Relational Databases with MySQL Week 6 Coding Assignment Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document to the repository. Additionally, push an .sql file with all your queries and your Java project code to the same repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

This week you will be working together as a **team** to create a full CRUD application.

Your console CRUD application will need to use a database to store all the application data.

As a team, decide what you want your project to do. Get instructor approval early in the week before beginning development.

You need to have at least 3 entities.

Users should be able to interact via the console (i.e. Scanner(System.in)))

Everyone must individually submit their own assignment documents and the full code for the entire project. Inside the code, use comments to make it clear which code you specifically wrote.

Although git provides collaboration functionality, you are not required to use it to collaborate back and forth with your teammates. You can use any method you decide on as a team.

Everyone will be graded on their individual contributions.

Goaltivity	
Project Team Members:	
Myles	

Myles

Heather

Project Name:

My Contribution to the Project:

I did the journal and journal tags operations mainly but Myles and I both zoomed to help each other work through problems in the user and part of the menu. My complete contributions can be seen by the github repo https://github.com/Goaltivity/Goaltivity but this is the main group of code I worked on.

```
🖸 Menujava 🔃 UserDAO.java 🔑 JournalTagsDA... 🕘 TagDAO.java 🚇 JournalTags.j... 🕘 Userjava 🚇 Tag.java 🚇 Application.j... 🚇 Journal.java 🔃 *JournalDAO.j... 🗵 🚇 DBConnecti
   package dao;
  3⊝ import java.sql.Connection;
 4 import java.sql.PreparedStatement:
 5 import java.sql.ResultSet;
 6 import java.sql.SOLException:
   import java.util.ArrayList;
 8 import java.util.Date;
 9 import java.util.List;
10 import entity. Journal:
       11 public class JournalDAO {
13
14
15
16
18
19
20°
21
22°
23
24
25
26
27
28
29
30
31
        public ListJournal> getJournalEntriesByJournalTag(int tagId) throws SQLException {
   PreparedStatement ps = connection.prepareStatement(GET_JOURNALENTRIES_BY_TAGID_QUERY);
             ps.setInt(1, tagId);
             ResultSet rs = ps.executeQuery();
List<Journal> journalsByTag = new ArrayList<Journal>();
             while (rs.next()) {
                 journalsByTag
                          .add(populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
             return journalsByTag;
33e
34
35
        public JournalDAO() {
             connection = DBConnection.getConnection();
        public List<Journal> getJournals() throws SQLException {
   ResultSet rs = connection.prepareStatement(GET_JOURNALS_QUERY).executeQuery();
   List<Journal> journals = new ArrayList<Journal>();
37
38
🔑 Menujava 🕑 UserDAOjava 🔑 JournalTagsDA... 🕑 TagDAOjava 🕑 JournalTagsj... 🕑 Userjava 😥 Tagjava 🔑 Applicationj... 😢 JournalTjava
 31
32
               return journalsByTag;
33⊜
          public JournalDAO() {
 34
              connection = DBConnection.getConnection();
          public List<Journal> getJournals() throws SQLException {
   ResultSet rs = connection.prepareStatement(GET_JOURNALS_QUERY).executeQuery();
   List<Journal> journals = new ArrayList<Journal>();
 36⊜
37
 38
 39
               while (rs.next()) {
 40
                   journals.add(populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
 41
 42
               return journals;
 43
 449
          public Journal getJournalbyId(int id) throws SQLException {
              PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_BY_ID_QUERY);
 45
 46
               ps.setInt(1, id);
 47
               ResultSet rs = ps.executeQuery();
 48
               rs.next():
 49
               return populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5));
 50
          public int readGetJournalId(String title) throws SQLException {
 51⊖
 52
              PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_ID_QUERY);
ps.setString(1, title);
 53
54
 55
56
57
               ResultSet rs = ps.executeQuery();
              rs.next();
               journalId = rs.getInt(1);
 58
59
               return journalId;
 60e
          public void createNewJournal(String entryName, String content, int journalTagId, int id) throws SQLException {
              PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNAL_QUERY); ps.setString(1, entryName); ps.setString(2, content);
 61
62
 63
 64
              ps.setInt(3, id);
ps.executeUpdate();
 65
 66
               journalTagsDAO.createNewJournalTag(readGetJournalId(entryName), journalTagId);
 67
          public void deleteJournalById(int id) throws SQLException {
 68€
```

```
🔑 Menujava 🕑 UserDAOjava 🛂 JournalTagsDA... 🕑 TagDAOjava 🕑 JournalTagsj... 🗘 Userjava 🗗 Tagjava 🛂 Applicationj... 🗘 Journaljava
              ps.setInt(1, id);
 47
              ResultSet rs = ps.executeQuery();
 48
              rs.next();
 49
              return populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5));
 50
51<sup>©</sup>
52
53
54
55
56
57
         public int readGetJournalId(String title) throws SQLException {
             PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_ID_QUERY);
              ps.setString(1, title);
              ResultSet rs = ps.executeQuery();
             rs.next();
journalId = rs.getInt(1);
 58
59
              return journalId;
 60e
         public void createNewJournal(String entryName, String content, int journalTagId, int id) throws SQLException {
 61
62
63
             PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNAL_QUERY);
ps.setString(1, entryName);
ps.setString(2, content);
 64
65
66
             ps.setInt(3, id);
ps.executeUpdate();
              journalTagsDAO.createNewJournalTag(readGetJournalId(entryName), journalTagId);
 67
68⊜
         public void deleteJournalById(int id) throws SQLException {
 69
              journalTagsDAO.deleteJournalTagByJournalId(id);
             PreparedStatement ps = connection.prepareStatement(DELETE_JOURNAL_BY_ID_QUERY);
 70
71
             ps.setInt(1, id);
 72
73
74⊕
             ps.executeUpdate();
         public void updateJournalById(int id, String content) throws SQLException {
             PreparedStatement ps = connection.prepareStatement(UPDATE_JOURNAL_BY_ID_QUERY);
ps.setInt(2, id);
ps.setString(1, content);
 75
76
77
78
79
             ps.executeUpdate();
 80⊝
         private Journal populateJournal(int id, Date date, String title, String content, int userId) throws SQLException {
 81
82
             return new Journal(id, date, title, content, userId);
         }
 83 }
```

```
UserDAO.java
JournalTagsDA...
TagDAO.java
JournalTags.j...
                                                                                         🗓 Tag.java 🔑 ,
                                                                            User.java
 1 package entity;
 3 import java.util.ArrayList;
 4 import java.util.Date;
 5 import java.util.List;
6 public class Journal {
       private int id;
       private Date date;
 9
       private String title;
10
       private String content;
11
       private int user;
12
       private List<Tag> tags;
13
14⊝
       public Journal(int id, Date date, String title, String content) {
