

Movie Ticket Booking Project

Table of Contents

Sr. No.	Content	Page No.
1	Introduction	2
2	Problem Definition	3
3	Modules	4
4	Database	5
5	Implementation	6
6	Results	21
7	Conclusion	25
8	References	26

Introduction

In today's digital era, the demand for seamless and efficient entertainment booking solutions is at an all-time high. The movie ticket booking industry plays a pivotal role in offering customers a hassle-free experience to book and enjoy their favorite films. This project aims to develop a robust movie ticket booking application that allows users to easily select movies, choose showtimes, reserve seats, and confirm their bookings online. Built using Java for the backend and powered by SQL for managing data, this application provides an intuitive and user-friendly interface. From login to ticket printing, the platform supports users with smooth navigation through movie selections, seat reservations, payment processing, and ticket generation—ensuring a streamlined and enjoyable ticket booking experience.

Problem Definition

Many In a world where simplicity and efficiency are paramount, even basic movie ticket booking systems must address common challenges that users face when reserving seats. Despite offering only two movie options, traditional booking methods often fail to meet the demand for a smooth and user-friendly experience.

This project resolves the following key problems:

- Lack of clarity on seat availability, leading to confusion during booking.
- Limited options for selecting and confirming preferred showtimes.
- Inconvenient and time-consuming manual booking processes.
- Basic systems that do not provide streamlined, real-time feedback.

This project aims to provide a simplified yet powerful solution through a movie ticket booking system that ensures users can easily view, select, and book available seats for the two movie options, offering a hassle-free and efficient experience with clear, real-time updates.

Modules

User Authentication:

- **LoginPage.java:** This module handles user login, password input, and authentication. It verifies credentials and grants access to the booking system.

Movie Selection:

- **Select.java:** This module allows users to browse and select a movie. It might display available movies, genres, or any other movie-related details.

Date and Timing:

- **Timings.java:** This module handles the selection of show timings and dates for the selected movie. It displays available schedules based on movie selection.

Seat Selection and Booking:

- **Seats.java:** In this module, users select seats for the chosen movie and timing. The module manages seat availability and adds the booking details to the database.

Payment and Confirmation:

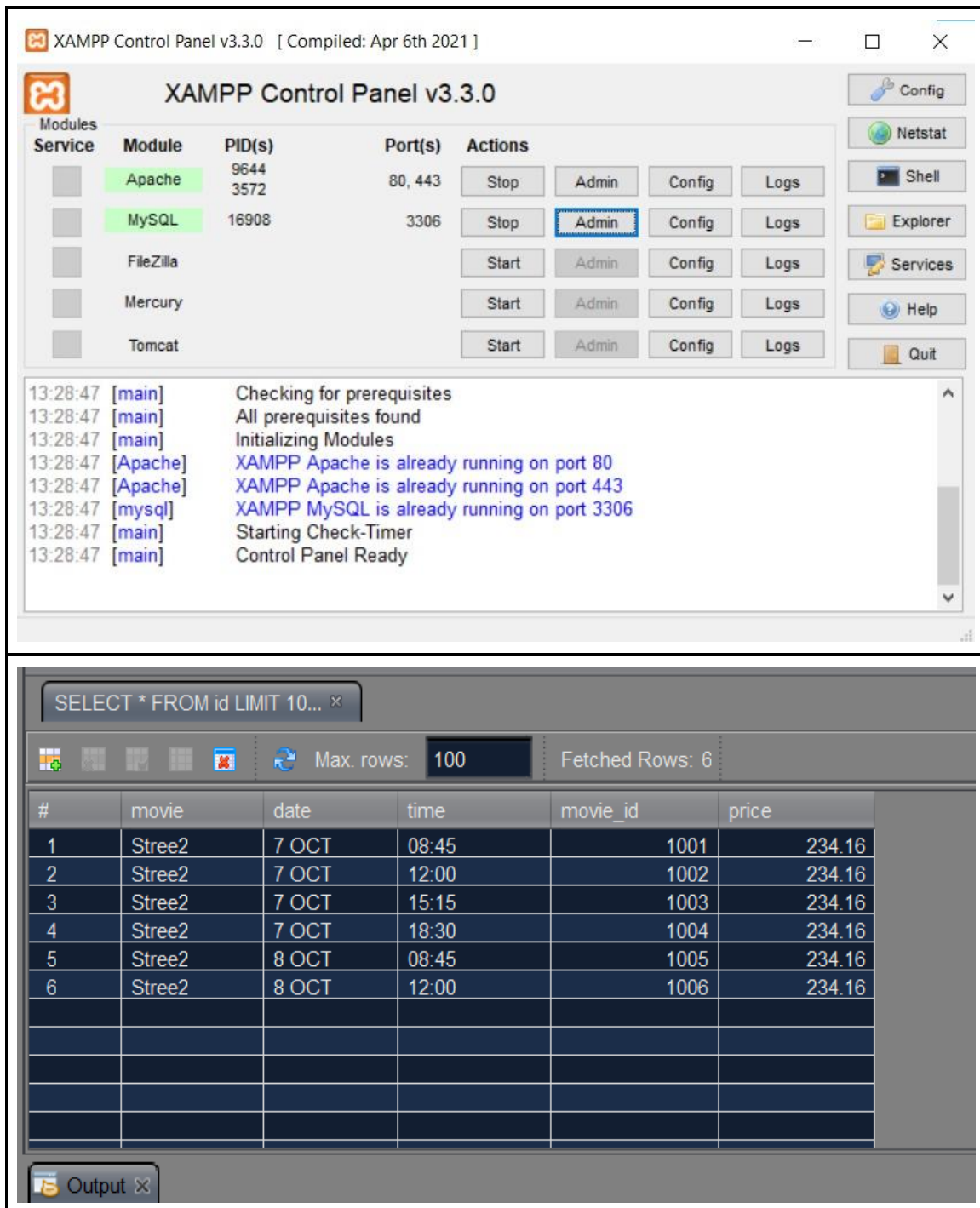
- **PayNConfirm.java:** This module displays a summary of the seat selection, calculates the total price, and confirms the booking.

Ticket Printing:

- **Print.java:** This module generates and prints a ticket that includes movie details, seat numbers, and other booking-related information.

This modular breakdown helps in maintaining the project by clearly separating different functionalities like user management, movie scheduling, seat bookings, and payment.

