

```
1 package com.company;
2
3 import java.security.NoSuchAlgorithmException;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.text.SimpleDateFormat;
7 import java.util.ArrayList;
8 import java.util.Calendar;
9 import java.util.Locale;
10 import java.util.Scanner;
11 import java.util.concurrent.atomic.AtomicReference;
12
13 public class AccountManager {
14     private final Scanner keyboard = new Scanner(
15         System.in);
16     private String username; //cannot be declared
17     final
18     private final MysqlExecute mysqlExecute;
19
20     public AccountManager(){
21         mysqlExecute = new MysqlExecute();
22     }
23     public String getUsername() {
24         return username;
25     }
26     public int getUserId(String username){
27         int userId = 0;
28         try{
29             ResultSet resultSet = mysqlExecute.
30             mysqlQuery("SELECT userId FROM UserList WHERE
31             username=\"" + username + "\";");
32             if(resultSet == null){
33                 throw new SQLException();
34             }
35             resultSet.next();
36             userId = resultSet.getInt(1);
37         } catch (SQLException exception){
38             exception.printStackTrace();
39         }
40         return userId;
41     }
42 }
```

```

38
39     public void signUp() throws
40         NoSuchAlgorithmException {
41         final String signingUsername, password,
42         passwordHint, userAddStatement;
43         final int interestedBranchId;
44         System.out.println("Please enter your
45             username:");
46         signingUsername = keyboard.nextLine();
47         if(signingUsername.equals("admin")){
48             System.out.println("Cannot sign-up to
49                 Administrator account in client version.");
50             return;
51         }
52
53         System.out.println("Please enter your
54             password:");
55         StringBuilder saltString = Sha256Hash.
56         generateSaltString();
57         password = Sha256Hash.doubleSHA256(keyboard
58         .nextLine(), saltString);
59
60         System.out.println("Please confirm (re-
61             enter) your password:");
62         final String passwordConfirm = Sha256Hash.
63         doubleSHA256(keyboard.nextLine(), saltString);
64         if(!password.equals(passwordConfirm)){
65             System.out.println("Password Does not
66                 match. Please Try Again.");
67             return;
68         }
69
70         System.out.println("Please enter your
71             password hint (optional):");
72         passwordHint = keyboard.nextLine();
73         System.out.println("Options available: 1.
74             Holidays or Other Special Days\t2. Computers\t3.
75             Politics");
76         System.out.println("Please choose your
77             interested subject:");
78         switch (keyboard.nextLine().toLowerCase(

```

```

64 Locale.ROOT).replaceAll("\\s", "").toLowerCase(
    Locale.ROOT)){
65         case "holidays", "otherspecialdays", "
holidaysorotherspecialdays", "1" ->
interestedBranchId = 1;
66         case "computers", "computer", "2" ->
interestedBranchId = 2;
67         case "politics", "politic", "3" ->
interestedBranchId = 3;
68         default -> {
69             System.out.println("Invalid option
.");
70             return;
71         }
72     }
73
74     try {
75         ResultSet userList = mysqlExecute.
mysqlQuery("SELECT username FROM UserList");
76         if(userList == null){
77             throw new SQLException();
78         }
79         while(userList.next()){
80             if(signingUsername.equals(userList
.getString(1))){
81                 System.out.println("Username
already exists!");
82                 return;
83             }
84         }
85         userAddStatement = "INSERT INTO
UserList(username, password, salt, passwordHint,
interestedBranchId, streak, streakLevel) VALUES (
\"" + signingUsername + "\",\"" + password + "\",
\"" + saltString + "\",\"" + passwordHint + "\",
+ interestedBranchId + ", 0, \"Newbie\");";
86         int resultValue = mysqlExecute.
mysqlUpdate(userAddStatement);
87         if(resultValue == -1){
88             throw new SQLException();
89         } else if(!

```

```

89 addInterestedSchedulesToTaskList(getUserId(
    signingUsername),interestedBranchId)){
90             System.out.println("Failed to
    allocate automatically-generated Interest-based
    schedules.");
91         } else{
92             System.out.println("User " +
    signingUsername + " successfully registered.");
93         }
94     } catch (SQLException exception){
95         exception.printStackTrace();
96     }
97 }
98 private boolean
addInterestedSchedulesToTaskList(int userId, int
branchId){
99     if(userId == 1){
100         System.out.println("You cannot modify/
    change anything about Administrator's account on
    the client version.");
101         return false;
102     }
103     try {
104         ArrayList<Integer> eventIds = new
ArrayList<>();
105         ResultSet eventsSet = mysqlExecute.
mysqlQuery("SELECT eventId, eventName, eventDate
    FROM InterestSchedule WHERE branchId = " +
branchId + ";");
106         if (eventsSet == null) {
107             return false;
108         }
109         while (eventsSet.next()) {
110             eventIds.add(eventsSet.getInt(1));
111         }
112         for (int eventId : eventIds) {
113             if (mysqlExecute.mysqlUpdate("
    INSERT INTO TaskList(userId, interestScheduleId)
    VALUES (" + userId + "," + eventId + ");") == -1
        ) {
114                 return false;

```

```

115             }
116         }
117     } catch (SQLException exception){
118         return false;
119     }
120     return true;
121 }
122
123     public void removeAccount(int userId) throws
124     SQLException {
125         if(userId == 1){
126             System.out.println("You cannot delete
127             Administrator's account on the client version.");
128             return;
129         }
130         if(mysqlExecute.mysqlUpdate("DELETE FROM
131             UserList WHERE userId = " + userId + ";") == -1){
132             throw new SQLException();
133         }
134         System.out.println("User \\" + username +
135             "\\" successfully deleted. Hope we see again.");
136     }
137
138     public boolean login() throws SQLException,
139     NoSuchAlgorithmException{
140         final String password, usernameTemp; // 
141         usernameTemp for security reasons
142         System.out.println("Please enter your
143             username:");
144         usernameTemp = keyboard.nextLine();
145         if(usernameTemp.equals("admin")){
146             System.out.println("Cannot login as
147             Administrator in a client version.");
148             return false;
149         }
150
151         if(usernameTemp.equals("")){
152             System.out.println("Username cannot be
153             a Blank");
154             return false;
155         }

```

```

147
148         String saltString = Sha256Hash.
149             getSaltString(usernameTemp);
150         boolean isUsernameWrong = saltString.
151             equals("-1");
152         System.out.println("Please enter your
153             password:");
154         password = Sha256Hash.doubleSHA256(
155             keyboard.nextLine(), new StringBuilder(saltString
156             ));
157
158         try {
159             ResultSet resultSet = mysqlExecute.
160             mysqlQuery("SELECT username, password FROM
161             UserList;");
162             if(resultSet == null){
163                 return false;
164             }
165             if(!isUsernameWrong) {
166                 while (resultSet.next()) {
167                     if (resultSet.getString(1).
168                         equals(usernameTemp) && resultSet.getString(2).
169                         equals(password)) {
170                         username = usernameTemp;
171                         System.out.println("Welcome " + username + "!\nNow you are logged in
172                         as " + username + ".");
173                         return true;
174                     }
175                 }
176             } //implicit else -> loginSucceed
177             = false
178         }
179         System.out.println("Login failed.\n
180             Please check your username and password.");
181     } catch (SQLException exception) {
182         exception.printStackTrace();
183     }
184     return false;
185 }
186

```

```

175     public void printStreakAndStreakLevel(int
176         userId) throws SQLException {
177         ResultSet streaksAndItsNumberResultSet =
178             mysqlExecute.mysqlQuery("SELECT userId, streak,
179             streakLevel FROM UserList");
180         if(streaksAndItsNumberResultSet == null){
181             throw new SQLException();
182         }
183         streaksAndItsNumberResultSet.next();
184         while(streaksAndItsNumberResultSet.next
185             () ){
186             if(userId ==
187                 streaksAndItsNumberResultSet.getInt(1)){
188                 System.out.println("You have " +
189                     streaksAndItsNumberResultSet.getInt(2) + " streaks
190                     .\t\tYou are currently '" +
191                     streaksAndItsNumberResultSet.getString(3) + "'"
192                     Level );
193             }
194         }
195         public boolean logout(){ //erase user's data
196             for security. (final and method scope)
197                 username = "";
198             return true;
199         }
200
201         interface EnterDateInterface {
202             boolean formatStringDate(String month);
203             boolean formatNumberedDate();
204             boolean formatThisDateOfWeekDate();
205             boolean formatNextDateOfWeekDate();
206             boolean formatTomorrowDate();
207             boolean formatTodayDate();
208         }
209
210         private String enterDateSchedule(){ //function