15
           this.setUser(-1);
16
           this.id = id;
17
           this.setDate(date);
18
           this.title = title;
19
           this.content = content;
20
           this.tags = new ArrayList<Tag>();
21
       }
23⊝
       public Journal(int id2, Date date2, String title2, String content2, int userById) {
24
           this.setUser(-1);
25
           this.id = id2;
26
           this.setDate(date2);
27
           this.title = title2;
28
           this.content = content2;
29
           this.user = userById;
30
       }
       public int getId() {
31⊖
32
           return id;
33
34⊝
       public void setId(int id) {
35
           this.id = id;
36
37⊝
       public String getTitle() {
```

```
[[] Menu.java U UserDAO.java & JournalTagsDA... U TagDAO.java U JournalTags.j... U User.java U Tag
  33
  34⊜
         public void setId(int id) {
  35
             this.id = id;
  36
  37⊝
         public String getTitle() {
  38
              return title;
  39
         }
  40⊝
         public void setTitle(String title) {
  41
             this.title = title;
  42
  43⊝
         public String getContent() {
  44
             return content;
  45
         }
  46⊜
         public void setContent(String content) {
  47
             this.content = content;
  48
  49⊝
         public Date getDate() {
  50
             return date;
  51
         public void setDate(Date date) {
  52⊜
  53
             this.date = date;
  54
  55⊝
         public int getUser() {
             return user;
  56
  57
  58⊝
         public void setUser(int user) {
  59
             this.user = user;
  60
  61⊜
         public List<Tag> getTags() {
  62
              return tags;
  63
         }
  64
  65⊝
         public void addTag(Tag tag) {
  66
             if (tag != null) {
  67
                  tags.add(tag);
  68
             }
  69
  70 }
```

```
UserDAO.java
                             Journal Tags DAO. java
                                                   TagDAO.java
🕖 Menu.java
 1 package entity;
 2
 30 import java.util.ArrayList;
 4 import java.util.List;
 5 public class JournalTags {
        private int journalId;
 7
        private int tagId;
        private List<Tag> tags;
 8
 9⊝
        public JournalTags(int journalId, int tagId) {
            this.journalId = journalId;
10
11
            this.tagId = tagId;
12
            this.tags = new ArrayList<Tag>();
13
14⊝
        public int getJournalId() {
15
            return journalId;
16
17⊝
        public void setJournalId(int journalId) {
18
            this.journalId = journalId;
19
20⊝
        public int getTagId() {
21
            return tagId;
22
23⊝
        public void setTagId(int tagId) {
24
            this.tagId = tagId;
25
26⊜
        public List<Tag> getTags() {
27
            return tags;
28
29⊝
        public void setTags(List<Tag> tags) {
30
            this.tags = tags;
31
        }
32 }
```

```
🔑 Menujava 🛮 UserDAOjava 📳 JournalTagsDAOjava 🗵 TagDAOjava 🗗 JournalTagsjava 🗗 Userjava 🖫 Tagjava 🖟 Applicationjava 🗗 JournalTags
   1 package dao;
   3 import java.sql.Connection;
  4 import java.sql.PreparedStatement;
  5 import java.sql.ResultSet;
  6 import java.sql.SQLException;
  7 import java.util.ArrayList;
  8 import java.util.List;
  9 import entity.JournalTags;
 10 public class JournalTagsDAO {
           lic class JournallagsDAO {
    private Connection connection;
    private final String GET_JOURNALTAGS_QUERY = "SELECT * FROM tags";
    private final String GET_JOURNALTAG_BY_JOURNALID_QUERY = "SELECT * FROM journal_tags WHERE id = ?";
    private final String GET_JOURNALTAG_BY_TAGID_QUERY = "SELECT * FROM journal_tags WHERE journal_tagsId = ?";
    private final String CREATE_NEW_JOURNALTAG_QUERY = "INSERT INTO journal_tags(journal, tag) VALUES (?,?)";
    private final String DELETE_JOURNALTAG_BY_ID_QUERY = "DELETE FROM journal_tags where id = ?";
    private final String DELETE_JOURNALTAG_BY_JOURNALID_QUERY = "DELETE FROM journal_tags where journal = ?";
 11
 12
 13
 15
 16
 18
 19⊜
            public JournalTagsDAO() {
                  connection = DBConnection.getConnection();
 20
 21
 22⊝
            public List<JournalTags> getJournalTags() throws SQLException {
                   ResultSet rs = connection.prepareStatement(GET_JOURNALTAGS_QUERY).executeQuery();
 23
                   List<JournalTags> journalTags = new ArrayList<JournalTags>();
 24
 25
                   while (rs.next()) {
 26
                        journalTags.add(populateJournalTags(rs.getInt(1), rs.getInt(2)));
 27
 28
                   return journalTags;
 29
            public JournalTags getJournalTagByJournalId(int journalId) throws SQLException {
    PreparedStatement ps = connection.prepareStatement(GET_JOURNALTAG_BY_JOURNALID_QUERY);
 30⊝
 31
 32
                   ps.setInt(1, journalId);
 33
                   ResultSet rs = ps.executeQuery();
 34
                   rs.next();
 35
                   return populateJournalTags(rs.getInt(1), rs.getInt(2));
 36
            public void createNewJournalTag(int journalId, int tagId) throws SQLException {
    PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNALTAG_QUERY);
 37⊝
 38
                  ps.setInt(1, journalId);
ps.setInt(2, tagId);
 39
 40
 41
                   ps.executeUpdate();
 42
            private JournalTags populateJournalTags(int journalTagId, int tagId) {
 430
```

```
Menu.java
16
17
       private final String DELETE_JOURNALTAG_BY_JOURNALID_QUERY = "DELETE FROM journal_tags where journal = ?";
18
19⊜
       public JournalTagsDAO() {
20
           connection = DBConnection.getConnection();
21
22⊝
       public List<JournalTags> getJournalTags() throws SQLException {
           ResultSet rs = connection.prepareStatement(GET_JOURNALTAGS_QUERY).executeQuery();
23
24
           List<JournalTags> journalTags = new ArrayList<JournalTags>();
25
           while (rs.next()) {
26
               journalTags.add(populateJournalTags(rs.getInt(1), rs.getInt(2)));
27
28
           return journalTags;
29
       public JournalTags getJournalTagByJournalId(int journalId) throws SQLException {
300
           PreparedStatement ps = connection.prepareStatement(GET_JOURNALTAG_BY_JOURNALID_QUERY);
31
           ps.setInt(1, journalId);
32
33
           ResultSet rs = ps.executeQuery();
34
           rs.next();
35
           return populateJournalTags(rs.getInt(1), rs.getInt(2));
36
370
       public void createNewJournalTag(int journalId, int tagId) throws SQLException {
38
           PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNALTAG_QUERY);
39
           ps.setInt(1, journalId);
40
           ps.setInt(2, tagId);
41
           ps.executeUpdate();
42
43⊖
       private JournalTags populateJournalTags(int journalTagId, int tagId) {
44
           return new JournalTags(journalTagId, tagId);
45
46⊜
       public void deleteJournalTag(int id) throws SQLException {
47
           PreparedStatement ps = connection.prepareStatement(DELETE_JOURNALTAG_BY_ID_QUERY);
48
           ps.setInt(1, id);
49
           ps.executeUpdate();
50
51⊝
       public void deleteJournalTagByJournalId(int journal) throws SQLException {
52
           PreparedStatement ps = connection.prepareStatement(DELETE_JOURNALTAG_BY_JOURNALID_QUERY);
53
           ps.setInt(1, journal);
54
           ps.executeUpdate();
55
56 }
57
58
```

