jsonplaceholder.typicode.com/posts>. The request body is set to raw JSON:

```

1 [ {
2   "body": "Post content",
3   "userId": 1
4 }
5 ]

```

The response status is 201 Created, and the response body is:

```

1 {
2   "body": "Post content",
3   "userId": 1,
4   "id": 1
5 }

```

The Headers tab shows the Content-Type header is application/json.

-PUT Update Existing Post

The screenshot shows the Postman interface with a collection named "Technical Assessment". A PUT request titled "Update Existing Post" is selected. The request URL is <https://jsonplaceholder.typicode.com/posts/1>. The request body is set to raw JSON:

```

1 { "title": "Updated Title", "body": "Updated content", "userId": 1 }
2

```

The response status is 200 OK, and the response body is:

```

1 {
2   "title": "Updated Title",
3   "body": "updated content",
4   "userId": 1,
5   "id": 1
6 }

```

Bantucan, Adrian S.

The screenshot shows the Postman interface with a collection named "avalon harlow's Workspace". A "PUT Update Existing Post" request is selected. The URL is <https://jsonplaceholder.typicode.com/posts/1>. The Headers tab shows "Content-Type: application/json". The Body tab contains the following JSON payload:

```
1 {
2   "title": "Updated Title",
3   "body": "updated content",
4   "userId": 1,
5   "id": 1
6 }
```

The response status is 200 OK with a response time of 436 ms and a size of 1.19 KB. The response body is identical to the request body.

-PUT Update Non-Existing Post

The screenshot shows the Postman interface with a collection named "avalon harlow's Workspace". A "PUT Update Non-Existing Post" request is selected. The URL is <https://jsonplaceholder.typicode.com/posts/999>. The Headers tab shows "Content-Type: application/json". The Body tab contains the following JSON payload:

```
1 [ { "title": "Test", "body": "test", "userId": 1 } ]
```

The response status is 500 Internal Server Error with a response time of 382 ms and a size of 1.86 KB. The response body shows an error stack trace:

```
1 TypeError: Cannot read properties of undefined (reading 'id')
2 at update (/app/node_modules/json-server/lib/server/router/plural1.js:262:24)
3 at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:8)
4 at next (/app/node_modules/express/lib/router/route.js:131:14)
5 at Route.dispatch (/app/node_modules/express/lib/router/route.js:112:3)
6 at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:8)
7 at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/index.js:281:22)
8 at /app/node_modules/express/lib/router/index.js:360:14)
9 at param (/app/node_modules/express/lib/router/index.js:360:14)
10 at param (/app/node_modules/express/lib/router/index.js:360:14)
11 at Function.process_params (/app/node_modules/express/lib/router/index.js:410:13)
```

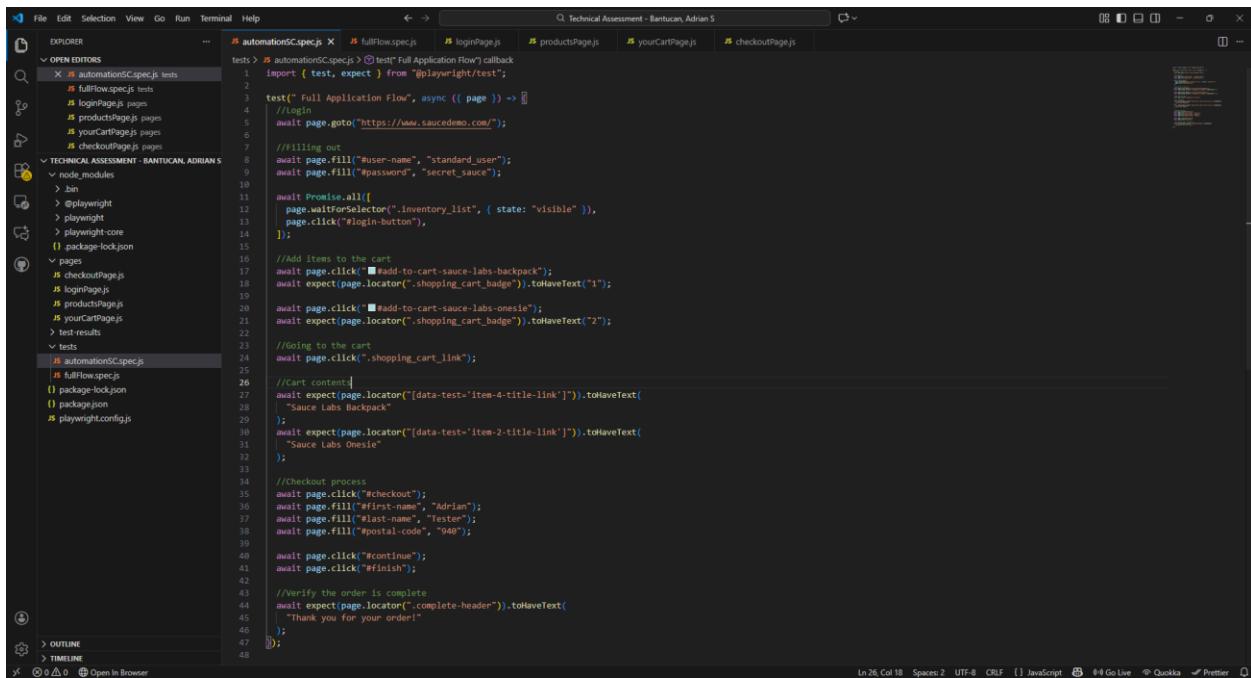
The Headers tab shows "Content-Type: application/json".