```

```

205  for taking and processing user-friendly date
      input
206
207      System.out.println("Proper formats (put
          only 1 digit of month if the month is before
          november)\t(enter \"tmr\" or \"today\" for
          scheduling tomorrow and today each):\nString
          format: MM dd YYYY HH:mm:ss\texample: feb 15 2021
          01:02:03 or\nNumber format: MM/dd/YYYY HH:mm:ss\t
          example: 2/15/2021 01:02:03\nDate of Week format
          : (put \"nxt\" to indicate next week) +
          abbreviated_string_of_date_of_week\texample: \"nxt
          fri\" & \"tue\"");
208
209      AtomicReference<String> date = new
      AtomicReference<>();
210      boolean isFormattingSuccess = false;
211
212      do {
213          date.set(keyboard.nextLine().
          toLowerCase(Locale.ROOT));
214          int dateEnteredLength = date.get().
          length();
215          EnterDateInterface formatDateNumber =
          new EnterDateInterface() {
216
217              final SimpleDateFormat timeFormat
              = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
218
219              @Override
220              public boolean formatStringDate(
          String month) {
221                  //format string form of date
                  //form of string: Ex. feb 15
222                  // 2021 HH:mm:ss
223
224                  if (dateEnteredLength == 20
          ) { //if day number is 2 digit
225                      //negative string index
                      value is the same for both length after the
                      varying digit day number

```

```

226                     date.set(date.get().
227                         substring(dateEnteredLength - 13,
228                             dateEnteredLength - 9) + "-" + month + "-" + date.
229                             get().substring(dateEnteredLength - 16,
230                               dateEnteredLength - 14) +
231                                     " " + date.get().
232                         substring(dateEnteredLength - 8, dateEnteredLength
233                           - 6) + ":" + date.get().substring(
234                             dateEnteredLength - 5, dateEnteredLength - 3) +
235                               ":" +
236                                     + date.get().
237                         substring(dateEnteredLength - 13,
238                             dateEnteredLength - 9) + "-" + month + "-0" + date
239                             .get().charAt(4) +
240                                     " " + date.get().
241                         substring(dateEnteredLength - 8, dateEnteredLength
242                           - 6) + ":" + date.get().substring(
243                             dateEnteredLength - 5, dateEnteredLength - 3) +
244                               ":" +
245                                     + date.get().
246                         substring(dateEnteredLength - 2));
247                     } else if (dateEnteredLength
248                         == 19) { //if day number is 1 digit
249                         date.set(date.get().
250                             substring(dateEnteredLength - 13,
251                                 dateEnteredLength - 9) + "-" + month + "-0" + date
252                                 .get().charAt(4) +
253                                     " " + date.get().
254                         substring(dateEnteredLength - 8, dateEnteredLength
255                           - 6) + ":" + date.get().substring(
256                             dateEnteredLength - 5, dateEnteredLength - 3) +
257                               ":" +
258                                     + date.get().
259                         substring(dateEnteredLength - 2));
260                     } else {
261                         System.out.println("The
262                             entered date does not follow the format. Please
263                             recheck");
264                         return false;
265                     }
266                     return true;
267                 }
268             }
269
270             @Override
271             public boolean formatNumberedDate
272                 () {
273                     //format number form of date
274                     //form of number: Ex. 2/15/
275                     2021 HH:mm:ss
276                     String monthDigit = date.get

```

```

244 (.substring(0, 2);
245             String dayDigit = date.get().
246                 substring(2, 5);
247             int monthDigitCount = 0;
248             int dayDigitCount = 0;
249             for(int i = 0; i < 2; i++){
250                 int monthCharacter =
251                     monthDigit.charAt(i);
252                     int dayCharacter =
253                         if(Character.toString(
254                             monthCharacter).matches("-?\\d+")){
255                             monthDigitCount++;
256                         }
257                         if(Character.toString(
258                             dayCharacter).matches("-?\\d+")){
259                             dayDigitCount++;
260                         }
261                         if(monthDigitCount == 2){ //if
262                             month digit is 2...
263                             if(dayDigitCount == 2 &&
264                             dateEnteredLength == 19){ //and day is digit is 2
265                                 date.set(date.get().
266                                     substring(dateEnteredLength - 13,
267                                     dateEnteredLength - 9) + "-" + date.get().
268                                     substring(0 , dateEnteredLength - 17) + "-" + date
269                                     .get().substring(dateEnteredLength - 16,
270                                     dateEnteredLength - 14) +
271                                         " " + date.get()
272                                         .substring(dateEnteredLength - 8,
273                                         dateEnteredLength - 6) + ":" + date.get().
274                                         substring(dateEnteredLength - 5, dateEnteredLength
275                                         - 3) + ":" +
276                                         + date.get().
277                                         substring(dateEnteredLength - 2));
278                         } else if(dayDigitCount
279                         == 1 && dateEnteredLength == 18){ //and day is
280                             digit 1

```

```

266                               date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-" + date.get().
    substring(0 , dateEnteredLength - 16) + "-0" +
    date.get().charAt(3) +
267                                         " " + date.get
    ().substring(dateEnteredLength - 8,
    dateEnteredLength - 6) + ":" + date.get().
    substring(dateEnteredLength - 5, dateEnteredLength
    - 3) + ":" +
268                                         + date.get().
    substring(dateEnteredLength - 2));
269             } else {
270                 System.out.println("
The entered date does not follow the format.
Please recheck");
271             return false;
272         }
273     } else if(monthDigitCount == 1
274     ){ //if month digit is 1...
275         if(dayDigitCount == 2 &&
    dateEnteredLength == 18){ //and day is digit 2
276             date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-0" + date.get().charAt(
    0) + "-" + date.get().substring(dateEnteredLength
    - 16, dateEnteredLength - 14) +
277                                         " " + date.get
    ().substring(dateEnteredLength - 8,
    dateEnteredLength - 6) + ":" + date.get().
    substring(dateEnteredLength - 5, dateEnteredLength
    - 3) + ":" +
278                                         + date.get().
    substring(dateEnteredLength - 2));
279         } else if(dayDigitCount
    == 1 && dateEnteredLength == 17){ //and day is
    digit 1
280             date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-0" + date.get().charAt(
    0) + "-0" + date.get().charAt(2) +

```

```

280                                     " " + date.get
281                                     ().substring(dateEnteredLength - 8,
282                                     dateEnteredLength - 6) + ":" + date.get().
283                                     substring(dateEnteredLength - 5, dateEnteredLength
284                                     - 3) + ":" +
285                                     + date.get().
286                                     substring(dateEnteredLength - 2));
287                                     } else {
288                                     System.out.println("The entered date does not follow the format.
289                                     Please recheck");
290                                     return false;
291                                     }
292                                     }
293                                     @Override
294                                     public boolean
295                                     formatThisDateOfWeekDate(){
296                                     Calendar timeNowCal = Calendar
297                                     .getInstance();
298                                     int timeNowDayOfWeek =
299                                     timeNowCal.get(Calendar.DAY_OF_WEEK);
300                                     Calendar timeScheduled =
301                                     Calendar.getInstance();
302                                     timeScheduled.set(Calendar.
303                                     HOUR, 11);
304                                     timeScheduled.set(Calendar.
305                                     MINUTE, 59);
306                                     timeScheduled.set(Calendar.
307                                     SECOND, 59);
308                                     timeScheduled.set(Calendar.
309                                     DAY_OF_WEEK, timeNowDayOfWeek);

```

```

304
305                     date.set(timeFormat.format(
306                         timeScheduled.getTime()));
307                     return true;
308
309                     @Override
310                     public boolean
311                         formatNextDateOfWeekDate(){
312                             Calendar timeNowCal = Calendar
313                             .getInstance();
314                             int timeNowWeekOfYear =
315                             timeNowCal.get(Calendar.WEEK_OF_YEAR);
316
317                             Calendar timeScheduled =
318                             Calendar.getInstance();
319                             timeScheduled.set(Calendar.
320                               HOUR, 11);
321                             timeScheduled.set(Calendar.
322                               MINUTE, 59);
323                             timeScheduled.set(Calendar.
324                               SECOND, 59);
325
326                             timeScheduled.set(Calendar.
327                               WEEK_OF_YEAR, timeNowWeekOfYear + 1);
328
329                             String dateOfWeekEntered =
330                             date.get().substring(date.get().length() - 4).
331                             strip();
332                             switch (dateOfWeekEntered) {
333                               case "mon" ->
334                                 timeScheduled.set(Calendar.DAY_OF_WEEK, 2);
335
336                               case "tues", "tue" ->
337                                 timeScheduled.set(Calendar.DAY_OF_WEEK, 3);
338
339                               case "wed" ->
340                                 timeScheduled.set(Calendar.DAY_OF_WEEK, 4);
341
342                               case "thu", "thur", "hurs"
343                                 -> timeScheduled.set(Calendar.DAY_OF_WEEK, 5);