Screenshots of Code:

```
🔑 Menujava 🗵 🛭 UserDAOjava 🔑 JournalTagsDAOjava 🕑 TagDAOjava 🕑 JournalTagsjava 🕑 Userjava 🚇 Tagjava 🚇 Applicationjava 🕑 JournalTags
   1 package application;
   3⊖ import java.sql.SQLException;
   4 import java.util.Arrays;
   5 import java.util.List;
   6 import java.util.Scanner;
   7 import dao.JournalDAO;
   8 import dao.JournalTagsDAO;
   9 import dao.TagDAO;
  10 import dao.UserDAO;
  11 import entity. Journal;
  12 import entity.JournalTags;
  13 import entity. Tag;
  14 import entity.User;
  15 public class Menu {
         private JournalDAO journalDao = new JournalDAO();
  16
         private JournalTagsDAO journalTagsDAO = new JournalTagsDAO();
private TagDAO tagDAO = new TagDAO();
  18
         private UserDAO userDAO = new UserDAO();
private Scanner scanner = new Scanner(System.in);
  20
         private List<String> loginOptions = Arrays.asList("Display all Users", "Select a User", "Create a User",
  21⊝
  private List<String> userOptions = Arrays.asList("Create a Journal Entry", "Display all Journal Entries",
    "Update a Journal Entry", "Delete a Journal Entry", "View journal tag options"
  24⊝
 25
  26
  27 //When they ask to update a journal entry
<u>3</u> 28⊝
      private List<String> journalUpdateOptions = Arrays.asList("Update a Journal Entry by Title",
 29
  29 "Update a Journal Entry by Content");
30 //When "See all possible Journal Tags" is selected
        private List<String> tagOptions = Arrays.asList("Display all tags",
  31⊖
                // "Search for tag via name",

"Search for Journal Entries via tag", "Create new tag", "Update tag", "Delete tag");
  33
  340
        public void start() {
             String selection = "";
  35
             do {
  37
                 printLogInMenu();
  38
                  selection = scanner.nextLine();
  39
                  trv {
 🔊 Menu.java 🛭 🗗 UserDAO.java 🔝 JournalTagsDAO.java 🕒 TagDAO.java 🕩 JournalTags.java 🕩 User.java 🕩 Tag.java 🔑 Application.java 🕩 Journal.java
          public void start() {
  34⊝
               String selection = "";
   35
   36
               do {
   37
                   printLogInMenu();
                    selection = scanner.nextLine();
   38
   39
   40
                        if (selection.equals("1")) {
  41
                            displayAllUsers();
   42
                        } else if (selection.equals("2")) {
  43
                            selectUser();
  44
                        } else if (selection.equals("3")) {
                            createUser();
  45
  46
                        } else if (selection.equals("4")) {
  17
                            deleteUser();
  48
                   } catch (SQLException e) {
   49
   50
                        e.printStackTrace();
   51
   52
                   System.out.println("Press enter to continue.");
   53
                    scanner.nextLine():
   54
               } while (!selection.equals("-1"));
   55
   56
   57⊜
           private void printLogInMenu() {
   58
               System.out.println("Select an Option: \n ---
               for (int i = 0; i < loginOptions.size(); i++) {</pre>
   59
   60
                   System.out.println(i + 1 + ") " + loginOptions.get(i));
   61
   62
   63⊜
           private void displayAllUsers() throws SQLException {
               List<User> users = userDAO.getUsers();
   65
               for (User user : users) {
   66
                    System.out.println(user.getUserId() + ": " + user.getFirstname() + " " + user.getLastname());
   67
  68
   69⊜
           private void selectUser() throws SQLException {
   70
               displayAllUsers();
               System.out.println("Enter your user ID");
   71
   72
               int id = Integer narseInt(scanner nextline()):
```

```
🚇 Menujava 🛭 🗓 UserDAOjava 💹 JournalTagsDAOjava 🔃 TagDAOjava 🔃 JournalTagsjava 🗓 Userjava 🗓 Tagjava 🔎 Applicationjava 🔃 Journaljava
               User firstname = userDAO.getUserById(id);
               System.out.println("-----");
System.out.println(" Welcome " + firstname.getFirstname());
  74
  75
  76
               System.out.println("-----
  77
               userOptionsMenu();
  78
  79⊝
          private void userOptionsMenu() {
               String selection = "";
String subselection = "";
  80
  81
  82
               do {
  83
                    printUserOptionsMenu();
  84
                    scanner = new Scanner(System.in);
  85
                    selection = scanner.nextLine();
  86
                    try {
                        if (selection.equals("1")) {
   System.out.println("What would you like to name your new journal entry \n");
  87
  88
                            String entryName = scanner.nextLine();
  89
  90
                            System.out.println("Enter journal content: ");
  91
                             String content = scanner.nextLine();
  92
                             System.out.println("Please confirm your identity by entering your user id.");
  93
                             displayAllUsers();
  94
                             int userId = Integer.parseInt(scanner.nextLine());
  95
                             tagDAO.displayAllTags();
  96
                            System.out.println("Add a tag id for this post: ");
int journalTagId = Integer.parseInt(scanner.nextLine());
  97
  98
  99
                             journalDao.createNewJournal(entryName, content, journalTagId, userId);
 100
                            userOptionsMenu();
                        } else if (selection.equals("2")) {
    System.out.println("Journal entries \n");
 102
                            displayAllEntries();
System.out.println("Press enter to continue.");
 103
 104
 105
                            scanner.nextLine();
 106
                            userOptionsMenu();
 107
                        } else if (selection.equals("3")) {
 108
                            displayAllEntries();
 109
                             System.out.println("Which journal entry would you like to update?");
 110
                             int id = Integer.parseInt(scanner.nextLine());
                             System out nrintln("Enter the new content:
🔑 Menu,java 🗵 🗓 UserDAO,java 🔑 JournalTagsDAO,java 🕑 TagDAO,java 🕩 JournalTags,java 🕑 User,java 🔑 Tag,java 🔑 Application,java 🕑 Journal,java
                            displayAllEntries();
System.out.println("Press enter to continue.");
 104
                            scanner.nextLine();
 105
 106
                            userOptionsMenu();
                        } else if (selection.equals("3")) {
 107
                            displayAllEntries();
System.out.println("Which journal entry would you like to update?");
 108
                            int id = Integer.parseInt(scanner.nextLine());
 110
                            System.out.println("Enter the new content:
                            String newContent = scanner.nextLine();
                            journalDao.updateJournalById(id, newContent);
                        } else if (selection.equals("4")) {
                            displayAllEntries();
System.out.println("Enter the id of the entry you would like to delete: ");
                            int id = Integer.parseInt(scanner.nextLine());
                            journalDao.deleteJournalById(id);
 118
                        } else if (selection.equals("4")) {
 119
                            System.out.println("Which entry would you like to delete? \n");
 120
                            int idToDelete = Integer.parseInt(scanner.nextLine());
 122
                            journalDao.deleteJournalById(idToDelete);
 123
                            printTagOptionsMenu();
                        } else if (selection.equals("5")) {
   System.out.println("Journal Tags \n");
 124
                            do {
                                printTagOptionsMenu();
 128
                                 scanner = new Scanner(System.in);
                                 subselection = scanner.nextLine();
                                 try {
                                     if (subselection.equals("1")) {
                                         tagDAO.displayAllTags();
                                     } else if (subselection.equals("2")) {
                                         tagDAO.displayAllTags();
 134
                                          System.out.println(
                                                   "Enter the id of the tag which you would like to view the journal entries from: ");
 136
                                          int tagId = Integer.parseInt(scanner.nextLine());
                                          displayAllJournalsByTag(journalDao.getJournalEntriesByJournalTag(tagId));
 138
 139
                                     } else if (subselection.equals("3")) {
                                         System.out.println("Enter new tag name: ");
String newTog = scannen newTine();
 140
```

```
🔑 Menujava 🗵 🗓 UserDAO.java 🔑 JournalTagsDAO.java 🕑 TagDAO.java 🗘 JournalTags.java 🗘 User.java 🗘 Tag.java 🔑 Application.java 🗘 Journal
                                           } else if (subselection.equals("3")) {
 139
                                                 System.out.println("Enter new tag name: ");
 140
 141
                                                String newTag = scanner.nextLine();
 142
                                                tagDAO.createNewTag(newTag);
                                           } else if (subselection.equals("4")) {
   System.out.println("Enter the id of the tag you would like to update: ");
 143
 144
                                                 int idOfTagToUpdate = Integer.parseInt(scanner.nextLine());
 145
                                                 System.out.println("Enter the new tag: ");
 146
                                                String updatedTag = scanner.nextLine();
 147
 148
                                                tagDAO.updateTagById(idOfTagToUpdate, updatedTag);
                                           } else if (subselection.equals("5")) {
 149
 150
                                                System.out.println("Enter the id of the tag you would like to remove: ");
 151
                                                 int idOfTagToRemove = Integer.parseInt(scanner.nextLine());
                                                tagDAO.deleteTag(idOfTagToRemove);
 153
                                            } else if (!(subselection.equals("-1"))) {
 154
                                                System.out.println("Invalid Option");
 155
 156
                                      } catch (SQLException e) {
                                          e.printStackTrace();
 157
 158
 159
                                } while (!(subselection.equals("-1")));
 160
                      } catch (SQLException e) {
 161
                           e.printStackTrace();
 163
 164
                 } while (!selection.equals("-1"));
 165
 166€
            private void displayAllJournalsByTag(List<Journal> journals) {
 167
                 for (Journal journal : journals) {
                      System.out.println(journal.getId() + ": " + journal.getDate() + " " + journal.getTitle() + "\n\t"
 168
                                + journal.getContent() + "\n\t");
 169
 170
                 }
 171
 172⊖
            private void displayAllEntries() throws SQLException {
                 List<Journal> journals = journalDao.getJournals();
 173
                 for (Journal journal : journals) {
 174
                      System.out.println(journal.getId() + ": " + journal.getDate() + " " + journal.getTitle() + "\n\t"
 175
                              + journal.getContent() + "\t");
 176
                                                                                                                 1 171/11
178
179
180
            }
 181
        private void createUser() throws SQLException {
   System.out.println("Enter your first name: ");
   String firstname = scanner.nextLine();
 182<sup>e</sup>
183
 184
            String firstname = scanner.nextLine();
System.out.println("Enter your last name: ");
String lastname = scanner.nextLine();
System.out.println("Enter your email address: ");
String emailaddress = scanner.nextLine();
userDAO.createNewUser(firstname, lastname, emailaddress);
 185
186
187
 188
 189
190
191<sup>®</sup>
         private void deleteUser() throws SQLException {
            System.out.println(
"Please delete all journal entries before attempting to delete a user. Enter -1 to go back if needed. \n Enter user ID for the account you want to remove displayAllUsers();

One of the account you want to remove displayAllUsers();
 192
193
194
195
196
197
198<sup>6</sup>
199
200
            int id = Integer.parseInt(scanner.nextLine());
userDAO.deleteUser(id);
        201
202
            }
 203
         fprivate void printTagOptionsMenu() {
   System.out.println("Select an Option: \n----
   for (int i = 0; : < tagOptions.size(); i++) {
    System.out.println(i + 1 + ") " + tagOptions.get(i));
}</pre>
 2048
 205
 207
 208
 210
 211 }
```