Database



Implementation

User Authentication (LoginPage.java):

The screenshot shows an IDE window titled 'Start Page.java'. The code is for a Java class named 'LoginPage' that extends 'JFrame'. It includes imports for 'java.sql.*' and 'javax.swing.JFrame'. The class has a constructor 'LoginPage()' and an 'initComponents()' method. A comment indicates that the 'initComponents()' method is generated by the 'Form Editor'. The code also includes a 'private void jTextFieldActionPerformed(java.awt.event.ActionEvent evt)' method and a 'private void jButtonActionPerformed(java.awt.event.ActionEvent evt)' method. The 'jButtonActionPerformed' method contains logic to connect to a MySQL database and execute an insert statement.

```
1 package p1;
2 import java.sql.*;
3 /**
4  *
5  * @author ayush
6  */
7 public class LoginPage extends javax.swing.JFrame {
8
9     /**
10      * Creates new form LoginPage
11      */
12     public LoginPage() {
13         initComponents();
14     }
15
16     /**
17      * This method is called from within the constructor to initialize the form.
18      * WARNING: Do NOT modify this code. The content of this method is always
19      * regenerated by the Form Editor.
20      */
21     @SuppressWarnings("unchecked")
22     // Generated Code
23
24     private void jTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
25         // TODO add your handling code here:
26     }
27
28     private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
29         // TODO add your handling code here:
30         String Username, Password;
31         Username=PasswordField1.getText();
32         Password=PasswordField1.getPassword();
33         try {
34             Class.forName("com.mysql.cj.jdbc.Driver");
35             Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+"user=root");
36             PreparedStatement pst;
37             pst=conn.prepareStatement("insert into username (Username, Password) values (?,?)");
38             Username=jTextField1.getText();
39             Password=PasswordField1.getPassword();
```

```
Start Page x LoginPage.java x
Source Design History
88 Username=jTextField1.getText();
89 Password=jPasswordField1.getText();
90 pst.setString(1, string:Username);
91 pst.setString(1: 2, string:Password);
92 pst.executeUpdate();
93
94 }
95 catch(Exception e){
96 {
97 System.out.println(e);
98 }
99
100 Select obj = new Select(Username);
101 obj.setVisible(a: true);
102 this.setVisible(b: false);
103 }
104
105 private void jPasswordField1ActionPerformed(java.awt.event.ActionEvent evt) {
106 .....// TODO: add your handling code here:
107 }
108
109 /**
110 .....@param args the command line arguments
111 .....*/
112 public static void main(String args[]) {
113 .....// Set the Nimbus look and feel.
114 Look and feel setting code (optional)
115
116 .....// Create and display the form
117 java.awt.EventQueue.invokeLater(new Runnable() {
118 public void run() {
119 new LoginPage().setVisible(a: true);
120 }
121 });
122
123 }
124
125 /** Variables declaration -- do not modify */
126 private javax.swing.JButton jButton1;
```

```

106 .....// TODO: add your handling code here:
107 .....
108 .....
109 ...../**
110 ..... * @param args the command line arguments
111 ..... */
112 .....public static void main(String args[]) {
113 .....    .....// Set the Nimbus look and feel
114 .....    .....Look and feel setting code (optional)
115 .....
116 .....    .....// Create and display the form
117 .....    .....java.awt.EventQueue.invokeLater(new Runnable() {
118 .....        .....public void run() {
119 .....            .....new LoginPage().setVisible(true);
120 .....        }
121 .....    });
122 .....}
123 .....
124 .....// Variables declaration - do not modify
125 .....private javax.swing.JButton jButton1;
126 .....private javax.swing.JLabel jLabel1;
127 .....private javax.swing.JLabel jLabel2;
128 .....private javax.swing.JLabel jLabel3;
129 .....private javax.swing.JPasswordField jPasswordField1;
130 .....private javax.swing.JTextField jTextField1;
131 .....// End of variables declaration
132 .....
133 .....

```

Movie Selection (Select.java):

```

1 package pl;
2 import java.sql.*;
3 .....
4 ...../**
5 ..... *
6 ..... * @author ayush
7 ..... */
8 public class Select extends javax.swing.JFrame {
9 .....
10 .....    ...../**
11 .....     ..... * Creates new form Select
12 .....     ..... * @param username the username to log in with
13 .....     ..... */
14 .....    public Select(String username) {
15 .....        .....this.Username=username;
16 .....        .....initComponents();
17 .....    }
18 .....
19 .....    ...../**
20 .....     ..... * This method is called from within the constructor to initialize the form
21 .....     ..... * WARNING: Do NOT modify this code. The content of this method is always
22 .....     ..... * regenerated by the Form Editor.
23 .....     ..... */
24 .....    @SuppressWarnings("unchecked")
25 .....    .....Generated Code
26 .....
27 .....    .....private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
28 .....        .....String movieName="Stree2";
29 .....        .....Timings obj=new Timings(movieName);
30 .....        .....obj.setVisible(true);
31 .....        .....this.setVisible(false);
32 .....        .....try {
33 .....            .....Class.forName("com.mysql.cj.jdbc.Driver");
34 .....            .....Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+"user=root");
35 .....            .....PreparedStatement pst;
36 .....            .....String q="insert into ticket (username,movie,id) values(?,?,?) "+"on duplicate key update username=values(username),movie=values(movie),id=valu
37 .....            .....pst=conn.prepareStatement(string,q);
38 .....            .....int i=1;