```

```
330
331                     case "fri" ->
332             timeScheduled.set(Calendar.DAY_OF_WEEK, 6);
333                     case "sat" ->
334             timeScheduled.set(Calendar.DAY_OF_WEEK, 7);
335                     case "sun" ->
336             timeScheduled.set(Calendar.DAY_OF_WEEK, 1);
337         }
338         date.set(timeFormat.format(
339             timeScheduled.getTime()));
340         return true;
341     }
342     @Override
343     public boolean formatTodayDate(){
344         Calendar timeScheduled =
345             Calendar.getInstance();
346         timeScheduled.set(Calendar.
347             HOUR, 11);
348         timeScheduled.set(Calendar.
349             MINUTE, 59);
350         timeScheduled.set(Calendar.
351             SECOND, 59);
352         date.set(timeFormat.format(
353             timeScheduled.getTime()));
354         return true;
355     }
356     @Override
357     public boolean formatTomorrowDate
358 (){
359         Calendar timeScheduled =
360             Calendar.getInstance();
361         timeScheduled.set(Calendar.
362             HOUR, 11);
363         timeScheduled.set(Calendar.
364             MINUTE, 59);
```

```

358                     timeScheduled.set(Calendar.
359                         SECOND, 59);
359                     timeScheduled.set(Calendar.
360                         DAY_OF_MONTH, timeScheduled.get(Calendar.
361                         DAY_OF_MONTH) + 1);
360
361                     date.set(timeFormat.format(
362                         timeScheduled.getTime()));
362                     return true;
363                 }
364             };
365
366
367             if (!(date.get().substring(0,1).
368                 matches("-?\\d+"))) {
368                     switch (date.get().substring(0, 3
369 )) {
369                         case "tmr", "tom" ->
370                             isFormattingSuccess = formatDateNumber.
371                             formatTomorrowDate();
370
371                         case "tod" ->
372                             isFormattingSuccess = formatDateNumber.
373                             formatTodayDate();
372
373                         case "jan" ->
374                             isFormattingSuccess = formatDateNumber.
375                             formatStringDate("01");
374
375                         case "feb" ->
376                             isFormattingSuccess = formatDateNumber.
377                             formatStringDate("02");
376
377                         case "mar" ->
378                             isFormattingSuccess = formatDateNumber.
379                             formatStringDate("03");
378
379                         case "apr" ->
380                             isFormattingSuccess = formatDateNumber.
380                             formatStringDate("04");
380

```

```
381                     case "may" ->
382                         isFormattingSuccess = formatDateNumber.
383                         formatStringDate("05");
384
385                     case "jun" ->
386                         isFormattingSuccess = formatDateNumber.
387                         formatStringDate("06");
388
389                     case "jul" ->
390                         isFormattingSuccess = formatDateNumber.
391                         formatStringDate("07");
392
393                     case "aug" ->
394                         isFormattingSuccess = formatDateNumber.
395                         formatStringDate("08");
396
397                     case "sep" ->
398                         isFormattingSuccess = formatDateNumber.
399                         formatStringDate("09");
400
401                     case "oct" ->
402                         isFormattingSuccess = formatDateNumber.
403                         formatStringDate("10");
404
405                     case "nov" ->
406                         isFormattingSuccess = formatDateNumber.
407                         formatStringDate("11");
408
409                     case "dec" ->
410                         isFormattingSuccess = formatDateNumber.
411                         formatStringDate("12");
412
413                     case "nxt", "nex" ->
414                         isFormattingSuccess = formatDateNumber.
415                         formatNextDateOfWeekDate();
416
417                     case "mon", "tue", "wed", "thu",
418                         "fri", "sat", "sun" -> isFormattingSuccess =
419                         formatDateNumber.formatThisDateOfWeekDate();
420
421                     default -> {
422                         //implicit
```

```

401     isFormattingSuccess = false
402             }
403         }
404     } else{
405         isFormattingSuccess =
406         formatDateNumber.formatNumberedDate();
407     }
408 } while (!isFormattingSuccess);
409
410     return String.valueOf(date);
411 }
412
413     public void addManualSchedule(int userId)
414     throws SQLException {
415         if(userId == 1){
416             System.out.println("You cannot modify/
417             change anything about Administrator's account on
418             the client version.");
419             return;
420         }
421         String descrpSchedule;
422         final String dateSchedule;
423         final String manualScheduleAddStatement;
424         System.out.println("To add a schedule,
425             first provide a description of the event:");
426         descrpSchedule = keyboard.nextLine();
427
428         System.out.println("Second, provide the
429             due date of the event:");
430         dateSchedule = enterDateSchedule();
431
432         descrpSchedule = SchedulePriority.
433             addPriority(descrpSchedule); //assign priority
434
435         manualScheduleAddStatement = "INSERT INTO
436             TaskList(userId,manualScheduleName,
437             manualScheduleDate) VALUES (" + userId + ",\"" +
438             descrpSchedule + "\",\"" + dateSchedule + "\");";
439         int resultValue = mysqlExecute.mysqlUpdate
440             (manualScheduleAddStatement);

```

```

431         if (resultValue == -1) {
432             throw new SQLException();
433         } else{
434             System.out.println("Schedule " +
435             descrpSchedule + " successfully added.");
436         }
437     }
438     public void editSchedule(int userId) throws
439         SQLException, InterruptedException {
440         if(userId == 1){
441             System.out.println("You cannot modify/
442             change anything about Administrator's account on
443             the client version.");
444             return;
445         }
446         printTasks(userId,username); //give info
447             about schedules
448
449         final String newScheduleDate;
450         String newScheduleName;
451
452         if(removeMethodOption.equals("taskid")){
453             System.out.println("Please select from
454             schedules above: (by its Task ID)\n(cauution:
455             cannot delete interest-based schedule by its Task
456             ID)");
457             String enteredTaskId = keyboard.
458             nextLine();
459
460             System.out.println("Enter the new
461             description for the new schedule: ");
462             newScheduleName = keyboard.nextLine
463             (); //string is an object so value retains