Application

```
Menu.java
UserDAO.java
                           Journal Tags DAO. java
Jag DAO. java
                                                               J
1 package application;
 2
3 3 import java.util.List;
4 import dao.TagDAO;
5 import entity.Tag;
 6 public class Application {
 7
       public static void main(String[] args) {
 8⊝
           Menu menu = new Menu();
 9
           menu.start();
10
11
       }
12
13 }
14
```

(Dao) DBConnection

```
Menu.java
             UserDAO.java
                            JournalTagsDA...
                                               ☑ TagDAO.java
                                                               JournalTags.java
                                                                                User.java
                                                                                            ☑ Tag.java
 1 package dao;
 3 mport java.sql.Connection; ...
 7 public class DBConnection {
 8
        private final static String URL = "jdbc:mysql://localhost:3306/goaltivity";
 9
        private final static String USERNAME = "root";
10
        private final static String PASSWORD = "password";
12
        private static Connection connection;
13
        private static DBConnection instance;
14
15⊜
        private DBConnection(Connection connection) {
            this.connection = connection;
b16
17
18
19⊜
        public static Connection getConnection() {
20
            if (instance == null) {
21
                try {
22
                    connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
23
                    instance = new DBConnection(connection);
24
                    System.out.println("Connection is successful");
25
                } catch (SQLException e) {
26
                    e.printStackTrace();
27
                }
28
30
            return DBConnection.connection;
31
32 }
33
```

JournalDAO

```
🗓 Menujava 🖟 UserDAOjava 🖟 JournalTagsDA... 🕒 TagDAOjava 🖟 JournalTags,java 🖟 Userjava 🖟 Tagjava 🖟 Application,java 🖟 JournalTagsDA... 🗘 TagDAOjava 🗵 🖟 DBConnection,java
       package dao;
   30 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
   6 import java.sql.SQLException;
7 import java.util.ArrayList;
8 import java.util.Date;
8 import java.util.Date;
9 import java.util.List;
10 import entity.Journal;
11 public class JournalDAO {
12    private Connection connection;
13    private JournalTagsDAO journalTagsDAO = new JournalTagsDAO();
14    private journalTagsDAO journalTagsDAO = new JournalTagsDAO();
15    private final String GET_JOURNALS_QUERY = "SELECT * FROM journal WHERE id = ?";
16    private final String GET_JOURNAL_BY_ID_QUERY = "INSERT INTO journal (title, content, user) VALUES (?,?,?)";
17    private final String DELETE_JOURNAL_BY_ID_QUERY = "DELETE FROM journal MHERE id =?";
18    private final String DELETE_JOURNAL_BY_ID_QUERY = "UPDATE journal SET content = ? WHERE id=?";
19    private final String GET_JOURNAL_ID_QUERY = "SELECT * FROM journal WHERE title = ?";
19    private final String GET_JOURNAL_ID_QUERY = "SELECT id FROM journal WHERE title = ?";
19    private final String GET_JOURNAL_ID_QUERY = "SELECT * FROM journal INNER JOURNAL ID_QUERY = "SELECT * FROM journal INNER JOURN
 22
23
24
                      PreparedStatement ps = connection.prepareStatement(GET_JOURNALENTRIES_BY_TAGID_QUERY);
ps.setInt(1, tagId);
                      Resultset rs = ps.executeQuery();
List<Journal> journalsByTag = new ArrayList<Journal>();
 25
26
27
                      while (rs.next()) {
                             journalsByTag
 28
29
30
31
                                            .add/populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
                      return journalsByTag;
32<sup>©</sup>
33
34
              public JournalDAO() {
    connection = DBConnection.getConnection();
 35<sup>©</sup>
36
37
               public List<Journal> getJournals() throws SQLException {
   ResultSet rs = connection.prepareStatement(GET_JOURNALS_QUERY).executeQuery();
   List<Journal> journals = new ArrayList<Journal>();
 38
                      while (rs.next()) {
   inurnals add/nonulatelournal/rs petInt(1) rs petNate(2) rs petString(3) rs petString(4) rs petInt(5))).
       1
 🚇 Menujava 🗵 UserDAOjava 🚇 JournalTagsDA... 🔎 TagDAOjava 🖺 JournalTagsjava 👤 Userjava 🔎 Tagjava 🚇 Applicationjava 🔎 JournalDAOjava 🛭
                  public List<Journal> getJournals() throws SQLException {
   ResultSet rs = connection.prepareStatement(GET_JOURNALS_QUERY).executeQuery();
   List<Journal> journals = new ArrayList<Journal>();
   35⊜
   36
   37
                          while (rs.next()) {
   38
   39
                                  journals.add(populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
   40
   41
                          return journals:
   42
   436
                  public Journal getJournalbyId(int id) throws SQLException {
   44
                          PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_BY_ID_QUERY);
   45
46
                          ps.setInt(1, id);
                          ResultSet rs = ps.executeOuerv();
   47
                          return populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5));
   48
   49
   50°
                  public int readGetJournalId(String title) throws SQLException {
                          int journalId = 0;
                          PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_ID_QUERY);
   52
53
54
55
56
                          ps.setString(1, title);
ResultSet rs = ps.executeQuery();
                          rs.next();
journalId = rs.getInt(1);
   57
58
                           return journalId;
   596
                  public void createNewJournal(String entryName, String content, int journalTagId, int id) throws SQLException {
                          PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNAL_QUERY);
ps.setString(1, entryName);
ps.setString(2, content);
   60
   61
   62
                          ps.setInt(3, id);
ps.executeUpdate();
   64
   65
                          journalTagsDAO.createNewJournalTag(readGetJournalId(entryName), journalTagId);
   66
   67⊜
                  public void deleteJournalById(int id) throws SQLException {
                          journalTagsDAO.deleteJournalTagByJournalId(id);
   68
   69
                          PreparedStatement ps = connection.prepareStatement(DELETE_JOURNAL_BY_ID_QUERY);
                          ps.setInt(1, id);
ps.executeUpdate();
   70
   71
                                   and undatalournal Ruld (int id String content) throws SOI Excention J
```

```
☑ UserDAOjava ☑ JournalTagsDA... ☑ TagDAOjava ☑ JournalTagsjava ☑ Userjava ☑ Tagjava ☑ Application.java ☑ JournalDava ☑ Jou
45
                                 ps.setInt(1, id);
  46
                                 ResultSet rs = ps.executeQuery();
  47
                                 rs.next();
   48
                                return populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5));
   49
   50⊝
51
                     public int readGetJournalId(String title) throws SQLException {
                                int journalId = 0;
PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_ID_QUERY);
   52
53
54
55
56
                                ps.setString(1, title);
                                ResultSet rs = ps.executeQuery();
                                rs.next();
journalId = rs.getInt(1);
   57
58
59<sup>®</sup>
                                 return journalId;
                     public void createNewJournal(String entryName, String content, int journalTagId, int id) throws SQLException {
    PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNAL_QUERY);
    ps.setString(1, entryName);
   60
   61
   62
                                ps.setString(2, content);
                                ps.setInt(3, id);
ps.executeUpdate();
   63
   64
   65
                                 journalTagsDAO.createNewJournalTag(readGetJournalId(entryName), journalTagId);
   66
                     public void deleteJournalById(int id) throws SQLException {
    journalTagsDAO.deleteJournalTagByJournalId(id);
   67<sup>©</sup>
   69
                                 PreparedStatement ps = connection.prepareStatement(DELETE_JOURNAL_BY_ID_QUERY);
   70
71
72
                                 ps.setInt(1, id);
                                ps.executeUpdate();
   73<sup>©</sup>
74
75
76
                     public void updateJournalById(int id, String content) throws SQLException {
                                PreparedStatement ps = connection.prepareStatement(UPDATE_JOURNAL_BY_ID_QUERY);
                               ps.setInt(2, id);
ps.setString(1, content);
ps.executeUpdate();
   77
   78
   798
                     private Journal populateJournal(int id, Date date, String title, String content, int userId) throws SQLException {
   80
                                return new Journal(id, date, title, content, userId);
 82 }
```