V. Automation Summary

Two approaches were used for automation. The straightforward approach consolidated all test modules into a single file (automationSC.spec.js), making it simple to run end-to-end tests quickly. The Page Object Model (POM) approach (fullFlow.spec.js) organized tests into separate pages and modules, following the structure I practiced during my internship. This approach improves maintainability, readability, reduces code duplication, and allows easier updates when the application changes.

The automation scripts cover login, product selection, and checkout workflows. To run the tests, use the following commands from the project root:

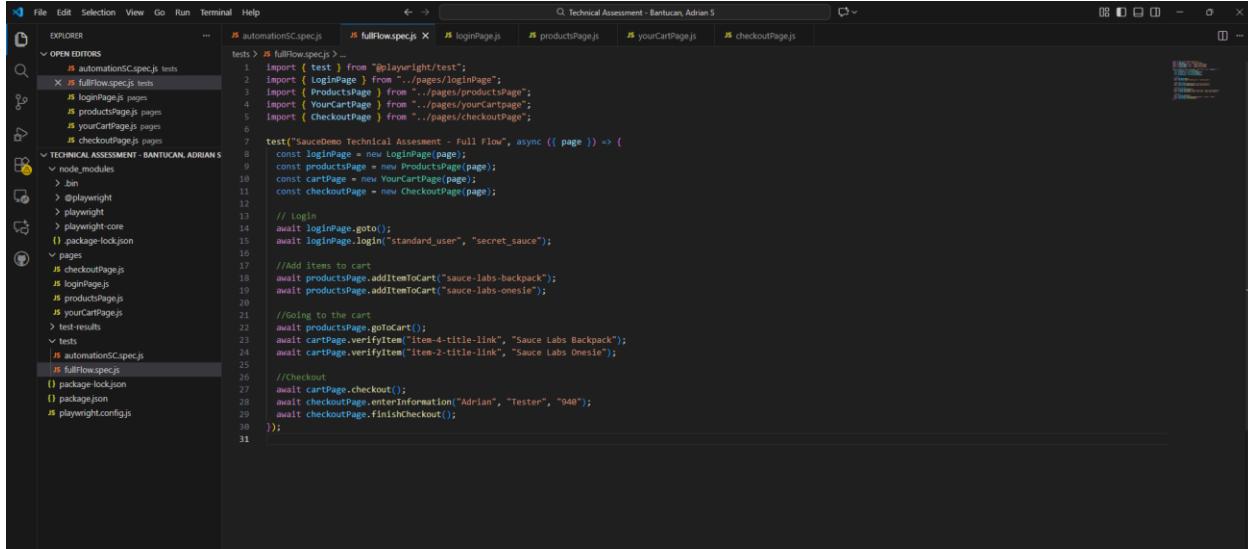
Screenshots:



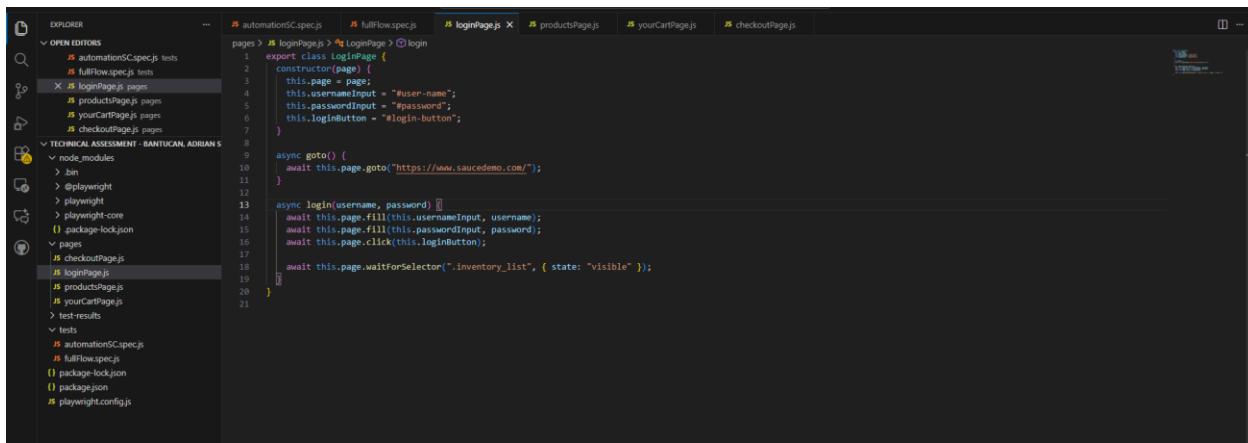
A screenshot of a code editor (Visual Studio Code) displaying a file structure and a large block of JavaScript code. The file structure on the left shows a folder named 'TECHNICAL ASSESSMENT - BANTUCAN, ADRIAN S' containing 'node_modules', 'bin', '@playwright', 'playwright', 'playwright-core', 'package-lock.json', 'pages' (containing 'checkoutPage.js', 'loginPage.js', 'productsPage.js', 'yourCartPage.js'), 'test-results' (containing 'automationSCSpec.js', 'fullFlow.spec.js'), 'package.json', and 'playwright.config.js'. The main editor area contains the 'fullFlow.spec.js' file, which is a Playwright test script. It imports the 'expect' module and defines a test function for a 'Full Application Flow'. The script uses various Playwright methods like 'page.goto', 'page.fill', 'page.click', and 'page.waitForSelector' to interact with a Sauce Labs demo website. The code is well-organized and follows a clear structure.

```
File Edit Selection View Go Run Terminal Help File Explorer OPEN EDITORS automationSCSpec.js fullFlow.spec.js loginPage.js productsPage.js yourCartPage.js checkoutPage.js tests > automationSCSpec.js > fullFlow.spec.js > loginPage.js > productsPage.js > yourCartPage.js > checkoutPage.js TECHNICAL ASSESSMENT - BANTUCAN, ADRIAN S node_modules bin @playwright playwright playwright-core package-lock.json pages checkoutPage.js loginPage.js productsPage.js yourCartPage.js test-results automationSCSpec.js fullFlow.spec.js package.json playwright.config.js // Full Application Flow, async ( page ) => { import { test, expect } from '@playwright/test'; //Login await page.goto('https://www.saucedemo.com/'); //Filling out await page.fill('#user-name', 'standard_user'); await page.fill('#password', 'secret_sauce'); await Promise.all([ page.waitForSelector('.inventory_list', { state: "visible" }), page.click("#login-button"), ]); //Add items to the cart await page.click('#add-to-cart-sauce-labs-backpack'); await expect(page.locator(".shopping_cart_badge")).toHaveText("1"); await page.click('#add-to-cart-sauce-labs-onesie'); await expect(page.locator(".shopping_cart_badge")).toHaveText("2"); //Going to the cart await page.click(".shopping_cart_link"); //Cart content await expect(page.locator("[data-test='item-4-title-link']")).toHaveText("Sauce Labs Backpack"); await expect(page.locator("[data-test='item-2-title-link']")).toHaveText("Sauce Labs Onesie"); //Checkout process await page.click("#checkout"); await page.fill("#first-name", "Adrian"); await page.fill("#last-name", "tester"); await page.fill("#postal-code", "040"); await page.click("#continue"); await page.click("#finish"); //Verify the order is complete await expect(page.locator(".complete-header")).toHaveText("Thank you for your order!"); }; };
```