```



```

Source Design History
96 int i=1;
97 pst.setString(i: 1, 'string:Username');
98 pst.setString(i: 2, 'string:movieName');
99 pst.setInt(i: 3, 'i: i');
100 pst.executeUpdate();
101 }
102 }
103 catch(Exception e){
104 {
105 System.out.println(e: e);
106 }
107 // TODO: add your handling code here:
108 }
109 }
110 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
111 String movieName="Deadpool&Wolverine";
112 Timings obj = new Timings(movieName);
113 obj.setVisible(b: true);
114 this.setVisible(b: false);
115 try{
116 Class.forName(classname:"com.mysql.cj.jdbc.Driver");
117 Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+"user=root");
118 PreparedStatement pst;
119 String q="insert into ticket (username, movie, id) values(?, ?, ?) " + "on duplicate key update username=values(username), movie=values(movie), id=valu
120 pst=conn.prepareStatement(string:q);
121 int i=1;
122 pst.setString(i: 1, 'string:Username');
123 pst.setString(i: 2, 'string:movieName');
124 pst.setInt(i: 3, 'i: i');
125 pst.executeUpdate();
126 }
127 }
128 catch(Exception e){
129 {
130 System.out.println(e: e);
131 }
132 // TODO: add your handling code here:
133 }
134 }

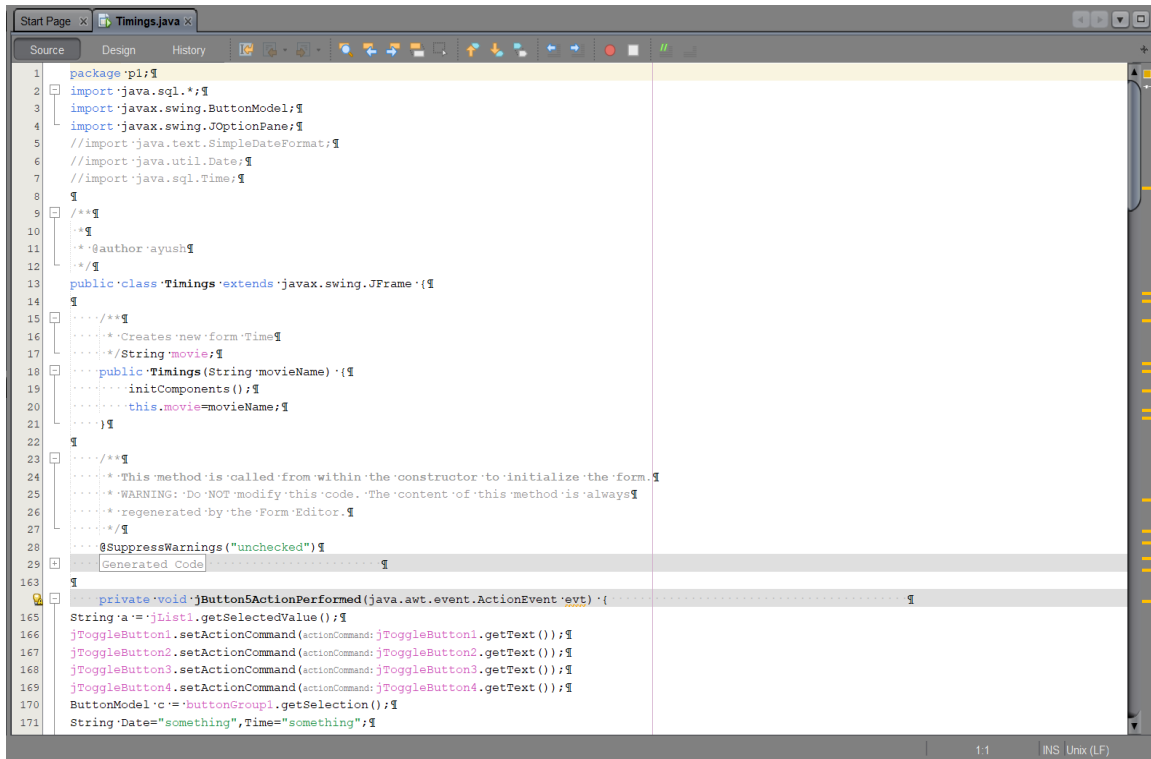
```

```

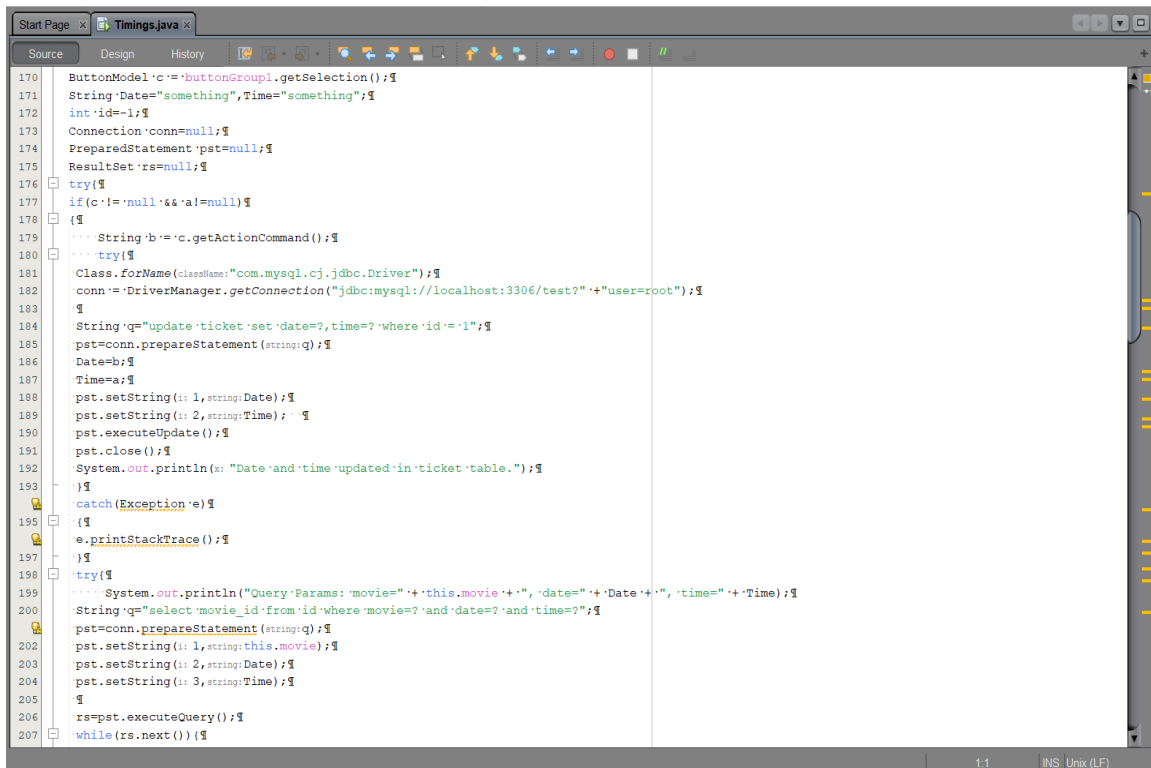
Source Design History
129 {
130 System.out.println(e: e);
131 }
132 // TODO: add your handling code here:
133 }
134 }
135 /**
136 * @param args the command line arguments
137 */
138 public static void main(String args[]) {
139 // Set the Nimbus look and feel
140 Look and feel setting code (optional)
141 }
142 // Create and display the form
143 // java.awt.EventQueue.invokeLater(new Runnable() {
144 // public void run() {
145 // new Select().setVisible(true);
146 // }
147 // });
148 }
149 // Variables declaration - do not modify
150 private javax.swing.JButton jButton1;
151 private javax.swing.JButton jButton2;
152 private javax.swing.JLabel jLabel1;
153 // End of variables declaration
154 }
155 private void setMaximumSize(int i, int i0) {
156 // throw new UnsupportedOperationException(message: "Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/C
157 }
158 }
159 private void setPreferredSize(int i, int i0) {
160 // throw new UnsupportedOperationException(message: "Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/C
161 }
162 }
163 }
164 }