JournalTagsDAO

```
🖟 Menujava 🛮 UserDAOjava 🚇 JournalTagsDA... 🛭 🖸 TagDAOjava 📳 JournalTagsjava 📳 Userjava 📳 Tagjava 🔑 Applicationjava 📳 Journal.java
     1 package dao;
     3 import java.sql.Connection;
     4 import java.sql.PreparedStatement;
    5 import java.sql.ResultSet;
    6 import java.sql.SQLException;
     7 import java.util.ArrayList;
    8 import java.util.List:
    9 import entity.JournalTags;
  10 public class JournalTagsDAO {
                   private Connection connection;
                  private Connection connection;
private final String GET_JOURNALTAGS_QUERY = "SELECT * FROM tags";
private final String GET_JOURNALTAG_BY_JOURNALID_QUERY = "SELECT * FROM journal_tags WHERE id = ?";
private final String GET_JOURNALTAG_BY_TAGID_QUERY = "SELECT * FROM journal_tags WHERE journal_tagsId = ?";
private final String CREATE_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags(journal_tags (journal_tags)";
private final String CREATE_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags(journal_tags)";
private final String CREATE_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags)";
private final String CREATE_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags);
private_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags);
private_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags);
private_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags];
private_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags];
private_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags];
private_NEW_JOURNALTAG_QUERY = "INSERT_INTO journal_tags];
private_NEW_JOURNALTAG_QUERY = "INSERT_INTO jo
  12
№14
  15
                   private final String DELETE_JOURNALTAG_BY_ID_QUERY = "DELETE FROM journal_tags where id = ?";
private final String DELETE_JOURNALTAG_BY_JOURNALID_QUERY = "DELETE FROM journal_tags where journal = ?";
  16
  17
  18
                   public JournalTagsDAO() {
  19⊝
                             connection = DBConnection.getConnection();
  20
  21
  22⊝
                   public List<JournalTags> getJournalTags() throws SQLException {
  23
                             ResultSet rs = connection.prepareStatement(GET_JOURNALTAGS_QUERY).executeQuery();
  24
                             List<JournalTags> journalTags = new ArrayList<JournalTags>();
  25
                             while (rs.next()) {
  26
                                      journalTags.add(populateJournalTags(rs.getInt(1), rs.getInt(2)));
  27
  28
                             return journalTags;
  29
   30⊝
                   public JournalTags getJournalTagByJournalId(int journalId) throws SQLException {
                            PreparedStatement ps = connection.prepareStatement(GET_JOURNALTAG_BY_JOURNALID_QUERY); ps.setInt(1, journalId);
  31
  32
  33
                             ResultSet rs = ps.executeQuery();
                             rs.next();
  35
                             return populateJournalTags(rs.getInt(1), rs.getInt(2));
  36
                    public void createNewJournalTag(int journalId, int tagId) throws SQLException {
  37⊝
                            PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNALTAG_QUERY);
ns setInt(1 __iournalId).
  38
```

```
☑ UserDAO.java 
☑ JournalTagsDA... 
☒ ☑ TagDAO.java 
☑ JournalTags.java
Menu.java
                                                                                 User.java
Tag.java
                                                                                                       Application.java
20
            connection = DBConnection.getConnection();
21
22⊝
        public List<JournalTags> getJournalTags() throws SQLException {
            ResultSet rs = connection.prepareStatement(GET_JOURNALTAGS_QUERY).executeQuery();
23
 24
            List<JournalTags> journalTags = new ArrayList<JournalTags>();
 25
            while (rs.next()) {
 26
                 journalTags.add(populateJournalTags(rs.getInt(1), rs.getInt(2)));
 27
 28
            return journalTags;
 29
        public JournalTags getJournalTagByJournalId(int journalId) throws SQLException {
 30⊝
 31
            PreparedStatement ps = connection.prepareStatement(GET_JOURNALTAG_BY_JOURNALID_QUERY);
            ps.setInt(1, journalId);
 32
 33
            ResultSet rs = ps.executeQuery();
 34
            rs.next();
 35
            return populateJournalTags(rs.getInt(1), rs.getInt(2));
 36
 37⊝
        public void createNewJournalTag(int journalId, int tagId) throws SQLException {
 38
            PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNALTAG_QUERY);
            ps.setInt(1, journalId);
ps.setInt(2, tagId);
 39
 40
41
            ps.executeUpdate();
42
43⊝
        private JournalTags populateJournalTags(int journalTagId, int tagId) {
44
            return new JournalTags(journalTagId, tagId);
45
46⊖
        public void deleteJournalTag(int id) throws SQLException {
            PreparedStatement ps = connection.prepareStatement(DELETE_JOURNALTAG_BY_ID_QUERY);
47
48
            ps.setInt(1, id);
49
            ps.executeUpdate();
 50
        public void deleteJournalTagByJournalId(int journal) throws SQLException {
 51⊝
 52
            PreparedStatement ps = connection.prepareStatement(DELETE_JOURNALTAG_BY_JOURNALID_QUERY);
 53
            ps.setInt(1, journal);
54
            ps.executeUpdate();
55
 56 }
57
```