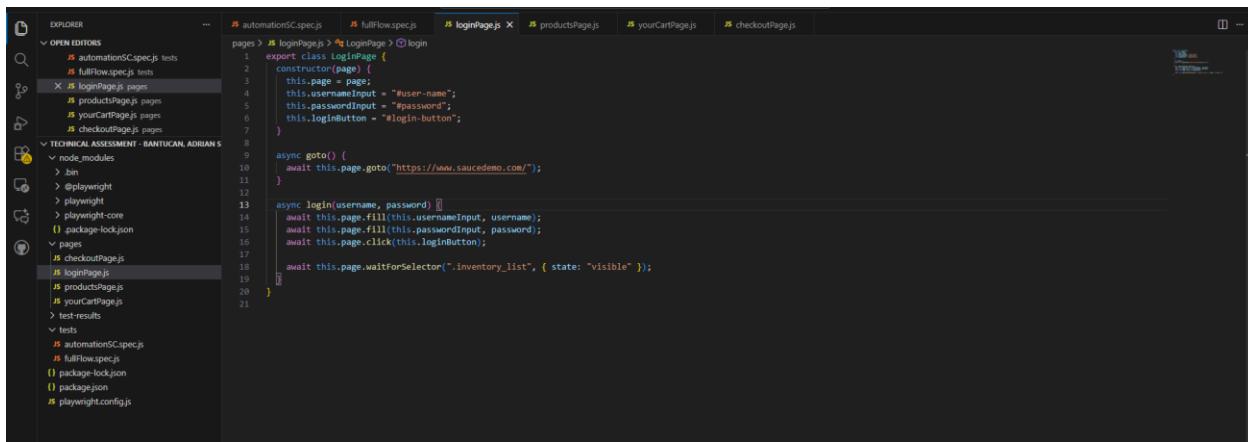
Bantucan, Adrian S.



```
File Edit Selection View Go Run Terminal Help
automationSCspec.js fullFlow.spec.js loginPage.js productsPage.js yourCartPage.js checkoutPage.js
EXPLORER
OPEN EDITORS
automationSCspec.js tests
fullFlow.spec.js tests
loginPage.js pages
productsPage.js pages
yourCartPage.js pages
checkoutPage.js pages
node_modules
bin
playwright
playwright-core
package-lock.json
pages
checkoutPage.js
loginPage.js
productsPage.js
yourCartPage.js
test-results
automationSCspec.js
fullFlow.spec.js
package-lock.json
package.json
playwright.config.js
TECHNICAL ASSESSMENT - BANTUCAN, ADRIANS
tests > fullFlow.spec.js ...
1 import { test } from '@playwright/test';
2 import { LoginPage } from './page/loginPage';
3 import { productsPage } from './page/productsPage';
4 import { YourCartPage } from './page/yourCartPage';
5 import { CheckoutPage } from './page/checkoutPage';
6
7 test('SauceDemo Technical Assessment - Full Flow', async ({ page }) => {
8     const loginPage = new LoginPage(page);
9     const productsPage = new ProductsPage(page);
10    const cartPage = new YourCartPage(page);
11    const checkoutPage = new CheckoutPage(page);
12
13    // Login
14    await loginPage.goto();
15    await loginPage.login("standard_user", "secret_sauce");
16
17    //Add items to cart
18    await productsPage.addItemToCart("sauce-labs-backpack");
19    await productsPage.addItemToCart("sauce-labs-onesie");
20
21    //Going to the cart
22    await productsPage.goToCart();
23    await cartPage.verifyItem("item-4-title-link", "Sauce Labs Backpack");
24    await cartPage.verifyItem("item-2-title-link", "Sauce Labs Onesie");
25
26    //Checkout
27    await cartPage.checkout();
28    await checkoutPage.enterInformation("Adrian", "Tester", "940");
29    await checkoutPage.finishInCheckout();
30 })
31
```



