

Course: IT202-008-S2025

Assignment: IT202 Milestone 2

Student: Gori H. (gs658)

Status: Submitted | Worksheet Progress: 100%

Potential Grade: 10.00/10.00 (100.00%)

Received Grade: 0.00/10.00 (0.00%)

Grading Link: <https://learn.ethereallab.app/assignment/v3/IT202-008-S2025/it202-milestone-2/grading/gs658>

Instructions

1. Refer to Milestone2 of this doc:

<https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88EWVwfo/view>

2. Ensure you read all instructions and objectives before starting.
3. Ensure you've gone through each lesson related to this Milestone
4. Switch to the Milestone2 branch

1. `git checkout Milestone2` (ensure proper starting branch)
2. `git pull origin Milestone2` (ensure history is up to date)

5. Fill out the below worksheet

- Ensure there's a comment with your UCID, date, and brief summary of the snippet in each screenshot
- Ensure proper styling is applied to each page
- Ensure there are no visible technical errors; only user-friendly messages are allowed

6. Once finished, click "Submit and Export"

7. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github

1. `git add .`
2. `git commit -m "adding PDF"`
3. `git push origin Milestone2`
4. On Github merge the pull request from Milestone2 to dev
5. On Github create a pull request from dev to prod and immediately merge. (This will trigger the prod deploy to make the heroku prod links work)

8. Upload the same PDF to Canvas

9. Sync Local

10. `git checkout dev`

11. `git pull origin dev`

100%

Section #1: (2 pts.) Data

100%

Task #1 (0.50 pts.) - API Data Table

Combo Task:

Weight: 25%

Objective: *API Data Table*

≡, Image Prompt

Weight: 34%

Details:

- Show the table(s) structure of the table made to hold your API
 - There should be at least 3 properties/fields beyond `id`, `created`, `modified`, and respective `is_api` or `api_id` columns
 - Logical types and constraints should be set
 - Show at least a few records from each API table (samples should include both API fetched data and custom entries noted by an `api_id` or `is_api` column)

movie table structure



Saved: 5/7/2025 11:35:21 PM

Url Prompt

Weight: 33%

Details:

- Add a direct link to your sql file from Github (should end in .sql)

URL #1

<https://github.com/goribau/qs658->



<https://github.com/goribantu/gs658>

IT2018Milestone2/public_html/project/sql/006_create_table_movies.sql



Saved: 5/7/2025 11:35:21 PM

Text Prompt

Weight: 33%

Details:

- Briefly note what each field/property represents

Your Response:

Id is a unique identifier for each movie (primary key), title is the movie's name, year is when the movie was released, rating is rating of the movie (out of 5), is_api indicates whether the movie data came from the API (1) or was added manually (0), and plot gives a brief description of what the movie is about.



Saved: 5/7/2025 11:35:21 PM

100%

Task #2 (0.50 pts.) - Data Transformation

Combo Task:

Weight: 25%

Objective: *Data Transformation*

Image Prompt

Weight: 40%

Details:

- Show the code that handles the data transformation/conversion (from the API)
- Show a Heroku dev example of API getting fetched and loaded into the DB

```
/*  
 * error_log("OMDB Response: " . var_export($results, true));  
 */  
// Handles data transformation/conversion of the API call we just received.  
// It decodes JSON string into a PHP associative array using json_decode  
// and safely pulls the "Plot" field, which says "not available". This data  
// is then used to insert into Movies table.  
if ($results["status"] == 200 AND isset($results["response"])) {  
    $results = json_decode($results["response"], true);  
    $is_api = 1;  
    $plot = $results["Plot"] ?? "Plot not available";  
} else {  
    $results = [];  
}  
*/
```

code that handles data transformation/conversion from API





api gets fetched



SELECT * FROM 'Movies' LIMIT 100						
	ID	Title	Year	Rating	Total	Created
>	6	Avengers	2020	5	2025-05-04 23:00:01	
>	7	Saw	2009	4	2025-05-04 23:00:35	
>	8	Ghostman	2021	3	2025-05-05 00:08:47	
>	10	The Bee Movie	2022	2	2025-05-05 02:15:32	
>	11	Sonic the Hedgehog 2	2024	3.5	2025-05-05 02:18:05	

loaded into db



Saved: 5/7/2025 11:59:28 PM

Url Prompt

Weight: 20%

Details:

- Include Heroku prod link to API fetch page
- Include pull request link for this feature

URL #1

[https://gs658-it202-008-
prod-96409305ed03.herokuapp.com/api/movie-
search.php](https://gs658-it202-008-prod-96409305ed03.herokuapp.com/api/movie-search.php)



URL

<https://gs658-it202-008-prod-96409305ed03.herokuapp.com/api/movie-search.php>



Saved: 5/7/2025 11:59:28 PM

Text Prompt

Weight: 40%

Details:

- Will you be using all the data or just a subset?
- Do you need to convert any data or extract pieces of a property's value?
- Briefly explain the transformation/extraction logic to get the API data to the shape your application intends to use

Your Response:

I will be using just a subset of the data from my API. I will be using the movie title, year, and plot. I didn't need to convert too much data, I did extract specific fields from the JSON response to display onto my pages. The JSON response was decoded and looped through the Search array.



Saved: 5/7/2025 11:59:28 PM

100%

Task #3 (0.50 pts.) - API Duplicates

Text Prompt

Weight: 25%

Objective: *API Duplicates*

Details:

- Consider how duplicate entries are handled
 - What happens if you get the same result from the API for an entity that exists but has been unmodified?
 - What happens if you get the same result from the API for an entity that exists but has been modified?

Your Response:

If I get a duplicate movie from the API that already exists and hasn't been modified, my website prevents adding it to avoid any duplicate entities. If the API returns a version that has been modified (ex. a different title or rating), it is added as a new entity.



Saved: 5/8/2025 12:02:51 AM

100%

Task #4 (0.50 pts.) - API Fetching

Text Prompt

Weight: 25%

Objective: *API Fetching*

Details:

- Consider how your application will trigger this data fetch
 - (i.e., periodic via cron job, poor man's cron job, on app start/awake, etc)
 - (i.e., User or Admin triggered via a specific page/action)
 - (i.e., Lazy loaded during searches for stale/missing content)
 - What happens if you get the same result from the API for an entity that exists but has been modified?

Your Response:

My application will trigger this fetch data when a user searches for a specific movie title manually on the search page. This means it's a form of lazy loading since data is pulled only when needed. If the API returns a modified version of a movie that already exists, it will be added again.



Saved: 5/8/2025 12:06:17 AM

100%

Section #2: (1.5 pts.) Data Creation

100%

Task #1 (0.60 pts.) - Validations

Combo Task:

Weight: 40%

Objective: *Validations*

Image Prompt

Weight: 50%

Details:

- Show the html validations (code)
- Show the javascript validations (code)
- Show the php validations (code)

```
1. <form method="post" action="addMovie.php">
2.   <input type="text" name="title" value="The Godfather" required/>
3.   <input type="text" name="year" value="1972"/>
4.   <input type="text" name="rating" value="9.2"/>
5.   <input type="button" value="Add Movie"/>
6. 
```

showing HTML validation

```
18 <script>
19   function validate(form) {
20     // implementing javascript validation
21     // gses8 - s/1/2s ensures it returns false for previously
22     // existing movies and true if successfully added
23     let isValid = true;
24     const title = form.title.value;
25     const year = form.year.value;
26     const rating = form.rating.value;
27     if (title.length === 0 || title === null) {
28       flash("Empty Title", "danger");
29       isValid = false;
30     }
31     if (year.length === 0 || year === null) {
32       flash("Empty Year", "danger");
33       isValid = false;
34     }
35     return isValid;
36   }
37 </script>
```

showing JS validation

```
39 <?php
40 // RxC58 - 5/1/25 - PHP Code validation
41 if (isset($_POST["title"]) && isset($_POST["year"])) {
42   $title = $_POST["title"]; //$_POST["title"];
43   $year = $_POST["year"]; //$_POST["year"];
44   $rating = $_POST["rating"]; //$_POST["rating"];
45   $t1_ap1 = 0; // 0 for custom form
46   $plot = ""; // empty plot
47
48   $hasError = false;
49   if (empty($title)) {
50     flash("Title must be provided <br>");
51     $hasError = true;
52   }
53
54   if (empty($year)) {
55     flash("Year must be provided <br>");
56     $hasError = true;
57 }
```

showing php validation



Saved: 5/8/2025 12:23:55 AM

Text Prompt

Weight: 50%

Details:

- Briefly explain the html validations used
- Briefly explain the javascript validation logic
- Briefly explain the php validation logic

Your Response:

HTML validation: HTML validation uses built-in HTML attributes like required, type="text", and value="The Godfather". It also uses client-side JavaScript validation via the validate() function.

HTML - I used basic HTML validations like setting type= "number" for numeric inputs. I added min, max, and step to make sure the movie ratings and years stay within a valid range.

Javascript - The validate() function checks if the title and year fields are empty. If they are, it flashes an error & prevents the form from submitting. It also ensures it returns false for previously existing movies and true if successfully added.

PHP - The validation code checks if the required fields like movie title and year are there (not empty). This makes sure that invalid data doesn't get into the database.

100%

Task #2 (0.60 pts.) - Examples

Image Prompt

Weight: 40%

Objective: Examples

Details:

- Show a successful creation
- Demonstrate various validation errors
- Ensure heroku url is visible

Movie Title common Year Published 2012 Rating (out of 10) 8.0

Add Movie

Movie Title is required

adding movie



Movie Title The Shawshank Redemption Year Published 1994 Rating (out of 10) 9.2

Add Movie

Movie successfully added!

successfully added movie



A screenshot of a web-based movie database application. At the top, there's a navigation bar with links like "Home", "About", "Contact", "Logout", "Search", "Add Movie", "Edit Movie", "Delete Movie", "View Movie", "View All Movies", "View All Genres", "View All Directors", "View All Actors", and "View All Countries". Below the navigation is a search bar with fields for "Movie Title", "Year Released", "Rating (out of 10)", and "Genre". A large red button labeled "Add Movie" is prominently displayed.

100%

Task #3 (0.30 pts.) - Links

🔗 Url Prompt

Weight: 20%

Objective: *Links*

Details:

- Include the heroku prod link to this page
- Include pull request link for this feature

URL #1

[https://gs658-it202-008-
prod-96409305ed03.herokuapp.com/project/movie-
add.php](https://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/movie-add.php)



URL

<https://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/movie-add.php>



Saved: 5/8/2025 12:28:32 AM

100%

Section #3: (1.5 pts.) Data List Page (Many Items)

100%

Task #1 (0.60 pts.) - Code

Combo Task:

Weight: 40%

Objective: *Code*

🖼 Image Prompt

Weight: 50%

Weight: 50%

Details:

- Show the code that handles the filter/sort logic
 - Show the code that generates the list/grid output
 - Styling must be applied

code that handles filter/sort logic

1

code that generates list/grid output

10

styling for filter/sort

6

```
54 /* g5658 5/1/25 Applies styling for movie list table */
55 ~table {
56   width: 100%;
57   border-collapse: collapse;
58   margin-top: 20px;
59   background-color: #f0f0f0;
60   color: #fff;
61 }
62
63 table th,
64 ~table td {
65   padding: 12px 15px;
66   border: 1px solid #444;
67   text-align: left;
68 }
```



Saved: 5/8/2025 12:42:40 AM

Text Prompt

Weight: 50%**Details:**

- Briefly explain how the filter/sort logic works and formulates the results
- Briefly explain how the list/grid output is generated
- Briefly note the styling choices

100%

Task #2 (0.60 pts.) - Examples

Image Prompt

Weight: 40%**Objective:** *Examples***Details:**

- Show a few variations of filter/sort applied (including no records)
- Each list item should be a summary
- Each list item should have the following links
 - A link to a single view of the specific entity (i.e., a details page)
 - A link to delete this entity (this may be an admin-only functionality, but it should be present for the respective role)
 - A link to edit this entity (this may be an admin-only functionality, but it should be present for the respective role)
- Ensure heroku url is visible