TagDAO

```
🔑 Menujava 🛮 UserDAO.java 👪 JournalTagsDA... 🔻 TagDAO.java 🖾 🗗 JournalTags.java 🔻 User.java 🕩 Tag.java
                                                                                                             Application.ja
  1 package dao;
  3 import java.sql.Connection;
  4 import java.sql.PreparedStatement;
  5 import java.sql.ResultSet;
  6 import java.sql.SQLException;
  7 import java.util.ArrayList;
  8 import java.util.List;
  9 import entity.Tag;
 10 public class TagDAO {
         private static final String CREATE_NEW_TAG_QUERY = "INSERT INTO tags(name) VALUES(?)";
         private static final String UPDATE_TAG_BY_ID_QUERY = "UPDATE tags SET name = ? WHERE id = ?";
private static final String DELETE_TAG_BY_ID_QUERY = "DELETE FROM tags WHERE id = ?";
  12
  13
         private Connection connection;
 14
         private final String GET_ALL_TAGS_QUERY = "SELECT * FROM tags";
 15
         private final String GET_TAG_BY_ID_QUERY = "SELECT * FROM tags WHERE id = ?";
  16
         private final String GET_TAG_BY_NAME_QUERY = "SELECT * FROM tags WHERE name = ?";
 17
  18⊖
         public TagDAO() {
             connection = DBConnection.getConnection();
  19
  20
  219
          st Retrieves a tag based on it's unique id/
  22
  23
           * @param id The unique id of the tag.
 24
          * @return The tag if found, otherwise null.
  25
  26
           * @throws SQLException
  27
  28⊜
          st Retrieves a tag based on it's name.
  29
  30
           * @param id The unique name of the tag.
  31
          * @return The tag if found, otherwise null.
  32
  33
           * @throws SQLException
  34
  35⊜
         public Tag getTagByName(String name) throws SQLException {
             PreparedStatement ps = connection.prepareStatement(GET_TAG_BY_NAME_QUERY);
 36
  37
              ps.setString(1, name);
              ResultSet rs = ps.executeQuery();
 38
  39
              rs next().
```

```
🔑 Menu.java 🔃 UserDAO.java 🔑 JournalTagsDA... 🕒 TagDAO.java 🛭 🗗 JournalTags.java 🕒 User.java 🔑 Tag.java 👪 Application.ja
 37
             ps.setString(1, name);
 38
             ResultSet rs = ps.executeQuery();
 39
             rs.next();
 40
             return populateTags(rs.getInt(1), rs.getString(2));
 41
 42⊖
        private Tag populateTags(int id, String name) {
 43
             return new Tag(id, name);
 44
 45⊜
        public Tag getTagById(int id) throws SQLException {
             PreparedStatement ps = connection.prepareStatement(GET_TAG_BY_ID_QUERY);
 46
 47
             ps.setInt(1, id);
 48
             ResultSet rs = ps.executeQuery();
 49
             rs.next();
             return populateTags(rs.getInt(1), rs.getString(2));
 50
 51
 52⊜
         * Retrieves all of the available tags.
 53
 54
          * @return The enumeration of tag, if no tags are present then an empty list is
 55
 56
                   returned.
          * @throws SQLException
 57
 58
 59⊜
         public List<Tag> getAllTags() throws SQLException {
             ResultSet rs = connection.prepareStatement(GET_ALL_TAGS_QUERY).executeQuery();
 60
 61
             List<Tag> tags = new ArrayList<Tag>();
             while (rs.next()) {
 62
 63
                 tags.add(populateTags(rs.getInt(1), rs.getString(2)));
 64
 65
             return tags;
 66
        }
 67⊜
         * Creates a new tag entry.
 68
 69
 70
         * @param tag The new tag
          * @param name
 71
 72
          * @return The newly create tag if successful, otherwise null.
 73
          * @throws SQLException
 74
 75⊜
         nublic void createNewTag(String name) throws SOLExcention {
```

```
☑ Menujava ☑ UserDAO.java ☑ JournalTagsDA... ☑ TagDAO.java ☒ ☑ JournalTags.j... ☑ User.java ☑ Tag.java ☑ Application.j... ☑ JournalTags.j...
           * @param name
  71
  72
           * @return The newly create tag if successful, otherwise null.
  73
          * @throws SQLException
  74
  75⊝
          public void createNewTag(String name) throws SQLException {
  76
             PreparedStatement ps = connection.prepareStatement(CREATE_NEW_TAG_QUERY);
  77
             ps.setString(1, name);
  78
             ps.executeUpdate();
  79
  809
           * Updates the values of an existing tag.
  81
  82
          * @param id
  83
                          The id of the existing tag to modify.
  84
          * @param tag
                          The new tag information.
           * @param newTag
  85
           * @return The modified tag information if successful, otherwise null.
  86
           * @throws SQLException
  87
  88
  89⊜
          public void updateTagById(int id, String newTag) throws SQLException {
  90
             PreparedStatement ps = connection.prepareStatement(UPDATE_TAG_BY_ID_QUERY);
  91
              ps.setInt(2, id);
             ps.setString(1, newTag);
  92
  93
             ps.executeUpdate();
  94
  95⊜
  96
          * Deletes an existing tag from the database.
  97
  98
          * @param id The id of the tag to remove.
           * @return The removed tag if successful, null if otherwise.
  99
           * @throws SQLException
 100
 101
 102⊜
          public void deleteTag(int id) throws SQLException {
 103
             PreparedStatement ps = connection.prepareStatement(DELETE_TAG_BY_ID_QUERY);
 104
             ps.setInt(1, id);
 105
             ps.executeUpdate();
 106
 107⊜
          public void displayAllTags() throws SQLException {
 108
             List<Tag> tags = getAllTags();
 109
              for (Tag tag : tags) {
                  System.out.println(tag.getId() + ": " + tag.getName());
 110
 111
 112
         }
 113 }
```