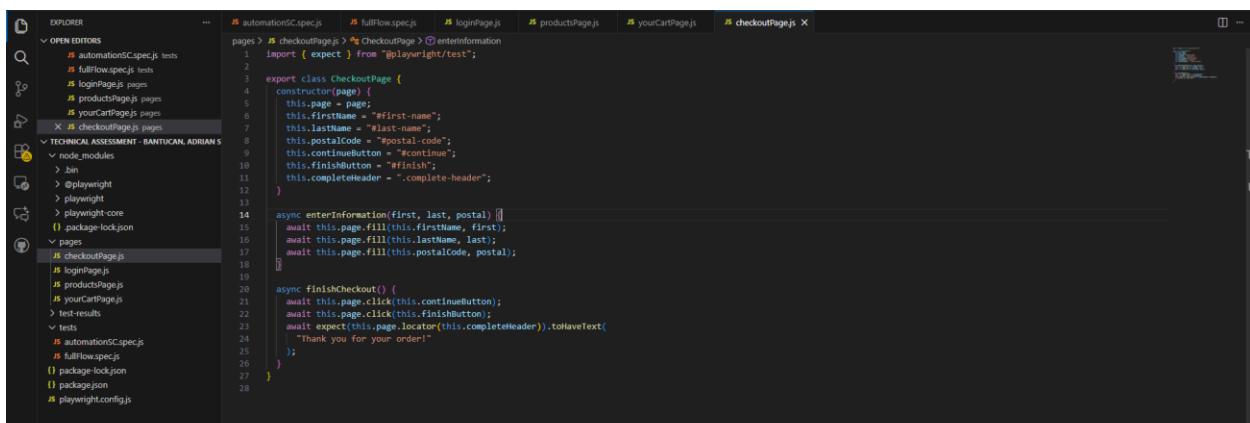
```
File Edit Selection View Go Run Terminal Help
automationSCspec.js fullFlow.spec.js loginPage.js productsPage.js yourCartPage.js checkoutPage.js
EXPLORER
OPEN EDITORS
automationSCspec.js tests
fullFlow.spec.js tests
loginPage.js pages
productsPage.js pages
yourCartPage.js pages
checkoutPage.js pages
node_modules
bin
playwright
playwright-core
package-lock.json
pages
checkoutPage.js
loginPage.js
productsPage.js
yourCartPage.js
test-results
automationSCspec.js
fullFlow.spec.js
package-lock.json
package.json
playwright.config.js
TECHNICAL ASSESSMENT - BANTUCAN, ADRIANS
pages > loginPage.js ...
1 export class LoginPage {
2     constructor(page) {
3         this.page = page;
4         this.usernameInput = "#user-name";
5         this.passwordInput = "#password";
6         this.loginButton = "#login-button";
7     }
8
9     async goto() {
10         await this.page.goto("https://www.saucedemo.com/");
11     }
12
13     async login(username, password) {
14         await this.page.fill(this.usernameInput, username);
15         await this.page.fill(this.passwordInput, password);
16         await this.page.click(this.loginButton);
17         await this.page.waitForSelector(".inventory_list", { state: "visible" });
18     }
19 }
20 }
```