List of Movies				
Title	Year	Rating	Authors	Watchlist
Avatar: The Way of Water	2022	8	Walter Edith Christopher	Remove
Avengers: Endgame	2019	9	Walter Edith Christopher	Remove
Avengers: Infinity War	2018	9	Walter Edith Christopher	Remove

main page with links to view, edit, and delete pages

rating min 4 max 5 filter applied, limit 10 movies

Title	Year	Rating	Authors
The Godfather	1972	4	Mario Puzo Francis Ford Coppola
The Godfather: Part II	1974	4	Mario Puzo Francis Ford Coppola
The Shawshank Redemption	1994	4	Stephen King Frank Darabont
The Dark Knight	2008	4	Christopher Nolan Deneuve
Seven	1995	4	James Cameron James Cameron
Seven Samurai	1954	4	Akira Kurosawa Kurosawa
Seven	2017	4	David Fincher Edward Norton
Seven	1985	4	William Goldman William Goldman
Seven	1985	4	William Goldman William Goldman

rating min 4 max 5 filter applied, limit 10 movies

Title	Year	Rating	Authors
The Godfather	1972	4	Mario Puzo Francis Ford Coppola
The Godfather: Part II	1974	4	Mario Puzo Francis Ford Coppola
The Shawshank Redemption	1994	4	Stephen King Frank Darabont
The Dark Knight	2008	4	Christopher Nolan Deneuve
Seven	1995	4	James Cameron James Cameron
Seven Samurai	1954	4	Akira Kurosawa Kurosawa
Seven	2017	4	David Fincher Edward Norton
Seven	1985	4	William Goldman William Goldman
Seven	1985	4	William Goldman William Goldman
Seven	1985	4	William Goldman William Goldman

Task #3 (0.30 pts.) - Links

Url Prompt

Weight: 20%

Objective: *Links*

Details:

- Include the heroku prod link to this page
- Include pull request link for this feature

URL #1

[https://gs658-it202-008-
prod-96409305ed03.herokuapp.com/project/movie-
list.php](https://gs658-it202-008-prod-9640930-project/movie-list.php)



URL

[https://gs658-it202-008-prod-9640930-
project/movie-list.php](https://gs658-it202-008-prod-9640930-project/movie-list.php)



Saved: 5/8/2025 12:42:59 AM

100%

Section #4: (1.5 pts.) View Details Page (Single Item)

Task #1 (0.60 pts.) - Code

Combo Task:

Weight: 40%

Objective: Code

Image Prompt

Weight: 50%

Details:

- Show the code that handles loading of the record (and how the id retrieved for usage)
 - Include the code that handles invalid/missing ids
- Show the code that generates the entity output
- Styling must be applied

```

4 // Check if the movie_id is provided
5 $movie_id = isset($_GET['id']) & (int)$_GET['id'] != null;
6 if (!($movie_id)) {
7     flash("Movie ID is required", "danger");
8     redirect("movie-list.php");
9     exit;
10 }
11
12 $db = GetDB();
13 $query = "SELECT * FROM Movies WHERE id = :id";
14 $params = [":id" => $movie_id];
15
16 // Prepare and execute the query
17 $stmt = $db->prepare($query);
18 $stmt->bindValue(":id", $movie_id, PDO::PARAM_INT);
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
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
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
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
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
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
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
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
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
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
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
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
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
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
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
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
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
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
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
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
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
1589
1589
1590
1591
1592
1593
1594
1595
1596
1596
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
1689
1689
1690
1691
1692
1693
1694
1695
1696
1696
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
1789
1789
1790
1791
1792
1793
1794
1795
1796
1796
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
1889
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
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
1989
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
2089
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
21
```

code that generates movie outputs

```
34     body {  
35         text-align: center;  
36         color: #white;  
37         background-color: #333333;  
38         font-family: 'segoe ui', tahoma, geneva, verdana, sans-serif;  
39     }  
40
```

css styling

```
70     > p {  
71         margin-top: 10px;  
72         color: #ccc;  
73     }  
74
```

more css styling



Saved: 5/8/2025 1:02:53 AM

100%

Task #2 (0.60 pts.) - Examples

Image Prompt

Weight: 40%

Objective: *Examples*

Details:

- Show a few examples of different entities
- Each entity item should have more details than the summary view (if possible)
- Each entity should have the following links
 - A link to edit the single entity (this may be an admin-only thing, but it should be present for the respective role)
 - A link to delete the single entity (this may be an admin-only thing, but it should be present for the respective role)

The screenshot shows a web browser window with a dark theme. The main content area displays a "Movie Details" page for "The 300: A Space Odyssey". The page includes the title, year (2008), rating (PG), and a link to "View | Edit | Delete". Below the main content is a large, empty black rectangular area.

entity #1 view page



The screenshot shows a web browser window with a dark theme. The main content area displays a "Movie Details" page for "2001: A Space Odyssey". The page includes the title, year (1968), rating (A), and a link to "View | Edit | Delete". Below the main content is a large, empty black rectangular area.

entity 1 view/edit/delete page



The screenshot shows a web browser window with a dark theme. The main content area displays a "Movie Details" page for "The Jupiter Train". The page includes the title, year (2003), rating (R), and a link to "View | Edit | Delete". Below the main content is a large, empty black rectangular area.

entity #2 view page



Task #3 (0.30 pts.) - Links

⊕ Url Prompt

Weight: 20%

Objective: *Links*

Details:

- Include the heroku prod link to this page

- Include the Heroku prod link to this page
 - Include pull request link for this feature

URL #1

<https://gs658-it202-008->



6

<https://gs658-it202-008-prod-96409301>



Saved: 5/8/2025 1:04:12 AM

100%

Section #5: (1.5 pts.) Edit Page

100%

Task #1 (0.60 pts.) - Code

Combo Task:

Weight: 40%

Objective: Code



Weight: 50%

Details:

- Show the code that handles loading of the record (and how the id retrieved for usage)
 - Include the code that handles invalid/missing ids
 - Show the code that generates the form with the correct data types and populates it
 - Show the html, javascript, and php validation (should be similar to create)
 - Styling must be applied

```

4 // Check if the ID parameter is provided
5 if (!isset($_GET['id'])) {
6     flash("Movie ID is required.", "danger");
7     redirect("movies_list.php"); // Redirect to the movie list page if ID is not
8     die();
9 }
10
11 $movie_id = $_GET['id'];
12
13 $db = getDB();
14
15 // Fetch the current movie data from the database
16 $query = "SELECT * FROM Movies WHERE id = :id LIMIT 1";
17 $stmt = $db->prepare($query);
18 $stmt->bindParam(':id', $movie_id, PDO::PARAM_INT);
19 $stmt->execute();
20 $movie = $stmt->fetch(PDO::FETCH_ASSOC);
21
22 if (!$movie) {
23     flash("Movie not found.", "danger");
24     redirect("movies_list.php"); // Redirect if movie is not found
25     die();
26 }

```

code that handles loading of the record/how the

code that handles loading of the record/now the

```
23 // checks invalid / missing ID
24 if (!$movie) {
25     Flash("Movie not found.", "danger");
26     redirect("movies_list.php"); // Redirect if movie is not found
27     die();
28 }
```

checks for invalid/missing ID

```
11 <?php
12     $movie = $db->fetchObject('SELECT * FROM movies WHERE id = ?',
13         array($id));
14
15     if ($movie) {
16         $title = $movie->title;
17         $year = $movie->year;
18         $rating = $movie->rating;
19     }
20
21     $form = new Form('updateMovieForm');
22     $form->addText('title', $title);
23     $form->addText('year', $year);
24     $form->addText('rating', $rating);
25
26     echo $form->render();
27
28     $script = "function updateMovie() {
29         var title = document.getElementById('title').value;
30         var year = document.getElementById('year').value;
31         var rating = document.getElementById('rating').value;
32
33         if (title == '' || year == '' || rating == '') {
34             alert('Title, Year, and Rating are required fields.');
35         } else {
36             var movie_id = document.getElementById('movie_id').value;
37
38             var update_query = 'UPDATE Movies SET title = ?title, year = ?year, rating = ?rating WHERE id = ?';
39             var update_stmt = $db->prepare($update_query);
40             $update_stmt->bindValue(':title', $title, PDO::PARAM_STR);
41             $update_stmt->bindValue(':year', $year, PDO::PARAM_INT);
42             $update_stmt->bindValue(':rating', $rating, PDO::PARAM_STR);
43             $update_stmt->bindValue(':id', $movie_id, PDO::PARAM_INT);
44
45             $update_stmt->execute();
46         }
47     }
48
49     <!-->
```