UserDAO

```
🛾 Menu.java 🔟 UserDAO.java 🖂 🔎 JournalTagsDA... 🗗 TagDAO.java 🔃 JournalTags.j... 🗘 User.java 🗘 Tag.java 🕡 Application.j...
                                                                                                                                           Journal.java
1 package dao;
 3 import java.sql.Connection;
 4 import java.sql.PreparedStatement;
 5 import java.sql.ResultSet;
 6 import java.sql.SQLException;
 7 import java.util.ArrayList;
 8 import java.util.List;
 9 import entity.User;
10 public class UserDAO {
        private final String CREATE_NEW_USER_QUERY = "INSERT INTO user(firstname, lastname, emailaddress) VALUES (?, ?, ?)";
11
        private static final String UPDATE_USER_QUERY = 'INSEK! INIO user(firstname, lastname, emailaddress) VAL
private static final String UPDATE_USER_QUERY = "UPDATE user SET emailaddress = ? WHERE userId = ?";
private static final String DELETE_USER_QUERY = "DELETE FROM USER WHERE id = ?";
private final String GET_ALL_USERS_QUERY = "SELECT * FROM user";
private final String GET_USER_BY_ID_QUERY = "SELECT * FROM user WHERE id = ?";
private Connection connection.
12
14
15
         private Connection connection;
16
17⊝
        public UserDAO() {
             connection = DBConnection.getConnection();
18
19
20⊝
        public List<User> getUsers() throws SQLException {
21
             ResultSet rs = connection.prepareStatement(GET_ALL_USERS_QUERY).executeQuery();
22
             List<User> User = new ArrayList<User>();
23
             while (rs.next()) {
24
                  User.add(populateUser(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4)));
25
26
             return User;
27
28⊜
         public User getUserById(int id) throws SQLException {
29
             PreparedStatement ps = connection.prepareStatement(GET_USER_BY_ID_QUERY);
30
             ps.setInt(1, id);
31
             ResultSet rs = ps.executeQuery();
32
             rs.next();
33
             return populateUser(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4));
34
35⊝
         public void createNewUser(String firstname, String lastname, String emailaddress) throws SQLException {
36
             PreparedStatement ps = connection.prepareStatement(CREATE_NEW_USER_QUERY);
37
             ps.setString(1, firstname);
38
             ps.setString(2, lastname);
39
             ps.setString(3, emailaddress);
40
             ps.executeUpdate();
41
         public void updateUser(int id, String emailaddress) throws SQLException {
429
43
             PreparedStatement ps = connection.prepareStatement(UPDATE_USER_QUERY);
```

```
Menujava  ② UserDAOjava  ②  ② JournalTagsDA...  ② TagDAOjava  ② JournalTagsj...  ② Userjava  ② Tagjava  ② Applicationj...  ② JournalJava
private Static Times String DELETE_USER_QUERY = "SELECT * FROM user";
14  private final String GET_ALL_USERS_QUERY = "SELECT * FROM user";
         private final String GET_USER_BY_ID_QUERY = "SELECT * FROM user WHERE id = ?";
 15
 16
         private Connection connection;
 17⊝
         public UserDAO() {
              connection = DBConnection.getConnection();
 18
 19
 20⊝
         public List<User> getUsers() throws SQLException {
             ResultSet rs = connection.prepareStatement(GET_ALL_USERS_QUERY).executeQuery();
 21
 22
              List<User> User = new ArrayList<User>();
 23
             while (rs.next()) {
 24
                  User.add(populateUser(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4)));
 25
 26
             return User;
 27
 28⊝
         public User getUserById(int id) throws SQLException {
 29
             PreparedStatement ps = connection.prepareStatement(GET_USER_BY_ID_QUERY);
 30
 31
              ResultSet rs = ps.executeQuery();
 32
              rs.next();
 33
             return populateUser(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4));
 34
 35⊜
         public void createNewUser(String firstname, String lastname, String emailaddress) throws SQLException {
 36
             PreparedStatement ps = connection.prepareStatement(CREATE_NEW_USER_QUERY);
 37
             ps.setString(1, firstname);
 38
             ps.setString(2, lastname);
 39
             ps.setString(3, emailaddress);
 40
             ps.executeUpdate();
 41
 42⊖
         public void updateUser(int id, String emailaddress) throws SQLException {
             PreparedStatement ps = connection.prepareStatement(UPDATE_USER_QUERY);
 43
 44
             ps.setInt(1, id);
 45
              ps.setString(2, emailaddress);
 46
             ps.executeUpdate();
 47
 48⊖
         public void deleteUser(int id) throws SQLException {
             \label{eq:prepared} Prepared Statement \ ps \ = \ connection.prepare Statement \ (\textit{DELETE\_USER\_QUERY});
 49
 50
              ps.setInt(1, id);
 51
              ps.executeUpdate();
 52
         private User populateUser(int id, String firstname, String lastname, String emailaddress) throws SQLException {
 53Θ
 54
             return new User(id, firstname, lastname, emailaddress);
 55
 56 }
```

Journal

```
🔑 Menujava 🕑 UserDAOjava 🔑 JournalTagsDA... 🕑 TagDAOjava 🕑 JournalTagsj... 🕑 User,java 🕑 Tagjava 🔑 Applicationj... 🕑 Journal.java 🗵
 1 package entity;
 3 import java.util.ArrayList;
 4 import java.util.Date;
  5 import java.util.List;
 6 public class Journal (
       private int id;
        private Date date;
        private String title;
        private String content;
 10
        private String content;
private int user;
private List<Tag> tags;
 11
12
13
14⊜
        public Journal(int id, Date date, String title, String content) {
15
16
            this.setUser(-1);
             this.id = id;
17
18
             this.setDate(date);
             this.title = title;
 19
             this.content = content;
20
21
22
23<sup>©</sup>
24
25
26
27
28
29
             this.tags = new ArrayList<Tag>();
        public Journal(int id2, Date date2, String title2, String content2, int userById) {
             this.setUser(-1);
             this.id = id2;
             this.setDate(date2);
             this.title = title2;
             this.content = content2;
             this.user = userById;
 31⊜
        public int getId() {
 32
             return id;
33
34⊜
        public void setId(int id) {
 35
             this.id = id;
 36
 37⊝
        public String getTitle() {
 38
             return title;
 39
 40⊝
        public void setTitle(String title) {
 41
             this.title = title;
42
43<sup>©</sup>
        public String getContent() {
```

```
29
           this.user = userById;
 30
 31⊖
       public int getId() {
 32
           return id;
 33
 34⊜
       public void setId(int id) {
 35
           this.id = id;
 36
 37⊝
       public String getTitle() {
 38
           return title;
 39
 40⊝
       public void setTitle(String title) {
41
42
           this.title = title;
439
444
45
469
47
48
499
51
529
53
54
56
57
589
62
63
64
659
66
67
68
69
70
       public String getContent() {
           return content;
       public void setContent(String content) {
           this.content = content;
       public Date getDate() {
           return date;
       public void setDate(Date date) {
           this.date = date;
       public int getUser() {
           return user;
       public void setUser(int user) {
           this.user = user;
       public List<Tag> getTags() {
           return tags;
       public void addTag(Tag tag) {
           if (tag != null) {
               tags.add(tag);
```

JournalTags

```
Menu.java
             UserDAO.java
                             Journal Tags DA...
                                               TagDAO.java
                                                               🗓 JournalTags.j... 🖂 🔃 User.jav
  1 package entity;
 2
 3⊝ import java.util.ArrayList;
 4 import java.util.List;
 5 public class JournalTags {
        private int journalId;
 7
        private int tagId;
 8
        private List<Tag> tags;
 9⊝
        public JournalTags(int journalId, int tagId) {
10
            this.journalId = journalId;
11
            this.tagId = tagId;
12
            this.tags = new ArrayList<Tag>();
13
14⊝
        public int getJournalId() {
            return journalId;
15
16
17⊝
        public void setJournalId(int journalId) {
18
            this.journalId = journalId;
19
20⊝
        public int getTagId() {
21
            return tagId;
22
23⊜
        public void setTagId(int tagId) {
24
            this.tagId = tagId;
25
26⊜
        public List<Tag> getTags() {
27
            return tags;
28
        }
29⊝
        public void setTags(List<Tag> tags) {
30
            this.tags = tags;
31
        }
32 }
```

Tag

```
Menu.java
            UserDAO.java
                            Journal Tags DA...
                                            TagDAO
 1 package entity;
  3 public class Tag {
        private int id;
  4
        private String name;
  5
        public Tag(int id, String name) {
  6⊜
 7
            this.id = id;
            this.name = name;
 8
 9
        public int getId() {
10⊝
11
            return id;
12
        public void setId(int id) {
13⊝
            this.id = id;
14
15
        public String getName() {
16⊜
17
            return name;
18
        public void setName(String name) {
19⊝
20
            this.name = name;
21
        }
22 }
```