```

Date and Timing (Timings.java):



```
1 package pl;
2 import java.sql.*;
3 import javax.swing.ButtonModel;
4 import javax.swing.JOptionPane;
5 //import java.text.SimpleDateFormat;
6 //import java.util.Date;
7 //import java.sql.Time;
8
9 /**
10  *
11  * @author ayush
12  */
13 public class Timings extends javax.swing.JFrame {
14
15     /**
16      * Creates new form Time
17      * @param movie
18      */
19     public Timings(String movieName) {
20         initComponents();
21         this.movie = movieName;
22     }
23
24     /**
25      * This method is called from within the constructor to initialize the form.
26      * WARNING: Do NOT modify this code. The content of this method is always
27      * regenerated by the Form Editor.
28      */
29     @SuppressWarnings("unchecked")
30     // Generated Code
31
32     private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
33         String a = jList1.getSelectedValue();
34         jButton1.setActionCommand(jButton1.getText());
35         jButton2.setActionCommand(jButton2.getText());
36         jButton3.setActionCommand(jButton3.getText());
37         jButton4.setActionCommand(jButton4.getText());
38         ButtonModel c = buttonGroup1.getSelection();
39         String Date = "something", Time = "something";
40     }
```



```
170 ButtonModel c = buttonGroup1.getSelection();
171 String Date = "something", Time = "something";
172 int id = 1;
173 Connection conn = null;
174 PreparedStatement pst = null;
175 ResultSet rs = null;
176 try {
177     if (c != null && a != null) {
178         String b = c.getActionCommand();
179         try {
180             Class.forName("com.mysql.cj.jdbc.Driver");
181             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?user=root");
182             String q = "update ticket set date=?, time=? where id=?";
183             pst = conn.prepareStatement(q);
184             Date = b;
185             Time = a;
186             pst.setString(1, string: Date);
187             pst.setString(2, string: Time);
188             pst.executeUpdate();
189             pst.close();
190             System.out.println("Date and time updated in ticket table.");
191         } catch (Exception e) {
192             e.printStackTrace();
193         }
194     }
195     try {
196         System.out.println("Query Params: movie=" + this.movie + ", date=" + Date + ", time=" + Time);
197         String q = "select movie_id from id where movie=? and date=? and time=?";
198         pst = conn.prepareStatement(q);
199         pst.setString(1, string: this.movie);
200         pst.setString(2, string: Date);
201         pst.setString(3, string: Time);
202         rs = pst.executeQuery();
203         while (rs.next()) {
204             // ...
205         }
206     }
207 }
```

```

Start Page x Timings.java x
Source Design History
206 rs=pst.executeQuery();
207 while(rs.next()){
208     id=rs.getInt(string:"movie_id");
209 }
210 System.out.println("Fetched 'movie_id:' +id);
211 rs.close();
212 pst.close();
213 }
214 catch(Exception e){
215     {
216         e.printStackTrace();
217     }
218 }
219 try{
220     String q="update 'ticket' set 'movie_id=?' where 'id=1'";
221     pst=conn.prepareStatement(string:q);
222     pst.setInt(1,1; id);
223     pst.executeUpdate();
224     pst.close();
225     System.out.println(x: "Movie 'ID' updated 'in' 'tickets' 'table.'");
226 }
227 catch(Exception e){
228     {
229         e.printStackTrace();
230     }
231 }
232 Seats 'obj' := new Seats (Date,Time);
233 obj.setVisible(b: true);
234 this.setVisible(w: false);
235 }
236 else-if(a!=null){
237     {
238         JOptionPane.showMessageDialog(parentComponent: null, 'message: "Please 'select' a 'date'"');
239     }
240     else-if(c!=null){
241         {
242             JOptionPane.showMessageDialog(parentComponent: null, 'message: "Please 'select' a 'time' 'slot'"');
243         }
244     }

```

```

Start Page x Timings.java x
Source Design History
242     JOptionPane.showMessageDialog(parentComponent: null, 'message: "Please 'select' a 'time' 'slot'"');
243 }
244 else{
245     JOptionPane.showMessageDialog(parentComponent: null, 'message: "Please 'select' 'date' and 'time'"'); // Handle 'case' 'when' 'no' 'button' 'is' 'selected'
246 }
247 finally{
248     try{
249         if(rs!=null) rs.close();
250         if(pst!=null) pst.close();
251         if(conn!=null) conn.close();
252     } catch (Exception e) {
253         e.printStackTrace();
254     }
255 }
256 buttonGroup1.clearSelection();
257 }
258 // TODO add your handling code here:
259 }
260
261 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
262     // TODO add your handling code here:
263 }
264
265 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
266     // TODO add your handling code here:
267 }
268
269 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
270     // TODO add your handling code here:
271 }
272
273 private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
274     // TODO add your handling code here:
275 }
276
277 /**
278  * @param args the command-line arguments
279  */

```

```

272  private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
273      // TODO add your handling code here:
274  }
275
276
277  /**
278   * @param args the command line arguments
279   */
280  public static void main(String args[]) {
281      // Set the Nimbus look and feel
282      // Look and feel setting code (optional)
283
284      // Create and display the form
285      // java.awt.EventQueue.invokeLater(new Runnable() {
286      //     public void run() {
287      //         new Timings().setVisible(true);
288      //     }
289  });
290
291  // Variables declaration - do not modify
292  private javax.swing.ButtonGroup buttonGroup1;
293  private javax.swing.JButton jButton5;
294  private javax.swing.JLabel jLabel1;
295  private javax.swing.JLabel jLabel2;
296  private javax.swing.JList<String> jList1;
297  private javax.swing.JScrollPane jScrollPane1;
298  private javax.swing.JToggleButton jButton1;
299  private javax.swing.JToggleButton jButton2;
300  private javax.swing.JToggleButton jButton3;
301  private javax.swing.JToggleButton jButton4;
302  // End of variables declaration
303  }
304
305

```

Seat Selection and Booking (Seats.java):

```

1  package pl;
2
3  import java.awt.*;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.sql.*;
7  import java.util.*;
8  import javax.swing.JToggleButton;
9
10 /**
11  *
12  * @author ayush
13  */
14 public class Seats extends JFrame {
15
16     /**
17      * Creates new form seats
18      */
19     public Seats(String Date, String Time) {
20         this.Date = Date;
21         this.Time = Time;
22         initComponents();
23         disableBookedSeats();
24     }
25
26     /**
27      * This method is called from within the constructor to initialize the form.
28      * WARNING: Do NOT modify this code. The content of this method is always
29      * regenerated by the Form Editor.
30      */
31     @SuppressWarnings("unchecked")
32     // Generated Code
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
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

Start Page x Seats.java x
Source Design History
272 toggleButtons.add(e: button2);
273 toggleButtons.add(e: button3);
274 toggleButtons.add(e: button4);
275 toggleButtons.add(e: button5);
276 toggleButtons.add(e: button6);
277 toggleButtons.add(e: button7);
278 toggleButtons.add(e: button8);
279 toggleButtons.add(e: button9);
280 toggleButtons.add(e: button10);
281 toggleButtons.add(e: button11);
282 toggleButtons.add(e: button12);
283 toggleButtons.add(e: button13);
284 toggleButtons.add(e: button14);
285 toggleButtons.add(e: button15);
286 toggleButtons.add(e: button16);
287 toggleButtons.add(e: button17);
288 toggleButtons.add(e: button18);
289 toggleButtons.add(e: button19);
290 toggleButtons.add(e: button20);
291
292 for(JToggleButton button: toggleButtons)
293 {
294     button.setBackground(bg: Color.GRAY);
295     button.setEnabled(b: true);
296 }
297
298 try{
299     Class.forName(className:"com.mysql.cj.jdbc.Driver");
300     Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+"user=root");
301     String q="select seats from tickets where id=1 and date=? and time=?";
302     PreparedStatement pst=conn.prepareStatement(q);
303     pst.setString(i: 1, "string:Date");
304     pst.setString(i: 2, "string:Time");
305     ResultSet rs=pst.executeQuery();
306     while(rs.next())
307     {
308         String bookedSeats=rs.getString("seats");
309         String[] bookedSeatsArray = bookedSeats.split(",");
310     }
311 }

```

309.1 INS Unix (LF)

```

Start Page x Seats.java x
Source Design History
309
310 for(JToggleButton button: toggleButtons)
311 {
312     for(String bookedSeat: bookedSeatsArray)
313     {
314         if(button.getText().equals(bookedSeat))
315         {
316             button.setEnabled(b: false);
317             button.setBackground(bg: Color.red);
318         }
319     }
320 }
321
322 catch(Exception e)
323 {
324     System.out.println(e);
325 }
326
327
328
329 private void saveSelectedSeats() {
330     int nofseat=0;
331     StringBuilder s = new StringBuilder();
332     java.util.List<JToggleButton> toggleButtons = new java.util.ArrayList<>();
333     toggleButtons.add(e: button1);
334     toggleButtons.add(e: button2);
335     toggleButtons.add(e: button3);
336     toggleButtons.add(e: button4);
337     toggleButtons.add(e: button5);
338     toggleButtons.add(e: button6);
339     toggleButtons.add(e: button7);
340     toggleButtons.add(e: button8);
341     toggleButtons.add(e: button9);
342     toggleButtons.add(e: button10);
343     toggleButtons.add(e: button11);
344     toggleButtons.add(e: button12);
345     toggleButtons.add(e: button13);
346