code that generates the form with the correct data types & populates it

```
11 <?php
12     $movie = $db->fetchObject('SELECT * FROM movies WHERE id = ?',
13         array($id));
14
15     if ($movie) {
16         $title = $movie->title;
17         $year = $movie->year;
18         $rating = $movie->rating;
19     }
20
21     $form = new Form('updateMovieForm');
22     $form->addText('title', $title);
23     $form->addText('year', $year);
24     $form->addText('rating', $rating);
25
26     echo $form->render();
27
28     $script = "function updateMovie() {
29         var title = document.getElementById('title').value;
30         var year = document.getElementById('year').value;
31         var rating = document.getElementById('rating').value;
32
33         if (title == '' || year == '' || rating == '') {
34             alert('Title, Year, and Rating are required fields.');
35         } else {
36             var movie_id = document.getElementById('movie_id').value;
37
38             var update_query = 'UPDATE Movies SET title = ?title, year = ?year, rating = ?rating WHERE id = ?';
39             var update_stmt = $db->prepare($update_query);
40             $update_stmt->bindValue(':title', $title, PDO::PARAM_STR);
41             $update_stmt->bindValue(':year', $year, PDO::PARAM_INT);
42             $update_stmt->bindValue(':rating', $rating, PDO::PARAM_STR);
43             $update_stmt->bindValue(':id', $movie_id, PDO::PARAM_INT);
44
45             $update_stmt->execute();
46         }
47     }
48
49     <!-->
```

HTML validation

```
46 // Handling form submission for updating the movie details
47 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
48     $title = trim($_POST['title']);
49     $year = intval($_POST['year']);
50     $rating = floatval($_POST['rating']);
51
52     // Validate inputs
53     if (empty($title) || empty($year) || empty($rating)) {
54         Flash("Title, Year, and Rating are required fields.", "danger");
55     } else {
56         // Update the movie in the database
57         $update_query = "UPDATE Movies SET title = :title, year = :year, rating = :rating WHERE id = :id";
58         $update_stmt = $db->prepare($update_query);
59         $update_stmt->bindValue(':title', $title, PDO::PARAM_STR);
60         $update_stmt->bindValue(':year', $year, PDO::PARAM_INT);
61         $update_stmt->bindValue(':rating', $rating, PDO::PARAM_STR);
62         $update_stmt->bindValue(':id', $movie_id, PDO::PARAM_INT);
63
64         $update_stmt->execute();
65     }
66 }
```

PHP validation

```
34. </body> {
35.     text-align: center;
36.     color: white;
37.     background-color: #000000;
38.     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
39. }
40.
```

css styling



100%

Task #2 (0.60 pts.) - Examples

Image Prompt

Weight: 40%

Objective: *Examples*

Details:

- Show a before and after of updating an entity (clearly caption)
- Successful update should have an appropriate message shown
- The updated data should be shown in the form
- Any errors should have user-friendly messages shown
- Show some examples of validation errors (html, javascript, php)



antman view page before



antman view page after with successful message



100%

Task #3 (0.30 pts.) - Links

Url Prompt

Weight: 20%

Objective: *Links*

Details:

- Include the heroku prod link to this page
- Include pull request link for this feature

URL #1

<https://gs658-it202-008->

prod-96409305ed03.herokuapp.com/project/movie-edit.php



URL

<https://gs658-it202-008-prod-9640930!>



Saved: 5/8/2025 1:04:26 AM

100%

Section #6: (1.5 pts.) Delete Logic

100%

Task #1 (0.60 pts.) - Code

Combo Task:

Weight: 40%

Objective: *Code*

Image Prompt

Weight: 50%

Details:

- Show the code that handles deletion of the record (and how the id retrieved for usage)
 - Include the code that handles invalid/missing ids
- Handle any necessary roles/permissions (i.e., it's not likely that anyone can delete any entity they choose)
- Show the redirect logic that goes back to the previous page (where the route came from), if it was a list page, it should maintain the filter/sort criteria

```
11 $movie_id = $_GET['id']; // retrieves the movie id from the query string
12
13 $db = getDB();
14
15 // Fetch the movie data to ensure it exists
16 $query = "SELECT * FROM Movies WHERE id = :id LIMIT 1";
17 $stmt = $db->prepare($query);
18 $stmt->bindParam(':id', $movie_id, PDO::PARAM_INT);
19 $stmt->execute();
20 $movie = $stmt->fetch(PDO::FETCH_ASSOC);
21
22 if (!$movie) {
23     flash("Movie not found.", "danger");
24     redirect("movie-list.php"); // Redirect if movie is not found
25     die();
26 }
27
28 // Delete the movie from the database
29 $query = "DELETE FROM Movies WHERE id = :id";
30 $stmt = $db->prepare($query);
31 $stmt->bindParam(':id', $movie_id, PDO::PARAM_INT);
```

code that handles deletion of the record/how the id is retrieved for usage

```
4 // Checks if the ID parameter is provided
5 if (!isset($_GET['id'])) {
6     flash("Movie ID is required.", "danger");
7     redirect("movie-list.php"); // Redirect to the movie list page if ID is not provided
8     die();
9 }
10
11 $movie_id = $_GET['id']; // retrieves the movie id from the query string
12
13 $db = getDB();
14
15 // Fetch the movie data to ensure it exists
16 $query = "SELECT * FROM Movies WHERE id = :id LIMIT 1";
17 $stmt = $db->prepare($query);
18 $stmt->bindParam(':id', $movie_id, PDO::PARAM_INT);
19 $stmt->execute();
20 $movie = $stmt->fetch(PDO::FETCH_ASSOC);
21
22 if (!$movie) {
23     flash("Movie not found.", "danger");
24     redirect("movie-list.php"); // Redirect if movie is not found
25     die();
26 }
```

code that handles invalid/missing ids

```
try {
    $stmt->execute();
    flash("Movie deleted successfully.", "success");
    redirect("movie-list.php"); // Redirect to the movie list after successful deletion
} catch (Exception $e) {
    flash("Error deleting movie: " . $e->getMessage(), "danger");
    redirect("movie-list.php"); // Redirect back to the movie list on error
}
?>
```

redirect logic that goes back to the previous page

Text Prompt

Weight: 50%

Details:

- Briefly explain how the deletion logic works
- Is it a soft or hard delete?
- Note any permissions/rules applied.
- Briefly explain how the redirect maintains the previous route and any filter/sort data

Your Response:

100%

Task #2 (0.60 pts.) - Examples

Image Prompt

Weight: 40%

Objective: *Examples*

Details:

- Show a successful delete message
- Show a before and after database view of the record being deleted (clearly captioned)
- Successful update should have an appropriate message shown
- The updated data should be shown in the form
- Any errors should have user-friendly messages shown

A screenshot of a web browser displaying a movie list. The page has a header with various navigation links. Below the header is a table titled "List of Movies" with columns for Title, Year, Runtime, Aspects, and Watchlist. One row in the table shows the movie "The Way of the Watch". A red arrow points to a message at the top of the page that says "movie list page view before avatar was deleted".

A screenshot of a web browser displaying a movie list. The page has a header with various navigation links. Below the header is a table titled "List of Movies" with columns for Title, Year, Runtime, Aspects, and Watchlist. The same row as in the previous screenshot is visible, showing "The Way of the Watch". A red arrow points to the same "movie list page view before avatar was deleted" message at the top of the page.