```

```

458         System.out.println("Enter the new due
459             date for the new schedule: ");
460
461             newScheduleName = SchedulePriority.
462                 addPriority(newScheduleName); //assign priority
463
464             ResultSet taskIdsDB = mysqlExecute.
465                 mysqlQuery("SELECT taskId FROM TaskList WHERE
466                     userId = " + userId);
467
468             if(taskIdsDB == null){
469                 throw new SQLException();
470             }
471
472             while(taskIdsDB.next()){
473                 if(enteredTaskId.equals(taskIdsDB.
474                     getString(1))){
475                     //for checking same
476                     information update twice
477                     ResultSet
478                     manualScheduleNameAndDate= mysqlExecute.mysqlQuery
479                     ("SELECT manualScheduleName, manualScheduleDate
480                     FROM TaskList WHERE TaskId = \""
481                     + enteredTaskId
482                     + "\"");
483
484                     if(manualScheduleNameAndDate
485                     == null){
486                         throw new SQLException();
487                     }
488                     while (
489                         manualScheduleNameAndDate.next()){
490                             if(
491                                 manualScheduleNameAndDate.getString(1).equals(
492                                     newScheduleName) && manualScheduleNameAndDate.
493                                     getString(2).equals(newScheduleDate)){
494
495                             System.out.println("
496                             Cannot update a schedule with duplicate or same
497                             information.");
498
499                             return;
500
501                         }
502
503                     }
504
505                 }
506
507             }
508
509         }
510
511     }
512
513     public void updateSchedule(String newScheduleName, String newScheduleDate)
514     {
515
516         String sql = "UPDATE TaskList SET
517             scheduleName = ?,
518             scheduleDate = ? WHERE
519             TaskId = ?";
520
521         try {
522             PreparedStatement ps = connection.prepareStatement(sql);
523
524             ps.setString(1, newScheduleName);
525             ps.setString(2, newScheduleDate);
526             ps.setInt(3, enteredTaskId);
527
528             ps.executeUpdate();
529
530             System.out.println("Schedule updated successfully!");
531
532         } catch (SQLException e) {
533             e.printStackTrace();
534
535         }
536
537     }
538
539     public void deleteSchedule()
540     {
541
542         String sql = "DELETE FROM TaskList WHERE
543             TaskId = ?";
544
545         try {
546             PreparedStatement ps = connection.prepareStatement(sql);
547
548             ps.setInt(1, enteredTaskId);
549
550             ps.executeUpdate();
551
552             System.out.println("Schedule deleted successfully!");
553
554         } catch (SQLException e) {
555             e.printStackTrace();
556
557         }
558
559     }
560
561     public void printAllSchedules()
562     {
563
564         String sql = "SELECT * FROM TaskList";
565
566         try {
567             Statement st = connection.createStatement();
568
569             ResultSet rs = st.executeQuery(sql);
570
571             while(rs.next()){
572                 System.out.println("Task ID: " + rs.getInt("TaskId")
573                     + ", Schedule Name: " + rs.getString("scheduleName")
574                     + ", Schedule Date: " + rs.getString("scheduleDate"));
575
576             }
577
578         } catch (SQLException e) {
579             e.printStackTrace();
580
581         }
582
583     }
584
585     public void printScheduleByName(String scheduleName)
586     {
587
588         String sql = "SELECT * FROM TaskList WHERE
589             scheduleName = ?";
590
591         try {
592             PreparedStatement ps = connection.prepareStatement(sql);
593
594             ps.setString(1, scheduleName);
595
596             ResultSet rs = ps.executeQuery();
597
598             while(rs.next()){
599                 System.out.println("Task ID: " + rs.getInt("TaskId")
600                     + ", Schedule Name: " + rs.getString("scheduleName")
601                     + ", Schedule Date: " + rs.getString("scheduleDate"));
602
603             }
604
605         } catch (SQLException e) {
606             e.printStackTrace();
607
608         }
609
610     }
611
612     public void printScheduleByDate(String scheduleDate)
613     {
614
615         String sql = "SELECT * FROM TaskList WHERE
616             scheduleDate = ?";
617
618         try {
619             PreparedStatement ps = connection.prepareStatement(sql);
620
621             ps.setString(1, scheduleDate);
622
623             ResultSet rs = ps.executeQuery();
624
625             while(rs.next()){
626                 System.out.println("Task ID: " + rs.getInt("TaskId")
627                     + ", Schedule Name: " + rs.getString("scheduleName")
628                     + ", Schedule Date: " + rs.getString("scheduleDate"));
629
630             }
631
632         } catch (SQLException e) {
633             e.printStackTrace();
634
635         }
636
637     }
638
639     public void printScheduleById(int taskId)
640     {
641
642         String sql = "SELECT * FROM TaskList WHERE
643             TaskId = ?";
644
645         try {
646             PreparedStatement ps = connection.prepareStatement(sql);
647
648             ps.setInt(1, taskId);
649
650             ResultSet rs = ps.executeQuery();
651
652             while(rs.next()){
653                 System.out.println("Task ID: " + rs.getInt("TaskId")
654                     + ", Schedule Name: " + rs.getString("scheduleName")
655                     + ", Schedule Date: " + rs.getString("scheduleDate"));
656
657             }
658
659         } catch (SQLException e) {
660             e.printStackTrace();
661
662         }
663
664     }
665
666     public void printScheduleByPriority(int priority)
667     {
668
669         String sql = "SELECT * FROM TaskList WHERE
670             priority = ?";
671
672         try {
673             PreparedStatement ps = connection.prepareStatement(sql);
674
675             ps.setInt(1, priority);
676
677             ResultSet rs = ps.executeQuery();
678
679             while(rs.next()){
680                 System.out.println("Task ID: " + rs.getInt("TaskId")
681                     + ", Schedule Name: " + rs.getString("scheduleName")
682                     + ", Schedule Date: " + rs.getString("scheduleDate"));
683
684             }
685
686         } catch (SQLException e) {
687             e.printStackTrace();
688
689         }
690
691     }
692
693     public void printScheduleByNameAndDate(String scheduleName, String scheduleDate)
694     {
695
696         String sql = "SELECT * FROM TaskList WHERE
697             scheduleName = ? AND
698             scheduleDate = ?";
699
700         try {
701             PreparedStatement ps = connection.prepareStatement(sql);
702
703             ps.setString(1, scheduleName);
704             ps.setString(2, scheduleDate);
705
706             ResultSet rs = ps.executeQuery();
707
708             while(rs.next()){
709                 System.out.println("Task ID: " + rs.getInt("TaskId")
710                     + ", Schedule Name: " + rs.getString("scheduleName")
711                     + ", Schedule Date: " + rs.getString("scheduleDate"));
712
713             }
714
715         } catch (SQLException e) {
716             e.printStackTrace();
717
718         }
719
720     }
721
722     public void printScheduleByPriorityAndDate(int priority, String scheduleDate)
723     {
724
725         String sql = "SELECT * FROM TaskList WHERE
726             priority = ? AND
727             scheduleDate = ?";
728
729         try {
730             PreparedStatement ps = connection.prepareStatement(sql);
731
732             ps.setInt(1, priority);
733             ps.setString(2, scheduleDate);
734
735             ResultSet rs = ps.executeQuery();
736
737             while(rs.next()){
738                 System.out.println("Task ID: " + rs.getInt("TaskId")
739                     + ", Schedule Name: " + rs.getString("scheduleName")
740                     + ", Schedule Date: " + rs.getString("scheduleDate"));
741
742             }
743
744         } catch (SQLException e) {
745             e.printStackTrace();
746
747         }
748
749     }
750
751     public void printScheduleByPriorityAndName(int priority, String scheduleName)
752     {
753
754         String sql = "SELECT * FROM TaskList WHERE
755             priority = ? AND
756             scheduleName = ?";
757
758         try {
759             PreparedStatement ps = connection.prepareStatement(sql);
760
761             ps.setInt(1, priority);
762             ps.setString(2, scheduleName);
763
764             ResultSet rs = ps.executeQuery();
765
766             while(rs.next()){
767                 System.out.println("Task ID: " + rs.getInt("TaskId")
768                     + ", Schedule Name: " + rs.getString("scheduleName")
769                     + ", Schedule Date: " + rs.getString("scheduleDate"));
770
771             }
772
773         } catch (SQLException e) {
774             e.printStackTrace();
775
776         }
777
778     }
779
780     public void printScheduleByPriorityAndNameAndDate(int priority, String scheduleName, String scheduleDate)
781     {
782
783         String sql = "SELECT * FROM TaskList WHERE
784             priority = ? AND
785             scheduleName = ? AND
786             scheduleDate = ?";
787
788         try {
789             PreparedStatement ps = connection.prepareStatement(sql);
790
791             ps.setInt(1, priority);
792             ps.setString(2, scheduleName);
793             ps.setString(3, scheduleDate);
794
795             ResultSet rs = ps.executeQuery();
796
797             while(rs.next()){
798                 System.out.println("Task ID: " + rs.getInt("TaskId")
799                     + ", Schedule Name: " + rs.getString("scheduleName")
800                     + ", Schedule Date: " + rs.getString("scheduleDate"));
801
802             }
803
804         } catch (SQLException e) {
805             e.printStackTrace();
806
807         }
808
809     }
810
811     public void printScheduleByPriorityAndNameAndDateAndTime(int priority, String scheduleName, String scheduleDate, String scheduleTime)
812     {
813
814         String sql = "SELECT * FROM TaskList WHERE
815             priority = ? AND
816             scheduleName = ? AND
817             scheduleDate = ? AND
818             scheduleTime = ?";
819
820         try {
821             PreparedStatement ps = connection.prepareStatement(sql);
822
823             ps.setInt(1, priority);
824             ps.setString(2, scheduleName);
825             ps.setString(3, scheduleDate);
826             ps.setString(4, scheduleTime);
827
828             ResultSet rs = ps.executeQuery();
829
830             while(rs.next()){
831                 System.out.println("Task ID: " + rs.getInt("TaskId")
832                     + ", Schedule Name: " + rs.getString("scheduleName")
833                     + ", Schedule Date: " + rs.getString("scheduleDate")
834                     + ", Schedule Time: " + rs.getString("scheduleTime"));
835
836             }
837
838         } catch (SQLException e) {
839             e.printStackTrace();
840
841         }
842
843     }
844
845     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocation(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation)
846     {
847
848         String sql = "SELECT * FROM TaskList WHERE
849             priority = ? AND
850             scheduleName = ? AND
851             scheduleDate = ? AND
852             scheduleTime = ? AND
853             scheduleLocation = ?";
854
855         try {
856             PreparedStatement ps = connection.prepareStatement(sql);
857
858             ps.setInt(1, priority);
859             ps.setString(2, scheduleName);
860             ps.setString(3, scheduleDate);
861             ps.setString(4, scheduleTime);
862             ps.setString(5, scheduleLocation);
863
864             ResultSet rs = ps.executeQuery();
865
866             while(rs.next()){
867                 System.out.println("Task ID: " + rs.getInt("TaskId")
868                     + ", Schedule Name: " + rs.getString("scheduleName")
869                     + ", Schedule Date: " + rs.getString("scheduleDate")
870                     + ", Schedule Time: " + rs.getString("scheduleTime")
871                     + ", Schedule Location: " + rs.getString("scheduleLocation"));
872
873             }
874
875         } catch (SQLException e) {
876             e.printStackTrace();
877
878         }
879
880     }
881
882     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatus(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus)
883     {
884
885         String sql = "SELECT * FROM TaskList WHERE
886             priority = ? AND
887             scheduleName = ? AND
888             scheduleDate = ? AND
889             scheduleTime = ? AND
890             scheduleLocation = ? AND
891             scheduleStatus = ?";
892
893         try {
894             PreparedStatement ps = connection.prepareStatement(sql);
895
896             ps.setInt(1, priority);
897             ps.setString(2, scheduleName);
898             ps.setString(3, scheduleDate);
899             ps.setString(4, scheduleTime);
900             ps.setString(5, scheduleLocation);
901             ps.setString(6, scheduleStatus);
902
903             ResultSet rs = ps.executeQuery();
904
905             while(rs.next()){
906                 System.out.println("Task ID: " + rs.getInt("TaskId")
907                     + ", Schedule Name: " + rs.getString("scheduleName")
908                     + ", Schedule Date: " + rs.getString("scheduleDate")
909                     + ", Schedule Time: " + rs.getString("scheduleTime")
910                     + ", Schedule Location: " + rs.getString("scheduleLocation")
911                     + ", Schedule Status: " + rs.getString("scheduleStatus"));
912
913             }
914
915         } catch (SQLException e) {
916             e.printStackTrace();
917
918         }
919
920     }
921
922     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriority(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2)
923     {
924
925         String sql = "SELECT * FROM TaskList WHERE
926             priority = ? AND
927             scheduleName = ? AND
928             scheduleDate = ? AND
929             scheduleTime = ? AND
930             scheduleLocation = ? AND
931             scheduleStatus = ? AND
932             priority = ?";
933
934         try {
935             PreparedStatement ps = connection.prepareStatement(sql);
936
937             ps.setInt(1, priority);
938             ps.setString(2, scheduleName);
939             ps.setString(3, scheduleDate);
940             ps.setString(4, scheduleTime);
941             ps.setString(5, scheduleLocation);
942             ps.setString(6, scheduleStatus);
943             ps.setInt(7, priority2);
944
945             ResultSet rs = ps.executeQuery();
946
947             while(rs.next()){
948                 System.out.println("Task ID: " + rs.getInt("TaskId")
949                     + ", Schedule Name: " + rs.getString("scheduleName")
950                     + ", Schedule Date: " + rs.getString("scheduleDate")
951                     + ", Schedule Time: " + rs.getString("scheduleTime")
952                     + ", Schedule Location: " + rs.getString("scheduleLocation")
953                     + ", Schedule Status: " + rs.getString("scheduleStatus")
954                     + ", Priority: " + rs.getInt("priority"));
955
956             }
957
958         } catch (SQLException e) {
959             e.printStackTrace();
960
961         }
962
963     }
964
965     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3)
966     {
967
968         String sql = "SELECT * FROM TaskList WHERE
969             priority = ? AND
970             scheduleName = ? AND
971             scheduleDate = ? AND
972             scheduleTime = ? AND
973             scheduleLocation = ? AND
974             scheduleStatus = ? AND
975             priority = ? AND
976             priority = ?";
977
978         try {
979             PreparedStatement ps = connection.prepareStatement(sql);
980
981             ps.setInt(1, priority);
982             ps.setString(2, scheduleName);
983             ps.setString(3, scheduleDate);
984             ps.setString(4, scheduleTime);
985             ps.setString(5, scheduleLocation);
986             ps.setString(6, scheduleStatus);
987             ps.setInt(7, priority2);
988             ps.setInt(8, priority3);
989
990             ResultSet rs = ps.executeQuery();
991
992             while(rs.next()){
993                 System.out.println("Task ID: " + rs.getInt("TaskId")
994                     + ", Schedule Name: " + rs.getString("scheduleName")
995                     + ", Schedule Date: " + rs.getString("scheduleDate")
996                     + ", Schedule Time: " + rs.getString("scheduleTime")
997                     + ", Schedule Location: " + rs.getString("scheduleLocation")
998                     + ", Schedule Status: " + rs.getString("scheduleStatus")
999                     + ", Priority: " + rs.getInt("priority")
1000                    + ", Priority: " + rs.getInt("priority"));
1001
1002             }
1003
1004         } catch (SQLException e) {
1005             e.printStackTrace();
1006
1007         }
1008
1009     }
1010
1011     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4)
1012     {
1013
1014         String sql = "SELECT * FROM TaskList WHERE
1015             priority = ? AND
1016             scheduleName = ? AND
1017             scheduleDate = ? AND
1018             scheduleTime = ? AND
1019             scheduleLocation = ? AND
1020             scheduleStatus = ? AND
1021             priority = ? AND
1022             priority = ? AND
1023             priority = ?";
1024
1025         try {
1026             PreparedStatement ps = connection.prepareStatement(sql);
1027
1028             ps.setInt(1, priority);
1029             ps.setString(2, scheduleName);
1030             ps.setString(3, scheduleDate);
1031             ps.setString(4, scheduleTime);
1032             ps.setString(5, scheduleLocation);
1033             ps.setString(6, scheduleStatus);
1034             ps.setInt(7, priority2);
1035             ps.setInt(8, priority3);
1036             ps.setInt(9, priority4);
1037
1038             ResultSet rs = ps.executeQuery();
1039
1040             while(rs.next()){
1041                 System.out.println("Task ID: " + rs.getInt("TaskId")
1042                     + ", Schedule Name: " + rs.getString("scheduleName")
1043                     + ", Schedule Date: " + rs.getString("scheduleDate")
1044                     + ", Schedule Time: " + rs.getString("scheduleTime")
1045                     + ", Schedule Location: " + rs.getString("scheduleLocation")
1046                     + ", Schedule Status: " + rs.getString("scheduleStatus")
1047                     + ", Priority: " + rs.getInt("priority")
1048                     + ", Priority: " + rs.getInt("priority")
1049                     + ", Priority: " + rs.getInt("priority"));
1050
1051             }
1052
1053         } catch (SQLException e) {
1054             e.printStackTrace();
1055
1056         }
1057
1058     }
1059
1060     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3AndPriority4(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4, int priority5)
1061     {
1062
1063         String sql = "SELECT * FROM TaskList WHERE
1064             priority = ? AND
1065             scheduleName = ? AND
1066             scheduleDate = ? AND
1067             scheduleTime = ? AND
1068             scheduleLocation = ? AND
1069             scheduleStatus = ? AND
1070             priority = ? AND
1071             priority = ? AND
1072             priority = ? AND
1073             priority = ?";
1074
1075         try {
1076             PreparedStatement ps = connection.prepareStatement(sql);
1077