```

309.3 INS Unix (LF)

```

345 toggleButtons.add(e: button12);
346 toggleButtons.add(e: button13);
347 toggleButtons.add(e: button14);
348 toggleButtons.add(e: button15);
349 toggleButtons.add(e: button16);
350 toggleButtons.add(e: button17);
351 toggleButtons.add(e: button18);
352 toggleButtons.add(e: button19);
353 toggleButtons.add(e: button20);
354 for(JToggleButton button: toggleButtons){
355     {
356         if(button.isSelected()){
357             {
358                 String seatNumber = button.getText();
359                 s.append(str: seatNumber).append(str: ",");
360                 nofseat++;
361                 button.setBackground(bg: Color.GRAY); // Reset to gray after saving
362                 button.setSelected(s: false);
363                 System.out.println(s: s);
364             }
365         }
366         // Remove last comma if there are any selected seats
367         if(s.length() > 0){
368             s.setLength(s.length()-1);
369         }
370     }
371     try{
372         Class.forName(className: "com.mysql.cj.jdbc.Driver");
373         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+"user=root");
374         String q = "update ticket set seats=? ,nofseat=? where id=1";
375         PreparedStatement pst = conn.prepareStatement(string: q);
376         System.out.println(s.toString()+" "+ nofseat);
377         pst.setString(1, string: s.toString());
378         pst.setInt(2, 11: nofseat);
379         pst.executeUpdate();
380     }
381     catch(Exception e){
382         //

```

```

382     catch(Exception e){
383         System.out.println(s: e);
384     }
385 }
386
387 private void button1ActionPerformed(java.awt.event.ActionEvent evt) {
388     if(button1.getBackground() != Color.GREEN){
389         button1.setBackground(bg: Color.GREEN);
390     }
391     else{
392         button1.setBackground(bg: Color.GRAY);
393         // TODO add your handling code here:
394     }
395 }
396
397 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
398     //jButton2.addActionListener(new ActionListener() {
399     //public void actionPerformed(ActionEvent e) {
400     //saveSelectedSeats();
401     //}
402     //});
403     saveSelectedSeats();
404     PayNConfirm obj = new PayNConfirm(Date, Time);
405     obj.setVisible(s: true);
406     this.setVisible(s: false);
407     // TODO add your handling code here:
408 }
409
410 private void button2ActionPerformed(java.awt.event.ActionEvent evt) {
411     if(button2.getBackground() != Color.GREEN){
412         button2.setBackground(bg: Color.GREEN);
413     }
414     else{
415         button2.setBackground(bg: Color.GRAY);
416         // TODO add your handling code here:
417     }
418 }
419
420 private void button3ActionPerformed(java.awt.event.ActionEvent evt) {
421     if(button3.getBackground() != Color.GREEN){
422         button3.setBackground(bg: Color.GREEN);

```

```

Start Page x Seats.java x
Source Design History
544 private void button19ActionPerformed(java.awt.event.ActionEvent evt) {
545     if(button19.getBackground() != Color.green) {
546         button19.setBackground(bg: Color.green);
547     } else {
548         button19.setBackground(bg: Color.GRAY);
549         // TODO: add your handling code here:
550     }
551 }
552 private void button20ActionPerformed(java.awt.event.ActionEvent evt) {
553     if(button20.getBackground() != Color.green) {
554         button20.setBackground(bg: Color.green);
555     } else {
556         button20.setBackground(bg: Color.GRAY);
557         // TODO: add your handling code here:
558     }
559 }
560 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
561     // TODO: add your handling code here:
562 }
563
564 /**
565  * @param args the command line arguments
566  */
567 public static void main(String args[]) {
568     // Set the Nimbus look and feel
569     // Look and feel setting code (optional)
570     // </editor-fold>
571
572     // Create and display the form
573     java.awt.EventQueue.invokeLater(new Runnable() {
574         public void run() {
575             new Seats().setVisible(true);
576         }
577     });
578 }
579
580 // Variables declaration - do not modify

```

```

Start Page x Seats.java x
Source Design History
590 // </editor-fold>
591
592 // Create and display the form
593 java.awt.EventQueue.invokeLater(new Runnable() {
594     public void run() {
595         new Seats().setVisible(true);
596     }
597 });
598
599
600 // Variables declaration - do not modify
601 private javax.swing.JToggleButton button1;
602 private javax.swing.JToggleButton button10;
603 private javax.swing.JToggleButton button11;
604 private javax.swing.JToggleButton button12;
605 private javax.swing.JToggleButton button13;
606 private javax.swing.JToggleButton button14;
607 private javax.swing.JToggleButton button15;
608 private javax.swing.JToggleButton button16;
609 private javax.swing.JToggleButton button17;
610 private javax.swing.JToggleButton button18;
611 private javax.swing.JToggleButton button19;
612 private javax.swing.JToggleButton button2;
613 private javax.swing.JToggleButton button20;
614 private javax.swing.JToggleButton button3;
615 private javax.swing.JToggleButton button4;
616 private javax.swing.JToggleButton button5;
617 private javax.swing.JToggleButton button6;
618 private javax.swing.JToggleButton button7;
619 private javax.swing.JToggleButton button8;
620 private javax.swing.JToggleButton button9;
621 private javax.swing.JButton jButton1;
622 private javax.swing.JButton jButton2;
623 // End of variables declaration
624 }
625