Task #3 (0.30 pts.) - Links

Url Prompt

Weight: 20%

Objective: *Links*

Details:

- Include the heroku prod link to this page (even though it's not a full page, it's more like how logout works)
- Include pull request link for this feature

URL #1

<https://gs658-it202-008->

<prod-96409305ed03.herokuapp.com/project/movie-delete.php>



up

<https://gs658-it202-008-prod-9640930!>



Saved: 5/8/2025 1:04:33 AM

Section #7: (0.5 pts.) Misc

Task #1 (0.17 pts.) - Github Details

Combo Task:

Weight: 33.33%

Objective: *Github Details*

Image Prompt

Weight: 60%

Details:

From the Commits tab of the Pull Request screenshot the commit history

screenshot of commit history

Saved: 5/8/2025 1:49:15 AM

Url Prompt

Weight: 40%

Details:

Include the link to the Pull Request (should end in /pull/#)

URL #1

<https://github.com/goribanu/gs658->

IT2020021



URL

<https://github.com/goribanu/gs658>

Saved: 5/8/2025 1:49:15 AM

100%

Task #2 (0.17 pts.) - WakaTime - Activity

Image Prompt

Weight: 33.33%

Objective: *WakaTime - Activity*

Details:

- Visit the WakaTime.com Dashboard
- Click Projects and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary

Projects - gs658-IT202-008

19 hrs 55 mins over the last 7 Days in gs658-IT202-008 under all branches. 4

total wakatime activity



Files

- 3 hrs 19 mins public_html/project/movie-test.php
- 2 hrs 30 mins public_html/movies/1-movie-add.php
- 7 hrs 1 min public_html/movies/1-movie-edit.php
- 1 hr 2 mins public_html/movies/1-movie-delete.php
- 82 mins public_html/project/admin-wakatime.log
- 80 mins public_html/project/movies-wakatime.log
- 12 mins public_html/project/movies-wakatime.log
- 37 mins public_html/project/book-and-movie.php
- 26 mins public_html/m4/categories/pending.php
- 10 mins inc.php
- 24 mins index.php
- 23 mins public_html/project/login.php
- 19 mins public_html/project/movies-view.php
- 19 mins public_html/project/logout.php
- 564 mins public_html/project/testApi-movies.php
- 522 mins public_html/movies/1-table-movies.edl
- 18 mins public_html/project/testTable.edl
- 18 mins public_html/project/testTable.edl
- 16 mins public_html/movies/1-movie-database.log
- 15 mins public_html/movies/1-movie-database.log
- 14 mins public_html/m4/submit-movie.log
- 14 mins public_html/movies/1-movie-delete.php
- 14 mins public_html/project/logout.php
- 14 mins public_html/project/movies-view.php
- 14 mins public_html/categories.php
- 10 mins inc.functions.php

Branches

- 19 hrs 55 mins dev
- 2 hrs 30 mins M1-Homework
- 7 hrs 1 min M1-Homework
- 1 hr 2 mins M2-Homework
- 82 mins M2-Homework
- 80 mins M2-Homework
- 12 mins M2-Homework
- 37 mins M2-Homework
- 26 mins M2-Homework
- 10 mins M2-Homework

100%

Task #3 (0.17 pts.) - Reflection

Weight: 33.33%

Objective: *Reflection*

Sub-Tasks:

100%

Task #1 (0.33 pts.) - What did you learn?

Text Prompt

Weight: 33.33%

Objective: *What did you learn?*

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

I learned how to delete records from my movie database while checking for their IDs and user permissions. I also got more comfortable with handling redirects back to specific pages, like from the edit page back to the movie view page, and understanding the differences between soft and hard deletes.



Saved: 5/8/2025 1:52:00 AM

100%

Task #2 (0.33 pts.) - What was the easiest part of the assignr

Text Prompt

Weight: 33.33%

Objective: *What was the easiest part of the assignment?*

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

The easiest part of this assignment was the redirects because it was just one line of code and very easy to figure out how to do (clear concept). It was also easy to update nav.php with the certain restrictions (per user role) and I understood all the ideas clearly.



Saved: 5/8/2025 1:53:13 AM

100%

Task #3 (0.33 pts.) - What was the hardest part of the assignr

Text Prompt

Weight: 33.33%

Objective: *What was the hardest part of the assignment?*

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

The hardest part of this assignment was submitting all the screenshots. There were a lot of requirements, a lot of sections, and a lot of validations for each part. It was difficult to manage each one, and if an error occurred, it took a lot of time to debug and figure out what and where exactly the issue was.



Saved: 5/8/2025 1:54:23 AM