1078             ps.setInt(1, priority);
1079             ps.setString(2, scheduleName);
1080             ps.setString(3, scheduleDate);
1081             ps.setString(4, scheduleTime);
1082             ps.setString(5, scheduleLocation);
1083             ps.setString(6, scheduleStatus);
1084             ps.setInt(7, priority2);
1085             ps.setInt(8, priority3);
1086             ps.setInt(9, priority4);
1087             ps.setInt(10, priority5);
1088
1089             ResultSet rs = ps.executeQuery();
1090
1091             while(rs.next()){
1092                 System.out.println("Task ID: " + rs.getInt("TaskId")
1093                     + ", Schedule Name: " + rs.getString("scheduleName")
1094                     + ", Schedule Date: " + rs.getString("scheduleDate")
1095                     + ", Schedule Time: " + rs.getString("scheduleTime")
1096                     + ", Schedule Location: " + rs.getString("scheduleLocation")
1097                     + ", Schedule Status: " + rs.getString("scheduleStatus")
1098                     + ", Priority: " + rs.getInt("priority")
1099                     + ", Priority: " + rs.getInt("priority")
1100                     + ", Priority: " + rs.getInt("priority")
1101                     + ", Priority: " + rs.getInt("priority"));
1102
1103             }
1104
1105         } catch (SQLException e) {
1106             e.printStackTrace();
1107
1108         }
1109
1110     }
1111
1112     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3AndPriority4AndPriority5(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4, int priority5, int priority6)
1113     {
1114
1115         String sql = "SELECT * FROM TaskList WHERE
1116             priority = ? AND
1117             scheduleName = ? AND
1118             scheduleDate = ? AND
1119             scheduleTime = ? AND
1120             scheduleLocation = ? AND
1121             scheduleStatus = ? AND
1122             priority = ? AND
1123             priority = ? AND
1124             priority = ? AND
1125             priority = ? AND
1126             priority = ?";
1127
1128         try {
1129             PreparedStatement ps = connection.prepareStatement(sql);
1130
1131             ps.setInt(1, priority);
1132             ps.setString(2, scheduleName);
1133             ps.setString(3, scheduleDate);
1134             ps.setString(4, scheduleTime);
1135             ps.setString(5, scheduleLocation);
1136             ps.setString(6, scheduleStatus);
1137             ps.setInt(7, priority2);
1138             ps.setInt(8, priority3);
1139             ps.setInt(9, priority4);
1140             ps.setInt(10, priority5);
1141             ps.setInt(11, priority6);
1142
1143             ResultSet rs = ps.executeQuery();
1144
1145             while(rs.next()){
1146                 System.out.println("Task ID: " + rs.getInt("TaskId")
1147                     + ", Schedule Name: " + rs.getString("scheduleName")
1148                     + ", Schedule Date: " + rs.getString("scheduleDate")
1149                     + ", Schedule Time: " + rs.getString("scheduleTime")
1150                     + ", Schedule Location: " + rs.getString("scheduleLocation")
1151                     + ", Schedule Status: " + rs.getString("scheduleStatus")
1152                     + ", Priority: " + rs.getInt("priority")
1153                     + ", Priority: " + rs.getInt("priority")
1154                     + ", Priority: " + rs.getInt("priority")
1155                     + ", Priority: " + rs.getInt("priority")
1156                     + ", Priority: " + rs.getInt("priority"));
1157
1158             }
1159
1160         } catch (SQLException e) {
1161             e.printStackTrace();
1162
1163         }
1164
1165     }
1166
1167     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3AndPriority4AndPriority5AndPriority6(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4, int priority5, int priority6, int priority7)
1168     {
1169
1170         String sql = "SELECT * FROM TaskList WHERE
1171             priority = ? AND
1172             scheduleName = ? AND
1173             scheduleDate = ? AND
1174             scheduleTime = ? AND
1175             scheduleLocation = ? AND
1176             scheduleStatus = ? AND
1177             priority = ? AND
1178             priority = ? AND
1179             priority = ? AND
1180             priority = ? AND
1181             priority = ? AND
1182             priority = ?";
1183
1184         try {
1185             PreparedStatement ps = connection.prepareStatement(sql);
1186
1187             ps.setInt(1, priority);
1188             ps.setString(2, scheduleName);
1189             ps.setString(3, scheduleDate);
1190             ps.setString(4, scheduleTime);
1191             ps.setString(5, scheduleLocation);
1192             ps.setString(6, scheduleStatus);
1193             ps.setInt(7, priority2);
1194             ps.setInt(8, priority3);
1195             ps.setInt(9, priority4);
1196             ps.setInt(10, priority5);
1197             ps.setInt(11, priority6);
1198             ps.setInt(12, priority7);
1199
1200             ResultSet rs = ps.executeQuery();
1201
1202             while(rs.next()){
1203                 System.out.println("Task ID: " + rs.getInt("TaskId")
1204                     + ", Schedule Name: " + rs.getString("scheduleName")
1205                     + ", Schedule Date: " + rs.getString("scheduleDate")
1206                     + ", Schedule Time: " + rs.getString("scheduleTime")
1207                     + ", Schedule Location: " + rs.getString("scheduleLocation")
1208                     + ", Schedule Status: " + rs.getString("scheduleStatus")
1209                     + ", Priority: " + rs.getInt("priority")
1210                     + ", Priority: " + rs.getInt("priority")
1211                     + ", Priority: " + rs.getInt("priority")
1212                     + ", Priority: " + rs.getInt("priority")
1213                     + ", Priority: " + rs.getInt("priority")
1214                     + ", Priority: " + rs.getInt("priority"));
1215
1216             }
1217
1218         } catch (SQLException e) {
1219             e.printStackTrace();
1220
1221         }
1222
1223     }
1224
1225     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3AndPriority4AndPriority5AndPriority6AndPriority7(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4, int priority5, int priority6, int priority7, int priority8)
1226     {
1227
1228         String sql = "SELECT * FROM TaskList WHERE
1229             priority = ? AND
1230             scheduleName = ? AND
1231             scheduleDate = ? AND
1232             scheduleTime = ? AND
1233             scheduleLocation = ? AND
1234             scheduleStatus = ? AND
1235             priority = ? AND
1236             priority = ? AND
1237             priority = ? AND
1238             priority = ? AND
1239             priority = ? AND
1240             priority = ? AND
1241             priority = ?";
1242
1243         try {
1244             PreparedStatement ps = connection.prepareStatement(sql);
1245
1246             ps.setInt(1, priority);
1247             ps.setString(2, scheduleName);
1248             ps.setString(3, scheduleDate);
1249             ps.setString(4, scheduleTime);
1250             ps.setString(5, scheduleLocation);
1251             ps.setString(6, scheduleStatus);
1252             ps.setInt(7, priority2);
1253             ps.setInt(8, priority3);
1254             ps.setInt(9, priority4);
1255             ps.setInt(10, priority5);
1256             ps.setInt(11, priority6);
1257             ps.setInt(12, priority7);
1258             ps.setInt(13, priority8);
1259
1260             ResultSet rs = ps.executeQuery();
1261
1262             while(rs.next()){
1263                 System.out.println("Task ID: " + rs.getInt("TaskId")
1264                     + ", Schedule Name: " + rs.getString("scheduleName")
1265                     + ", Schedule Date: " + rs.getString("scheduleDate")
1266                     + ", Schedule Time: " + rs.getString("scheduleTime")
1267                     + ", Schedule Location: " + rs.getString("scheduleLocation")
1268                     + ", Schedule Status: " + rs.getString("scheduleStatus")
1269                     + ", Priority: " + rs.getInt("priority")
1270                     + ", Priority: " + rs.getInt("priority")
1271                     + ", Priority: " + rs.getInt("priority")
1272                     + ", Priority: " + rs.getInt("priority")
1273                     + ", Priority: " + rs.getInt("priority")
1274                     + ", Priority: " + rs.getInt("priority")
1275                     + ", Priority: " + rs.getInt("priority"));
1276
1277             }
1278
1279         } catch (SQLException e) {
1280             e.printStackTrace();
1281
1282         }
1283
1284     }
1285
1286     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3AndPriority4AndPriority5AndPriority6AndPriority7AndPriority8(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4, int priority5, int priority6, int priority7, int priority8, int priority9)
1287     {
1288
1289         String sql = "SELECT * FROM TaskList WHERE
1290             priority = ? AND
1291             scheduleName = ? AND
1292             scheduleDate = ? AND
1293             scheduleTime = ? AND
1294             scheduleLocation = ? AND
1295             scheduleStatus = ? AND
1296             priority = ? AND
1297             priority = ? AND
1298             priority = ? AND
1299             priority = ? AND
1300             priority = ? AND
1301             priority = ? AND
1302             priority = ? AND
1303             priority = ?";
1304
1305         try {
1306             PreparedStatement ps = connection.prepareStatement(sql);
1307
1308             ps.setInt(1, priority);
1309             ps.setString(2, scheduleName);
1310             ps.setString(3, scheduleDate);
1311             ps.setString(4, scheduleTime);
1312             ps.setString(5, scheduleLocation);
1313             ps.setString(6, scheduleStatus);
1314             ps.setInt(7, priority2);
1315             ps.setInt(8, priority3);
1316             ps.setInt(9, priority4);
1317             ps.setInt(10, priority5);
1318             ps.setInt(11, priority6);
1319             ps.setInt(12, priority7);
1320             ps.setInt(13, priority8);
1321             ps.setInt(14, priority9);
1322
1323             ResultSet rs = ps.executeQuery();
1324
1325             while(rs.next()){
1326                 System.out.println("Task ID: " + rs.getInt("TaskId")
1327                     + ", Schedule Name: " + rs.getString("scheduleName")
1328                     + ", Schedule Date: " + rs.getString("scheduleDate")
1329                     + ", Schedule Time: " + rs.getString("scheduleTime")
1330                     + ", Schedule Location: " + rs.getString("scheduleLocation")
1331                     + ", Schedule Status: " + rs.getString("scheduleStatus")
1332                     + ", Priority: " + rs.getInt("priority")
1333                     + ", Priority: " + rs.getInt("priority")
1334                     + ", Priority: " + rs.getInt("priority")
1335                     + ", Priority: " + rs.getInt("priority")
1336                     + ", Priority: " + rs.getInt("priority")
1337                     + ", Priority: " + rs.getInt("priority")
1338                     + ", Priority: " + rs.getInt("priority")
1339                     + ", Priority: " + rs.getInt("priority"));
1340
1341             }
1342
1343         } catch (SQLException e) {
1344             e.printStackTrace();
1345
1346         }
1347
1348     }
1349
1350     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3AndPriority4AndPriority5AndPriority6AndPriority7AndPriority8AndPriority9(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4, int priority5, int priority6, int priority7, int priority8, int priority9, int priority10)
1351     {
1352
1353         String sql = "SELECT * FROM TaskList WHERE
1354             priority = ? AND
1355             scheduleName = ? AND
1356             scheduleDate = ? AND
1357             scheduleTime = ? AND
1358             scheduleLocation = ? AND
1359             scheduleStatus = ? AND
1360             priority = ? AND
1361             priority = ? AND
1362             priority = ? AND
1363             priority = ? AND
1364             priority = ? AND
1365             priority = ? AND
1366             priority = ? AND
1367             priority = ? AND
1368             priority = ?";
1369
1370         try {
1371             PreparedStatement ps = connection.prepareStatement(sql);
1372
1373             ps.setInt(1, priority);
1374             ps.setString(2, scheduleName);
1375             ps.setString(3, scheduleDate);
1376             ps.setString(4, scheduleTime);
1377             ps.setString(5, scheduleLocation);
1378             ps.setString(6, scheduleStatus);
1379             ps.setInt(7, priority2);
1380             ps.setInt(8, priority3);
1381             ps.setInt(9, priority4);
1382             ps.setInt(10, priority5);
1383             ps.setInt(11, priority6);
1384             ps.setInt(12, priority7);
1385             ps.setInt(13, priority8);
1386             ps.setInt(14, priority9);
1387             ps.setInt(15, priority10);
1388
1389             ResultSet rs = ps.executeQuery();
1390
1391             while(rs.next()){
1392                 System.out.println("Task ID: " + rs.getInt("TaskId")
1393                     + ", Schedule Name: " + rs.getString("scheduleName")
1394                     + ", Schedule Date: " + rs.getString("scheduleDate")
1395                     + ", Schedule Time: " + rs.getString("scheduleTime")
1396                     + ", Schedule Location: " + rs.getString("scheduleLocation")
1397                     + ", Schedule Status: " + rs.getString("scheduleStatus")
1398                     + ", Priority: " + rs.getInt("priority")
1399                     + ", Priority: " + rs.getInt("priority")
1400                     + ", Priority: " + rs.getInt("priority")
1401                     + ", Priority: " + rs.getInt("priority")
1402                     + ", Priority: " + rs.getInt("priority")
1403                     + ", Priority: " + rs.getInt("priority")
1404                     + ", Priority: " + rs.getInt("priority")
1405                     + ", Priority: " + rs.getInt("priority")
1406                     + ", Priority: " + rs.getInt("priority"));
1407
1408             }
1409
1410         } catch (SQLException e) {
1411             e.printStackTrace();
1412
1413         }
1414
1415     }
1416
1417     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3AndPriority4AndPriority5AndPriority6AndPriority7AndPriority8AndPriority9AndPriority10(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4, int priority5, int priority6, int priority7, int priority8, int priority9, int priority10, int priority11)
1418     {
1419
1420         String sql = "SELECT * FROM TaskList WHERE
1421             priority = ? AND
1422             scheduleName = ? AND
1423             scheduleDate = ? AND
1424             scheduleTime = ? AND
1425             scheduleLocation = ? AND
1426             scheduleStatus = ? AND
1427             priority = ? AND
1428             priority = ? AND
1429             priority = ? AND
1430             priority = ? AND
1431             priority = ? AND
1432             priority = ? AND
1433             priority = ? AND
1434             priority = ? AND
1435             priority = ? AND
1436             priority = ?";
1437
1438         try {
1439             PreparedStatement ps = connection.prepareStatement(sql);
1440
1441             ps.setInt(1, priority);
1442             ps.setString(2, scheduleName);
1443             ps.setString(3, scheduleDate);
1444             ps.setString(4, scheduleTime);
1445             ps.setString(5, scheduleLocation);
1446             ps.setString(6, scheduleStatus);
1447             ps.setInt(7, priority2);
1448             ps.setInt(8, priority3);
1449             ps.setInt(9, priority4);
1450             ps.setInt(10, priority5);
1451             ps.setInt(11, priority6);
1452             ps.setInt(12, priority7);
1453             ps.setInt(13, priority8);
1454             ps.setInt(14, priority9);
1455             ps.setInt(15, priority10);
1456             ps.setInt(16, priority11);
1457
1458             ResultSet rs = ps.executeQuery();
1459
1460             while(rs.next()){
1461                 System.out.println("Task ID: " + rs.getInt("TaskId")
1462                     + ", Schedule Name: " + rs.getString("scheduleName")
1463                     + ", Schedule Date: " + rs.getString("scheduleDate")
1464                     + ", Schedule Time: " + rs.getString("scheduleTime")
1465                     + ", Schedule Location: " + rs.getString("scheduleLocation")
1466                     + ", Schedule Status: " + rs.getString("scheduleStatus")
1467                     + ", Priority: " + rs.getInt("priority")
1468                     + ", Priority: " + rs.getInt("priority")
1469                     + ", Priority: " + rs.getInt("priority")
1470                     + ", Priority: " + rs.getInt("priority")
1471                     + ", Priority: " + rs.getInt("priority")
1472                     + ", Priority: " + rs.getInt("priority")
1473                     + ", Priority: " + rs.getInt("priority")
1474                     + ", Priority: " + rs.getInt("priority")
1475                     + ", Priority: " + rs.getInt("priority")
1476                     + ", Priority: " + rs.getInt("priority"));
1477
1478             }
1479
1480         } catch (SQLException e) {
1481             e.printStackTrace();
1482
1483         }
1484
1485     }
1486
1487     public void printScheduleByPriorityAndNameAndDateAndTimeAndLocationAndStatusAndPriorityAndPriority2AndPriority3AndPriority4AndPriority5AndPriority6AndPriority7AndPriority8AndPriority9AndPriority10AndPriority11(int priority, String scheduleName, String scheduleDate, String scheduleTime, String scheduleLocation, String scheduleStatus, int priority2, int priority3, int priority4, int priority5, int priority6, int priority7, int priority8, int priority9, int priority10, int priority11, int priority12)
1488     {
1489
1490         String sql = "SELECT * FROM TaskList WHERE
1491             priority = ? AND
1492             scheduleName = ? AND
1493             scheduleDate = ? AND
1494             scheduleTime = ? AND
1495             scheduleLocation = ? AND
1496             scheduleStatus = ? AND
1497             priority = ? AND
1498             priority = ? AND
1499             priority = ? AND
1500             priority = ? AND
1501             priority = ? AND
1502             priority = ? AND
1503             priority = ? AND
1504             priority = ? AND
15
```

```

482                     if(mysqlExecute.mysqlUpdate("UPDATE TaskList SET manualScheduleName = '" + newScheduleName + "\", manualScheduleDate = '" + newScheduleDate + "\" WHERE userId = " + userId + " AND taskId = " + enteredTaskId) == -1){
483                         throw new SQLException();
484                     }
485                     System.out.println("Schedule information successfully changed.");
486                     return;
487                 }
488             }
489         } else if(removeMethodOption.equals("name")){
490             System.out.println("Please select from schedules above: (by its Name)\n(caption: cannot delete interest-based schedule by its Task ID)");
491             String enteredName = keyboard.nextLine();
492
493             System.out.println("Enter the new description for the new schedule: ");
494             newScheduleName = keyboard.nextLine();
495             //string is an object so value retains
496             System.out.println("Enter the new due date for the new schedule: ");
497             newScheduleDate = keyboard.nextLine();
498
499             ResultSet namesDB = mysqlExecute.mysqlQuery("SELECT manualScheduleName FROM TaskList WHERE userId = " + userId);
500             if(namesDB == null){
501                 throw new SQLException();
502             }
503
504             while(namesDB.next()){
505                 if(enteredName.equals(namesDB.getString(1))){
506                     //for checking same
507                     information update twice
508                     ResultSet