```

Payment and Confirmation (PayNConfirm.java):

```
Start Page x PayNConfirm.java x
Source Design History
1 package pl;
2 import java.awt.*;
3 import java.sql.*;
4 import javax.swing.*;
5 /**
6  *
7  * @author ayush
8  */
9 public class PayNConfirm extends javax.swing.JFrame {
10
11     /**
12      * Creates new form PayNConfirm
13      */
14     private String Date;
15     private String Time;
16     private int numSeat=0;
17     private float price=0;
18     private String movie=null;
19     public PayNConfirm(String Date,String Time) {
20         this.Date=Date;
21         this.Time=Time;
22         initComponents();
23         jButton2.setEnabled(false);
24         jButton2.setVisible(false);
25         findSeatsNPrice();
26     }
27
28     /**
29      * This method is called from within the constructor to initialize the form.
30      * WARNING: Do NOT modify this code. The content of this method is always
31      * regenerated by the Form Editor.
32      */
33     @SuppressWarnings("unchecked")
34     // Generated Code
35
36     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
37         Print obj = new Print(Date,Time,movie);
38         obj.setVisible(true);
39         this.setVisible(false);
40         // TODO add your handling code here:
41     }
42 }
```

```
Start Page x PayNConfirm.java x
Source Design History
109 // TODO add your handling code here:
110 }
111
112 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
113     findSeatsNPrice();
114     JOptionPane.showMessageDialog(parentComponent, null, "Payment Successful .....");
115     jButton2.setEnabled(true);
116     jButton2.setVisible(true);
117     jButton1.setForeground(Color.lightGray);
118     // TODO add your handling code here:
119 }
120
121 private void findSeatsNPrice() {
122     Connection conn=null;
123     PreparedStatement pst=null;
124     ResultSet rs=null;
125     int flag=0;
126     try {
127         Class.forName("com.mysql.cj.jdbc.Driver");
128         conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+ "user=root");
129         String q="select seats,nofseat,movie from ticket where id=1 and date=? and time=?";
130         pst=conn.prepareStatement(q);
131         pst.setString(1, Date);
132         pst.setString(2, Time);
133         rs=pst.executeQuery();
134         String bookedSeats=null;
135         while(rs.next()) {
136             bookedSeats=rs.getString("seats");
137             numSeat = rs.getInt("nofseat");
138             // SwingUtilities.invokeLater(() -> {
139             //     jLabel5.setText("");
140             //     jLabel7.setText("");
141             //     jLabel5.setText(text: bookedSeats);
142             //     jLabel7.setText(text: Integer.toString(numSeat));
143             // });
144             System.out.println("Seats: " + bookedSeats + " No. of Seats: " + numSeat + " Price: " + price);
145         }
146     }
```



```

146 ..... System.out.println("Seats: " + bookedSeats + " | No. of Seats: " + numSeat + " | Price: " + price);
147 ..... movie = rs.getString(string: "movie");
148 ..... }
149 ..... if (bookedSeats == null || numSeat == 0 || movie == null)
150 ..... {
151 .....     flag = 1;
152 ..... }
153 ..... rs.close();
154 ..... pst.close();
155 ..... }
156 ..... catch (Exception e)
157 ..... {
158 .....     System.out.println(x: e);
159 ..... }
160 ..... if (flag == 0)
161 ..... {
162 .....     try
163 .....     {
164 .....         Class.forName(classname: "com.mysql.cj.jdbc.Driver");
165 .....         conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+ "user=root");
166 .....         String q = "select price from id where movie=? and date=? and time=?";
167 .....         pst = conn.prepareStatement(string: q);
168 .....         pst.setString(i: 1, string: movie);
169 .....         pst.setString(i: 2, string: Date);
170 .....         pst.setString(i: 3, string: Time);
171 .....         rs = pst.executeQuery();
172 .....         while (rs.next())
173 .....         {
174 .....             float onePrice = rs.getFloat(string: "price");
175 .....             price = (float) numSeat * onePrice;
176 .....             JLabel6.setText(text: Float.toString(f: price));
177 .....         }
178 .....         rs.close();
179 .....         pst.close();
180 .....     }
181 .....     catch (Exception e)
182 .....     {
183 .....         System.out.println(x: e);
184 .....     }
185 ..... }

```

```

182 ..... System.out.println(x: e);
183 ..... }
184 ..... try
185 ..... {
186 .....     Class.forName(classname: "com.mysql.cj.jdbc.Driver");
187 .....     conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+ "user=root");
188 .....     String q = "update ticket set price = ? where id = 1 and date = ? and time = ?";
189 .....     pst = conn.prepareStatement(string: q);
190 .....     pst.setFloat(i: 1, f: price);
191 .....     pst.setString(i: 2, string: Date);
192 .....     pst.setString(i: 3, string: Time);
193 .....     pst.executeUpdate();
194 .....     pst.close();
195 ..... }
196 ..... catch (Exception e)
197 ..... {
198 .....     System.out.println(x: e);
199 ..... }
200 ..... else
201 ..... {
202 .....     flag = 0;
203 ..... }
204 ..... catch (Exception e)
205 ..... {
206 .....     System.out.println(x: e);
207 ..... }
208 ..... }
209 ..... /**
210 .....  * @param args the command line arguments
211 .....  */
212 ..... public static void main(String args[]) {
213 .....     /* Set the Nimbus look and feel */
214 .....     Look and feel setting code (optional)
215 ..... }
216 ..... /* Create and display the form */
217 ..... java.awt.EventQueue.invokeLater(new Runnable() {
218 .....     public void run() {
219 .....         new PayNConfirm().setVisible(true);
220 .....     }
221 ..... });

```

```

206     System.out.println(s: e);
207 }
208 }
209 /**
210  * @param args the command line arguments
211  */
212 public static void main(String args[]) {
213     /** Set the Nimbus look and feel */
214     Look and feel setting code (optional)
215
216     /** Create and display the form */
217     java.awt.EventQueue.invokeLater(new Runnable() {
218         public void run() {
219             new PayNConfirm().setVisible(true);
220         }
221     });
222 }
223
224 // Variables declaration - do not modify
225 private javax.swing.JButton jButton1;
226 private javax.swing.JButton jButton2;
227 private javax.swing.JLabel jLabel2;
228 private javax.swing.JLabel jLabel3;
229 private javax.swing.JLabel jLabel4;
230 private javax.swing.JLabel jLabel5;
231 private javax.swing.JLabel jLabel6;
232 private javax.swing.JLabel jLabel7;
233 private javax.swing.JLabel jLabel8;
234 // End of variables declaration
235
236 }
237

```

Ticket Printing (Print.java):

```

1 package pl;
2 import java.util.*;
3 import java.sql.*;
4 import java.awt.*;
5 import javax.swing.*;
6
7 /**
8  *
9  * @author ayush
10  */
11 public class Print extends javax.swing.JFrame {
12
13     /** Creates new form Print */
14     /** String Date, Time, movie */
15     private static int id=1000;
16     public Print(String Date, String Time, String movie) {
17         this.Date=Date;
18         this.Time=Time;
19         this.movie=movie;
20         initComponents();
21         print();
22     }
23
24     public static synchronized int getNextUniqueID(int id) {
25         return ++id;
26     }
27
28     /** This method is called from within the constructor to initialize the form.
29     WARNING: Do NOT modify this code. The content of this method is always
30     regenerated by the Form Editor.
31     */
32     @SuppressWarnings("unchecked")
33     Generated Code
34
35     private void print() {
36         Connection conn=null;
37

```

```

Start Page x Print.java x
Source Design History
220 {
221     Connection conn=null;
222     PreparedStatement pst=null;
223     ResultSet rs=null;
224     try{
225         try{
226             Class.forName(className:"com.mysql.cj.jdbc.Driver");
227             conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+"user=root");
228             String q="select username,seats,nofseat,price,movie_id from ticket where id=? and date=? and time=?";
229             pst=conn.prepareStatement(string:q);
230             pst.setString(1, string:Date);
231             pst.setString(2, string:Time);
232             rs=pst.executeQuery();
233             while(rs.next())
234             {
235                 String user:=rs.getString(string:"username");
236                 String seating:=rs.getString(string:"seats");
237                 int numSeats:=rs.getInt(string:"nofseat");
238                 float price:=rs.getFloat(string:"price");
239                 int movie_id:=rs.getInt(string:"movie_id");
240                 JLabel12.setText(text: user);
241                 JLabel13.setText(text: movie);
242                 JLabel14.setText(text: Date);
243                 JLabel15.setText(text: Time);
244                 JLabel16.setText(text: Integer.toString(1: numSeats));
245                 JLabel11.setText(text: Float.toString(1: price));
246                 System.out.println(1: price);
247                 JTextArea1.setEditable(1: false);
248                 JTextArea1.setLineWrap(1: true);
249                 JTextArea1.setWrapStyleWord(1: true);
250                 JTextArea1.setText(1: "");
251                 JTextArea1.append(text:seating);
252             }
253             rs.close();
254             pst.close();
255         }
256     }catch (Exception e){
257         {