User

```
UserDAO.java
JournalTagsDA...

☑ JournalTags.j...
☑ User.java 
☒ ☑ Tag.java
Menu.java
                                            TagDAO.java
 1 package entity;
 2
 3 public class User {
 4
        private int userId;
 5
        private String firstname;
 6
        private String lastname;
 7
        private String emailAddress;
 8
 9⊝
        public User(int id, String firstname, String lastname, String emailAddress)
10
            this.userId = id;
11
12
            this.firstname = firstname;
13
            this.lastname = lastname;
14
            this.emailAddress = emailAddress;
15
16⊜
        public int getUserId() {
17
            return userId;
18
        public void setUserId(int userId) {
19⊖
20
            this.userId = userId;
21
22⊖
        public String getFirstname() {
23
            return firstname;
24
25⊜
        public void setFirstname(String firstname) {
26
            this.firstname = firstname;
27
28⊝
        public String getLastname() {
29
            return lastname;
30
31⊖
        public void setLastname(String lastname) {
32
            this.lastname = lastname;
33
34⊝
        public String getEmailAddress() {
35
            return emailAddress;
36
37⊝
        public void setEmailAddress(String emailaddress) {
38
            this.emailAddress = emailaddress;
39
40 }
```

SQL Database

```
birthday_per_dept
Query 1
                           SQL File 2*
                                       SQL File 3*
        | 🗲 f 👰 🔘 | 😘 | 🕝 🚳 | Limit to 1000 rows
                                                         - | 🏡 | 🥩 🔍 🗻 🖘
  1 •
         create database if not exists goaltivity;
   2
         use goaltivity;
         drop table if exists journal_tags;
   3
         drop table if exists journal;
         drop table if exists tags;
   5
   6
         drop table if exists user;
         -- drop table if exists reminders;
         -- drop table if exists events;
   8
  9
      ⊖ create table user (
              id int auto increment not null,
 10
            firstname varchar(20) not null,
 11
            lastname varchar(30) not null,
 12
             emailaddress varchar(100) not null,
 13
             primary key (id)
 15
         );
 16
      id int auto increment not null,
 17
             date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
 18
            title varchar(256) not null,
 19
            content text not null,
 20
            user int not null,
 21
             primary key (id),
  22
             foreign key (user) references user(id)
 23
  24
        );
         -- Diary, Travel, Work, Computer, Photo, Lifestyle, General
 25
      26
 27
             id int auto_increment not null,
  28
           name varchar(20) NOT NULL,
```

```
24 );
25 -- Diary, Travel, Work, Computer, Photo, Lifestyle, General
id int auto_increment not null,
28
      name varchar(20) NOT NULL,
29
     primary key (id)
30
31
     -- A journal can have multiple tags associated with it.
journal int NOT NULL,
33
      tag int NOT NULL,
      id int NOT NULL auto_increment,
35
     primary key(id),
36
37
     -- foreign key(journal) references journal(id),
38
      foreign key(tag) references tags(id)
40 INSERT INTO tags (name) VALUES("Diary"),("Travel"), ("Work"), ("Computer"), ("Photo"), ("Lifestyle"), ("General");
41 SELECT * FROM tags;
42 • Select * from journal_tags;
```

Screenshots of Running Application:

```
Application (1) [Java Application] C:\Program Files\Java\jdk-16.0
Connection is successful
Select an Option:
 -----
1) Display all Users
2) Select a User
Create a User
4) Delete a User
3
Enter your first name:
kelly
Enter your last name:
shelly
Enter your email address:
kellyshelly@email.com
Press enter to continue.
Select an Option:
 -----
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
1: Jojo Jelly Bean Cinderella
3: kelly shelly
Press enter to continue.
Select an Option:
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
Please delete all journal entries before attempting to delete a user. Enter -1 to go back if needed.
Enter user ID for the account you want to remove:
1: Jojo Jelly Bean Cinderella
3: kelly shelly
Press enter to continue.
Select an Option:
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
```

```
Select an Option:
-----
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
1: Jojo Jelly Bean Cinderella
Enter your user ID
|-----
Welcome Jojo Jelly Bean
-----
Select an Option:
_____
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
```

```
Welcome Jojo Jelly Bean
-----
Select an Option:
_____
1) Create a Journal Entry
2) Display all Journal Entries
Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
What would you like to name your new journal entry
Entry 1
Enter journal content:
this is content
Please confirm your identity by entering your user id.
1: Jojo Jelly Bean Cinderella
1: Diary
2: Travel
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
Add a tag id for this post:
Select an Option:
-----
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
```

5) View journal tag options

```
Select an Option:

1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
2
Journal entries
2: 2021-08-17 fnhj
hhgek
TAG: 3: Trying my Best
3: 2021-08-17 Entry 1
this is content
TAG: 8: Depressos Espressos
```

Press enter to continue.

```
■ Console 

□
Application (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (Aug 17, 2021, 9:57:11 PM)
rress enter to continue.
Select an Option:
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
2: 2021-08-17 fnhj
        hhgek
         TAG: 3: Trying my Best
3: 2021-08-17 Entry 1
        this is content
         TAG: 8: Depressos Espressos
Which journal entry would you like to update?
Enter the new content:
This is the updated content, update update update
Select an Option:
_____
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
What would you like to name your new journal entry
bleh
Enter journal content:
why do I always accidentally push one
Please confirm your identity by entering your user id.
1: Jojo Jelly Bean Cinderella
```

```
■ Console 

□
Application (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (Aug 17, 20)
Enter Journal content:
why do I always accidentally push one
Please confirm your identity by entering your user id.
1: Jojo Jelly Bean Cinderella
1
1: Diary
2: Travel
3: Trying my Best
4: Computer
5: Photo
6: Lifestvle
8: Depressos Espressos
Add a tag id for this post:
Select an Option:
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
Journal entries
2: 2021-08-17 fnhj
        This is the updated content, update update update
         TAG: 3: Trying my Best
3: 2021-08-17 Entry 1
        this is content
         TAG: 8: Depressos Espressos
4: 2021-08-17 bleh
        why do I always accidentally push one
         TAG: 3: Trying my Best
Press enter to continue.
```

■ Console \(\times \) Application (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (Aug 17, 2021, 9:57:11 PM) 4) Delete a Journal Entry 5) View journal tag options 2: 2021-08-17 fnhj This is the updated content, update update update TAG: 3: Trying my Best 3: 2021-08-17 Entry 1 this is content TAG: 8: Depressos Espressos 4: 2021-08-17 bleh why do I always accidentally push one TAG: 3: Trying my Best Enter the id of the entry you would like to delete: Select an Option: ______ 1) Create a Journal Entry 2) Display all Journal Entries 3) Update a Journal Entry 4) Delete a Journal Entry 5) View journal tag options Journal entries 2: 2021-08-17 fnhj This is the updated content, update update update TAG: 3: Trying my Best

3: 2021-08-17 Entry 1 this is content

TAG: 8: Depressos Espressos

Press enter to continue.

Press enter to continue.

Select an Option:

1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
5
Journal Tags

Select an Option:

1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag

```
Select an Option:
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
Journal Tags
Select an Option:
 -----
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
Delete tag
1
1: Diary
2: Travel
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
Select an Option:
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
Delete tag
```

```
Select an Option:
 -----
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
Enter new tag name:
Mini Victories
Select an Option:
-----
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
1: Diary
2: Travel
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
9: Mini Victories
Select an Option:
-----
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
```

```
1) Display all tags
2) Search for Journal Entries via tag
Create new tag
Update tag
5) Delete tag
Enter the id of the tag you would like to remove:
Select an Option:
1) Display all tags
Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
1: Diary
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
9: Mini Victories
Select an Option:
 -----

    Display all tags

2) Search for Journal Entries via tag
3) Create new tag
Update tag
5) Delete tag
```

```
4) ираате тад
5) Delete tag
1
1: Diary
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
9: Mini Victories
Select an Option:
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
2
1: Diary
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
9: Mini Victories
Enter the id of the tag which you would like to view the journal entries from:
3: 2021-08-17 Entry 1
       this is content
Select an Option:
 -----
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
```

URL to GitHub Repository:

https://github.com/JoleneMel/GoaltivityGroupProject