```

```

506 manualScheduleNameAndDate = mysqlExecute.
    mysqlQuery("SELECT manualScheduleName,
    manualScheduleDate FROM TaskList WHERE
    manualScheduleName = \"" + enteredName + "\");
507                     if (manualScheduleNameAndDate
508 == null) {
509                     throw new SQLException();
510                 }
510                 while (
511 manualScheduleNameAndDate.next()) {
511                     if (
512 manualScheduleNameAndDate.getString(1).equals(
513 newScheduleName) && manualScheduleNameAndDate.
514 getString(2).equals(newScheduleDate)) {
515                         System.out.println(""
516
517                         if (mysqlExecute.mysqlUpdate(""
518 UPDATE TaskList SET manualScheduleName = \"" +
519 newScheduleName + "\", manualScheduleDate = \"" +
520 newScheduleDate + "\" WHERE userId = " + userId +
521 " AND manualScheduleName = \"" + enteredName + "\"
522 ") == -1) {
523                     throw new SQLException();
524                 }
525             }
526             System.out.println("Schedule
527 information successfully changed.");
528         return;
529     System.out.println("Did not find a
530 matching schedule; please Try Again.");

```

```

530         //function should be terminated if there
      is a match or selection method is wrong (task id
      or name)
531     }
532
533     public void completeOrRemoveSchedule(int
      userId) throws SQLException, InterruptedException
    {
534         if(userId == 1){
535             System.out.println("You cannot modify/
      change anything about Administrator's account on
      the client version.");
536             return;
537         }
538         printTasks(userId,username); //give info
      about schedules
539
540         System.out.println("Do you want to remove
      a schedule by its Task ID or its Name?");
541         String removeMethodOption = keyboard.
      nextLine().replaceAll("\\s","");
      .toLowerCase(Locale.
      ROOT);
542         if(removeMethodOption.equalsIgnoreCase("taskid")){
543             System.out.println("Please select from
      schedules above: (by its Task ID)\n(caution:
      cannot delete interest-based schedule by its Task
      ID)");
544             String removingTaskId = keyboard.
      nextLine();
545             if(mysqlExecute.mysqlUpdate("DELETE
      FROM TaskList WHERE userId = " + userId + " AND
      taskId = " + removingTaskId + ";") == -1){
546                 throw new SQLException();
547             }
548         } else if(removeMethodOption.
     equalsIgnoreCase("name")){
549             System.out.println("Please select from
      schedules above: (by its Name)\n(caution: cannot
      delete interest-based schedule by its Task ID)");
550             String removingTaskId = keyboard.