```

```

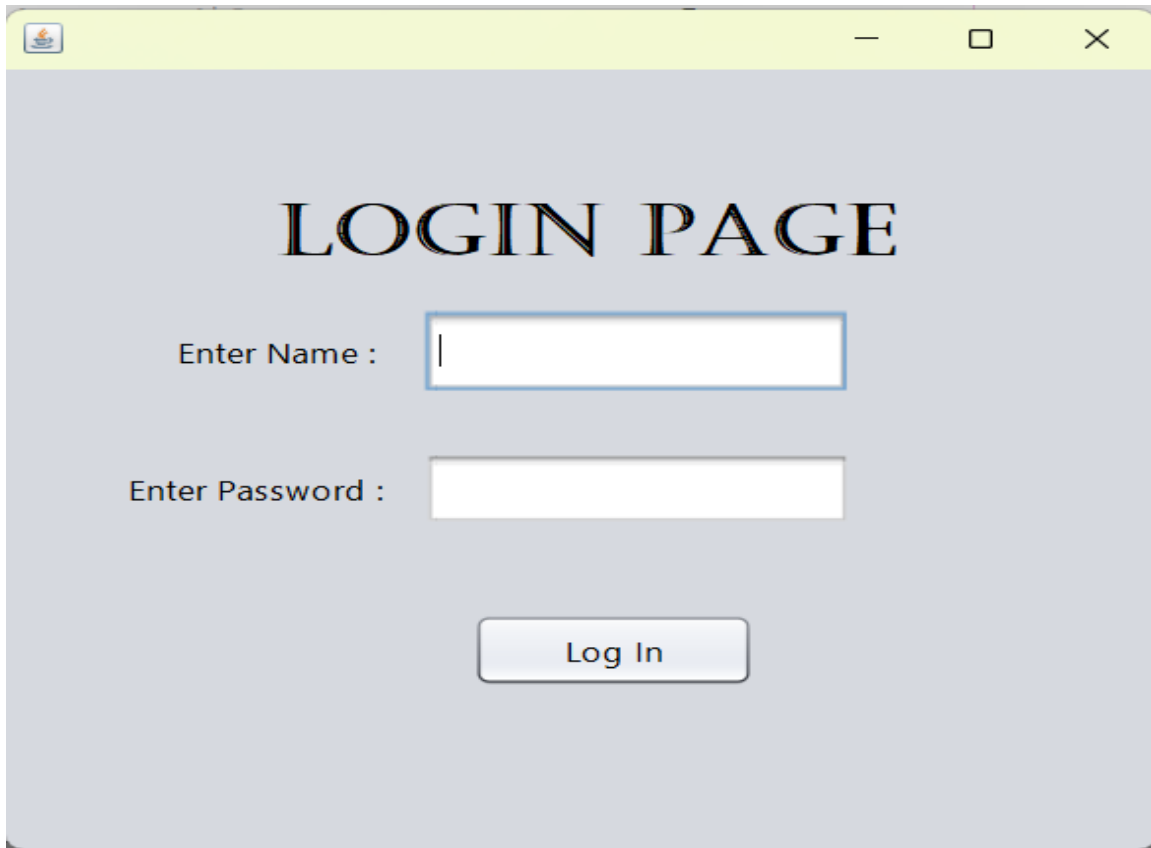
Start Page x Print.java x
Source Design History
257     }catch (Exception e){
258         System.out.println(1: e);
259     }
260     try{
261         Class.forName(className:"com.mysql.cj.jdbc.Driver");
262         conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/test?"+"user=root");
263         String q="update ticket set id=? where id=1;";
264         pst=conn.prepareStatement(string:q);
265         int unique_id=getNextUniqueID(id);
266         //String unique_id:=UUID.randomUUID().toString();
267         pst.setInt(1, 1: unique_id);
268         System.out.println(1: unique_id);
269         pst.executeUpdate();
270     }
271     }catch (Exception e){
272         {
273             System.out.println(1: e);
274         }
275     }catch (Exception e){
276         {
277             System.out.println(1: e);
278         }
279     }
280     /**
281     * @param args the command-line arguments
282     */
283     public static void main(String args[]) {
284         /** Set the Nimbus look and feel */
285         Look and feel setting code (optional)
286     }
287     /** Create and display the form */
288     // java.awt.EventQueue.invokeLater(new Runnable() {
289     // public void run() {
290     // new Print().setVisible(true);
291     // }
292     // })
293 }

```

```
309 //.....public void run() {
310 //.....new Print().setVisible(true);
311 //.....}
312 //.....}
313 }
314
315 //Variables declaration--do not modify-----
316 private javax.swing.JLabel jLabel1;
317 private javax.swing.JLabel jLabel10;
318 private javax.swing.JLabel jLabel11;
319 private javax.swing.JLabel jLabel12;
320 private javax.swing.JLabel jLabel13;
321 private javax.swing.JLabel jLabel14;
322 private javax.swing.JLabel jLabel15;
323 private javax.swing.JLabel jLabel16;
324 private javax.swing.JLabel jLabel17;
325 private javax.swing.JLabel jLabel18;
326 private javax.swing.JLabel jLabel19;
327 private javax.swing.JLabel jLabel2;
328 private javax.swing.JLabel jLabel20;
329 private javax.swing.JLabel jLabel3;
330 private javax.swing.JLabel jLabel4;
331 private javax.swing.JLabel jLabel5;
332 private javax.swing.JScrollPane jScrollPane1;
333 private javax.swing.JTextArea jTextArea1;
334 //End of variables declaration-----
335 }
336 }
```

336.1 | INS Unix (LF)