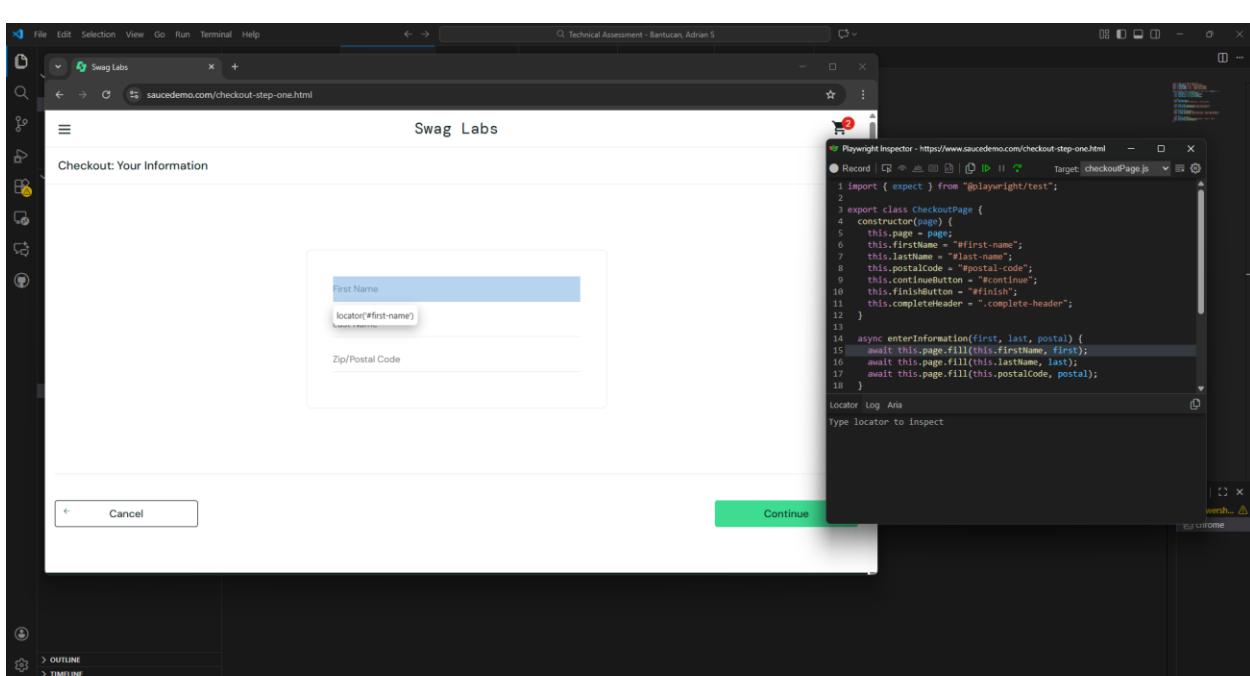
```
File Edit Selection View Go Run Terminal Help
automationSCspec.js fullFlow.spec.js loginPage.js productsPage.js yourCartPage.js checkoutPage.js
EXPLORER
OPEN EDITORS
automationSCspec.js tests
fullFlow.spec.js tests
loginPage.js pages
productsPage.js pages
yourCartPage.js pages
checkoutPage.js pages
node_modules
bin
playwright
playwright-core
package-lock.json
pages
checkoutPage.js
loginPage.js
productsPage.js
yourCartPage.js
test-results
automationSCspec.js
fullFlow.spec.js
package-lock.json
package.json
playwright.config.js
TECHNICAL ASSESSMENT - BANTUCAN, ADRIANS
pages > loginPage.js ...
1 export class LoginPage {
2     constructor(page) {
3         this.page = page;
4         this.usernameInput = "#user-name";
5         this.passwordInput = "#password";
6         this.loginButton = "#login-button";
7     }
8
9     async goto() {
10         await this.page.goto("https://www.saucedemo.com/");
11     }
12
13     async login(username, password) {
14         await this.page.fill(this.usernameInput, username);
15         await this.page.fill(this.passwordInput, password);
16         await this.page.click(this.loginButton);
17         await this.page.waitForSelector(".inventory_list", { state: "visible" });
18     }
19 }
20 }
```

Bantucan, Adrian S.

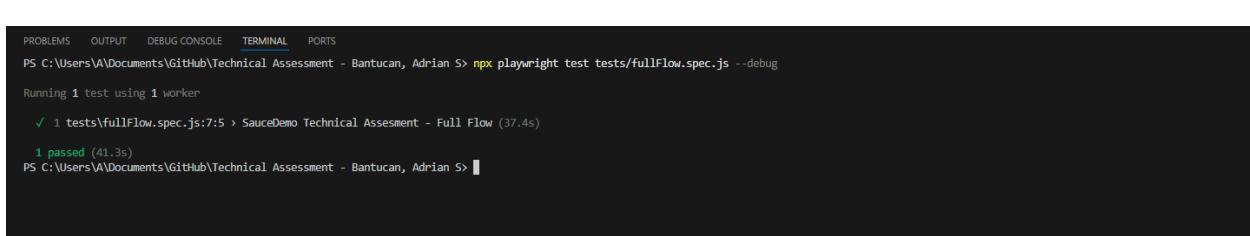
VS Code Editor Screenshots:

Browser Screenshot:



Terminal Screenshot:



VI. Bug Summary

Several bugs were identified during testing. Major bugs included checkout bypass through URL manipulation, completing checkout with an empty cart, session inconsistencies across browser tabs, and the ability to resubmit completed orders. Minor bugs primarily involved UI issues, such as oversized clickable areas. Each bug was documented with steps to reproduce, expected behavior, actual results, and suggested improvements. The severity and priority levels provide guidance for development teams to address critical issues first.

VII. Conclusion

Overall, the QA assessment confirmed that the application's key functionalities, including login, product selection, and checkout, are mostly functioning as intended. The web and API tests validated standard workflows and identified areas for improvement. Automation scripts successfully replicated manual tests, offering a reliable framework for future regression testing. The documented bugs highlight opportunities to enhance both functionality and user experience. This testing effort provides confidence in the application's stability and delivers actionable insights for the development team.

Note: If the video proof in the bug report is not working, please contact me.