```

```

550 nextLine();
551         if(mysqlExecute.mysqlUpdate("DELETE
552             FROM TaskList WHERE userId = " + userId + " AND
553             manualScheduleNam" +
554             "e = \"\" + removingTaskId + "
555             "\",") == -1){
556                 throw new SQLException();
557             }
558         } else {
559             System.out.println("Wrong option.");
560             return;
561         }
562         System.out.println("Task completed/deleted
563 .");
564
565         ResultSet streaksResultSet = mysqlExecute.
566         mysqlQuery("SELECT streak FROM UserList WHERE
567         userid = " + userId);
568         if(streaksResultSet == null){
569             throw new SQLException();
570         }
571         streaksResultSet.next();
572         int newStreaks = streaksResultSet.getInt(1
573 ) + 1;
574
575         ResultSet todayDailyStreakResultSet =
576         mysqlExecute.mysqlQuery("SELECT todayDailyStreak
577             FROM UserList WHERE userid = " + userId);
578         if(todayDailyStreakResultSet == null){
579             throw new SQLException();
580         }
581         todayDailyStreakResultSet.next();
582         int newTodayDailyStreak =
583         todayDailyStreakResultSet.getInt(1) + 1;
584
585         if(mysqlExecute.mysqlUpdate("UPDATE
586             UserList SET streak = " + newStreaks + " WHERE
587             userid = " + userId) == -1 ||
588             mysqlExecute.mysqlUpdate("UPDATE
589             UserList SET todayDailyStreak = " +
590             newTodayDailyStreak + " WHERE userId = " + userId