Results



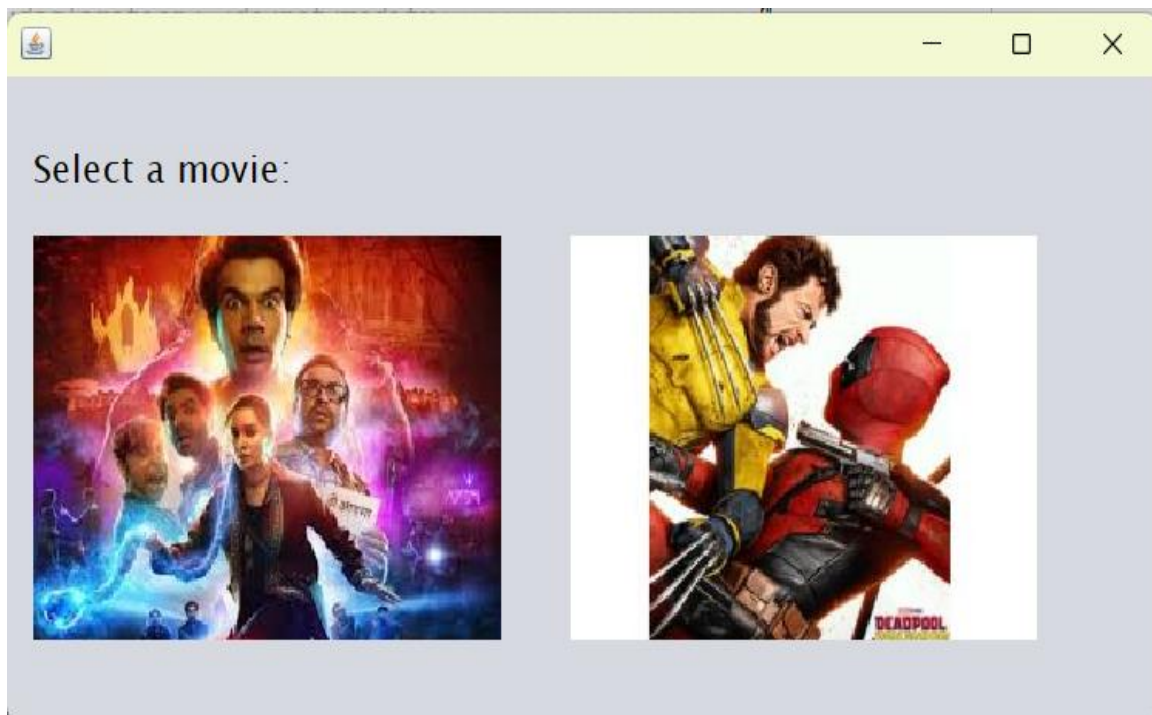
A screenshot of a web browser window with a light green title bar. The page has a light gray background and the title "LOGIN PAGE" in a large, black, serif font. Below the title, there are two input fields. The first is labeled "Enter Name :" and contains a single vertical line cursor. The second is labeled "Enter Password :". Below these fields is a button with the text "Log In".

LOGIN PAGE

Enter Name :

Enter Password :

Log In



—□×

Date :

7 OCT

8 OCT

9 OCT

10 OCT

Timings :

08:45

12:00

15:15

18:30

NEXT

—□×

Next

A1

A2

A3

A4

A5

B1

B2

B3

B4

B5

C1

C2

C3

C4

C5

D1

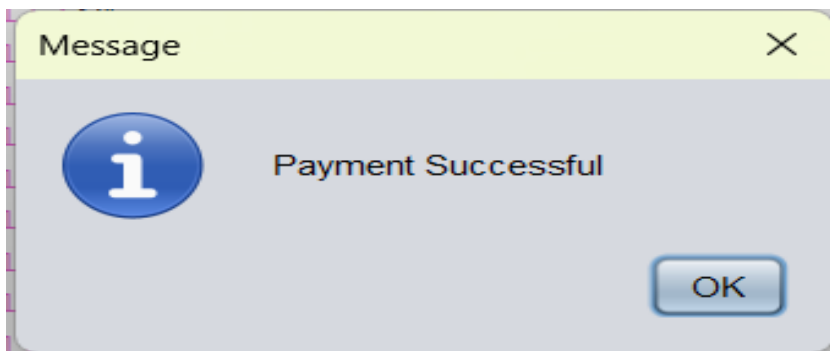
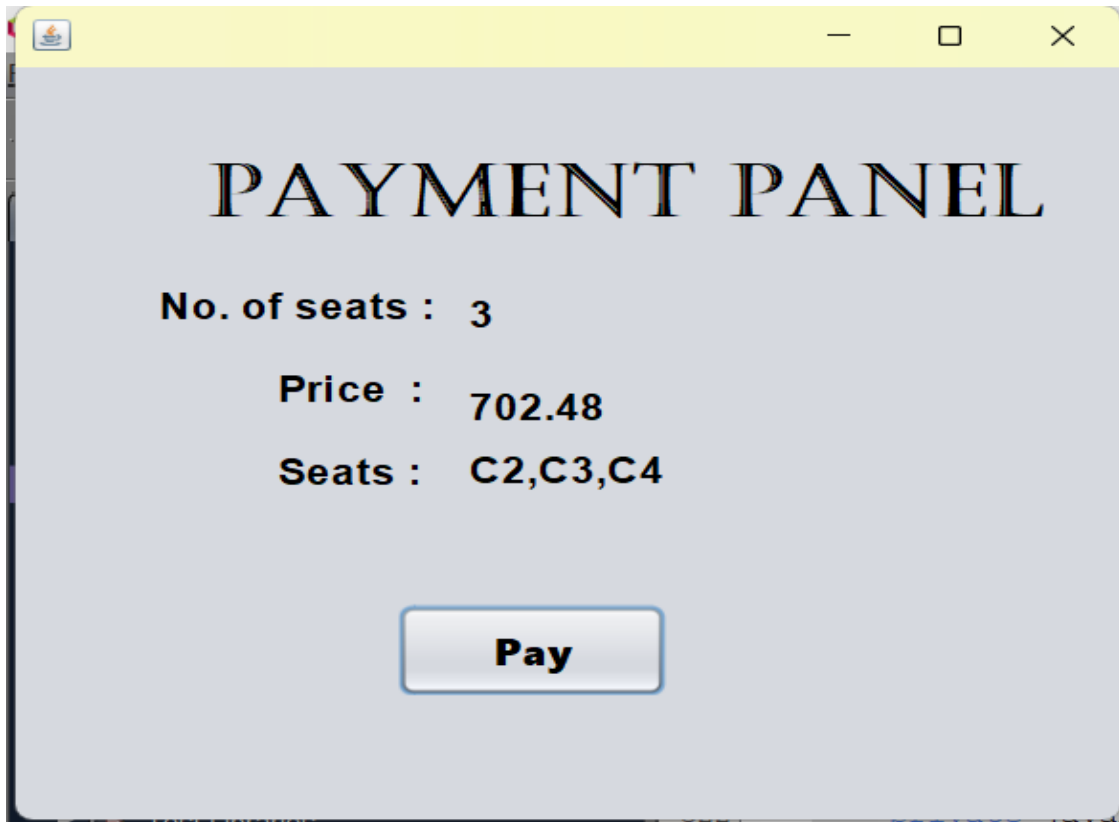
D2

D3

D4

D5

Screen



— □ ×

Movie Ticket

Username : Ayush

Movie Details:

Movie Name : Stree2

Date : 8 OCT Time : 12:00

Seats : No. of seats : 3

C2,C3,C4

Price : Rs. 702.48

Conclusion

In conclusion, our movie ticket booking system, though simple and limited to two movie options, effectively addresses the core challenges users face with traditional booking methods. By offering real-time seat availability, streamlined showtime selection, and a user-friendly interface, the system enhances the overall booking experience. It provides a fast, clear, and reliable solution for users, ensuring that even the most basic functionalities are executed with efficiency and ease. This project demonstrates that simplicity, when combined with thoughtful design, can still deliver a seamless and enjoyable user experience.

References

- **Official Java documentation:** <https://docs.oracle.com/javase/8/docs/>.
- **SQL Tutorial:** <https://www.w3schools.com/sql/>.