```

```

576 ) == -1)//update last time task was completed
577 {
578     throw new SQLException();
579 }
580
581     updateStreakLevel(newStreaks, userId);
582
583     System.out.println("\tOne more streak      "
584 );
585 }
586
587     public void updateStreakLevel(int newStreaks,
588         int userId) throws SQLException {
589         MysqlExecute mysqlExecute = new
590         MysqlExecute();
591         String newStreakLevel;
592         //optimize the code so that don't do the
593         //database operation if level is same; also set
594         //specific scope for levels
595         if(newStreaks <= 15){
596             newStreakLevel = "Newbie";
597         } else if(newStreaks <= 55){
598             newStreakLevel = "Apprentice";
599         } else if(newStreaks <= 125){
600             newStreakLevel = "Seasoned";
601         } else if(newStreaks <= 250){
602             newStreakLevel = "Cerebral";
603         } else if(newStreaks <= 550){
604             newStreakLevel = "Master";
605         } else{
606             newStreakLevel = "Legend";
607         }
608         if(mysqlExecute.mysqlUpdate("UPDATE
UserList SET streakLevel = '" + newStreakLevel +
"'" WHERE userId = " + userId) == -1){
609             throw new SQLException();
610         }
611     }
612
613     public void changeUsername(int userId) throws
614     NoSuchAlgorithmException, SQLException {

```

```

609         if(userId == 1){
610             System.out.println("You cannot modify/
611             change anything about Administrator's account on
612             the client version.");
613             return;
614         }
615         while(true) {
616             System.out.println("Please enter your
617             password for security check.");
618             StringBuilder saltString = new
619             StringBuilder(Sha256Hash.getSaltString(username));
620             final String passwordEntered =
621             Sha256Hash.doubleSHA256(keyboard.nextLine(),
622             saltString);
623
624             ResultSet password = mysqlExecute.
625             mysqlQuery("SELECT password FROM UserList WHERE
626             userId = " + userId);
627             if(password == null){
628                 throw new SQLException();
629             }
630             password.next();
631             if(!passwordEntered.equals(password.
632             getString(1))){
633                 System.out.println("Security check
634                 failed; please enter valid password.");
635                 continue;
636             }
637             System.out.println("Please enter your
638             new username:");
639             String newUsername = keyboard.nextLine
640             ();
641
642             System.out.println("Please confirm (re-
643             -type) your new username:");
644             String confirmNewUsername = keyboard.
645             nextLine();
646
647             if(newUsername.equals("") ||
648             confirmNewUsername.equals("")){
649                 System.out.println("Username

```

```

634 cannot be a Blank");
635             return;
636         }
637
638         ResultSet currentUsername =
639             mysqlExecute.mysqlQuery("SELECT username FROM
640             UserList WHERE userId = " + userId);
641         if(currentUsername == null){
642             throw new SQLException();
643         }
644         currentUsername.next();
645         if(newUsername.equals(currentUsername.
646             getString(1))){
647             System.out.println("Cannot change
648                 the username if the new username is equal to the
649                 current username.");
650             continue;
651         }
652         if (newUsername.equals(
653             confirmPassword)) {
654             if(mysqlExecute.mysqlUpdate(
655                 "UPDATE UserList SET username = '" + newUsername
656                 + "' WHERE userId = " + userId + ";") == -1){
657                 throw new SQLException();
658             }
659             System.out.println("Username
660                 successfully changed.");
661             return;
662         } else {
663             System.out.println("New Username
664                 does not match; please Try Again.");
665         }
666     }
667 }
668 public void changePassword(int userId) throws
669     NoSuchAlgorithmException, SQLException {
670     if(userId == 1){
671         System.out.println("You cannot modify/
672             change anything about Administrator's account on
673             the client version.");
674     }
675     return;

```

```

662         }
663         while (true) {
664             System.out.println("Please enter your
665             current password for security check.");
666             StringBuilder saltString = new
667             StringBuilder(Sha256Hash.getSaltString(username));
668             final String currentPasswordCheck =
669             Sha256Hash.doubleSHA256(keyboard.nextLine(),
670             saltString);
671             ResultSet currentPassword =
672             mysqlExecute.mysqlQuery("SELECT password FROM
673             UserList WHERE userId = " + userId);
674             if(currentPassword == null){
675                 throw new SQLException();
676             }
677             currentPassword.next();
678             if(!currentPasswordCheck.equals(
679             currentPassword.getString(1))){
680                 System.out.println("Security check
681                 failed; please enter valid password.");
682                 return;
683             }
684             System.out.println("Please enter your
685             new password:");
686             StringBuilder newSaltString =
687             Sha256Hash.generateSaltString();
688             final String newPassword = Sha256Hash.
689             doubleSHA256(keyboard.nextLine(), newSaltString);
690             System.out.println("Please confirm (
691             retype) your new password:");
692             final String confirmPassword =
693             Sha256Hash.doubleSHA256(keyboard.nextLine(),
694             newSaltString);
695             if(newPassword.equals(currentPassword.
696             getString(1))){
697                 System.out.println("Cannot change
698                 the password if the new password is equal to the
699                 current password.");
700                 continue;
701             }
702             if (newPassword.equals(

```

```

685 confirmPassword)) {
686             System.out.println("Please enter
687             your new password hint:");
688             String passwordHint = keyboard.
689             nextLine();
690             if(mysqlExecute.mysqlUpdate(
691                 "UPDATE UserList SET password = '" + newPassword
692                 + "' , salt = '" + newSaltString + "' ,
693                 passwordHint = '" + passwordHint + "' WHERE
694                 userId = " + userId + ";") == -1){
695                 throw new SQLException();
696             }
697             System.out.println("Password
698             successfully changed.");
699             return;
700         } else {
701             System.out.println("New Password
702             does not match; Please Try Again.");
703         }
704     }
705     public synchronized void printTasks(int userId
706             , String username) throws SQLException,
707             InterruptedException {
708         if (userId == 1) {
709             System.out.println("You cannot modify/
710             change anything about Administrator's account on
711             the client version.");
712             return;
713         }
714         ArrayList<Integer> interestScheduleIds =
715             new ArrayList<>();
716         ResultSet scheduleList = mysqlExecute.
717             mysqlQuery("SELECT taskId, manualScheduleName,
718             manualScheduleDate, interestScheduleId FROM
719             TaskList WHERE userId = " + userId);
720         if (scheduleList == null) {
721             throw new SQLException();
722         }
723         scheduleList.next();

```

```

710         while (scheduleList.getString(4) != null){
711             interestScheduleIds.add(scheduleList.
712             getInt(4));
713             scheduleList.next();
714             if(scheduleList.isAfterLast()) {
715                 break;
716             }
717             scheduleList.previous();
718
719
720             System.out.println("Schedules added by "
721             + username + ":" );
722             while(scheduleList.next()){
723                 System.out.println("\tTask ID " +
724                 scheduleList.getString(1) + ":" +
725                 scheduleList.getString(2) + " in " +
726                 scheduleList.getString(3));
727             }
728
729             System.out.println("Schedules added based
730             on " + username + "'s interest:");
731             Thread printInterestSchedulesThread = new
732             Thread(()-> { try { printInterestSchedules(
733                 interestScheduleIds); } catch (SQLException
734                 exception) { exception.printStackTrace(); } });
735             printInterestSchedulesThread.start();
736             printInterestSchedulesThread.join();
737         }
738
739         private void printInterestSchedules(ArrayList<
740             Integer> eventIds) throws SQLException {
741             for(int i = 0; i<eventIds.size(); i++) {
742                 ResultSet interestSchedule =
743                     mysqlExecute.mysqlQuery("SELECT eventName,
744                     eventDate FROM InterestSchedule WHERE eventId = "
745                     + eventIds.get(i) + ";");
746                 if (interestSchedule == null) {
747                     throw new SQLException();
748                 }
749                 interestSchedule.next();
750             }
751         }
752     }
753 }
```

```
738         System.out.println("\t" + (i+1) + ". "
+ interestSchedule.getString(1) + " in "
interestSchedule.getString(2));
739     }
740 }
741 }
```