

Exemplar code Iteration 2

A. models/bookmarks.rb

```
1 module Bookmark
2
3   def Bookmark.new(bookmark_name, link, description, creator, last_updated)
4     result = false
5     insert = "INSERT INTO bookmarks (bookmark_name, link, description, creator, last_updated, report_status)"
6     begin
7       $db.execute insert, bookmark_name, link, description, creator, last_updated, 1
8       result = true
9       puts "Insertion success"
10    rescue SQLite3::ConstraintException
11      puts "Insertion Failed"
12    end
13    return result
14  end
15
16  def Bookmark.find_by(search, tags)
17    # something is wrong, wont filter if 2 selected, only works for 1 at a time
18    # issue probably to do with dynamic generation of query
19    # the sql statement works perfectly in db browser
20    searchTerm = '%' + search + '%'
21    if tags == 0
22      query = "SELECT DISTINCT * FROM bookmarks WHERE bookmark_name LIKE (?) OR link LIKE (?);"
23      result = $db.execute query, searchTerm, searchTerm
24    else
25      tagsString = ""
26      tags.each_value do |tag|
27        tagsString = tagsString + tag.to_s + ', '
28      end
29      #get rid of extra ", " at the end of string
30      tagsString = tagsString.chop
31      tagsString = tagsString.chop
32
33      #build the query
34      query = "SELECT DISTINCT bookmarks.bookmark_id, bookmark_name, link, description,
35              creator, last_updated, report_status, rating, num_ratings
36              FROM bookmarks
37              JOIN tagged_bookmarks ON bookmarks.bookmark_id = tagged_bookmarks.bookmark_id
38              JOIN tags ON tagged_bookmarks.tag_id = tags.tag_id
39              WHERE tags.name IN ((?))
40              AND (bookmark_name LIKE (?) OR link LIKE (?));"
41
42      result = $db.execute query, tagsString, searchTerm, searchTerm
43    end
44
45    return result
46  end
47
48  def Bookmark.find_by_id(bookmark_id)
49    result = Array.new
50    query = "SELECT * FROM bookmarks WHERE bookmark_id = (?);"
51    result = $db.execute query, bookmark_id
52    return result[0]
53  end
54
55  def Bookmark.update(bookmark_name, link, description, last_updated, bookmark_id, active_status)
56    result = false
57    query = "UPDATE bookmarks SET bookmark_name = ?, link = ?,
58            description = ?, last_updated = ?, report_status = ? WHERE bookmark_id = ?;"
59    begin
60      $db.execute query, bookmark_name, link, description, last_updated, active_status, bookmark_id
61      result = true
62      puts "Insertion success"
63    rescue SQLite3::ConstraintException
```

```

94         puts "New rating submitted"
95     rescue SQLite3::ConstraintException
96         puts "Insertion Failed"
97     end
98     return result
99 end
100
101 #method to get the truncated url without subpages
102 def Bookmark.getHost(url)
103     uri = URI(url)
104     return uri.host
105 end
106
107 #returns the tags for a specific bookmark_id
108 def Bookmark.getTags(id)
109     query = "SELECT tag_id FROM tagged_bookmarks WHERE bookmark_id = ?"
110     result = $db.execute query, id
111     if (result.count != 0)
112         tag_list = Array.new
113         for i in 0..(result.count-1)
114             query = "SELECT name FROM tags WHERE tag_id = ?"
115             list = ($db.execute query, result[i])
116             tag_list[i] = list[0]
117         end
118         output = tag_list.join(", ")
119     else
120         output = "No Tags"
121     end
122     return output
123 end
124 end

```

B. database/createDatabase.rb

```

1  require 'sqlite3'
2  require 'csv'
3
4  csv_text_feedback = File.read("public/csv/feedback.csv")
5  csvFeedback = CSV.parse(csv_text_feedback, :headers => true)
6
7  csv_text_bookmark_history = File.read("public/csv/bookmark_history.csv")
8  csvBookmarkHistory = CSV.parse(csv_text_bookmark_history, :headers => true)
9
10 csv_text_bookmarks = File.read("public/csv/bookmarks.csv")
11 csvBookmarks = CSV.parse(csv_text_bookmarks, :headers => true)
12
13 csv_text_users = File.read("public/csv/users.csv")
14 csvUsers = CSV.parse(csv_text_users, :headers => true)
15
16 csv_text_favourites = File.read("public/csv/favourites.csv")
17 csvFavourites = CSV.parse(csv_text_favourites, :headers => true)
18
19 csv_text_login_history = File.read("public/csv/login_history.csv")
20 csvLoginHistory = CSV.parse(csv_text_login_history, :headers => true)
21
22 csv_text_tags = File.read("public/csv/tags.csv")
23 csvTags = CSV.parse(csv_text_tags, :headers => true)
24
25 csv_text_tagged_bookmarks = File.read("public/csv/tagged_bookmarks.csv")
26 csvTaggedBookmarks = CSV.parse(csv_text_tagged_bookmarks, :headers => true)
27
28 inputs = ARGV
29 if inputs[0] == "testdb"
30     DB = SQLite3::Database.new 'database/acme_test_db.sqlite'

```

```

32 DB = SQLite3::Database.new 'database/acme_db.sqlite'
33 end
34
35 # Create tables
36 DB.execute <<-SQL
37 CREATE TABLE IF NOT EXISTS "tagged_bookmarks" (
38     "bookmark_id" INTEGER,
39     "tag_id" INTEGER,
40     FOREIGN KEY("tag_id") REFERENCES "tags"("tag_id"),
41     FOREIGN KEY("bookmark_id") REFERENCES "bookmarks"("bookmark_id")
42 );
43 SQL
44
45 DB.execute <<-SQL
46 CREATE TABLE IF NOT EXISTS "tags" (
47     "tag_id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
48     "name" TEXT
49 );
50
51 SQL
52
53 DB.execute <<-SQL
54 CREATE TABLE IF NOT EXISTS "bookmark_history" (
55     "user_id" INTEGER,
56     "bookmark_id" INTEGER,
57     "date" INTEGER,
58     FOREIGN KEY("user_id") REFERENCES "users"("user_id"),
59     FOREIGN KEY("bookmark_id") REFERENCES "bookmarks"("bookmark_id")
60 );
61 SQL
62
63 DB.execute <<-SQL
64 CREATE TABLE IF NOT EXISTS "login_history" (
65     "user_id" INTEGER,
66     "date" INTEGER,
67     PRIMARY KEY("user_id","date")
68 );
69 SQL
70
71 DB.execute <<-SQL
72 CREATE TABLE IF NOT EXISTS "favourites" (
73     "user_id" INTEGER,
74     "bookmark_id" INTEGER,
75     FOREIGN KEY("user_id") REFERENCES "users"("user_id"),
76     PRIMARY KEY("user_id","bookmark_id"),
77     FOREIGN KEY("bookmark_id") REFERENCES "bookmarks"("bookmark_id")
78 );
79 SQL
80
81 DB.execute <<-SQL
82 CREATE TABLE IF NOT EXISTS "bookmarks" (
83     "bookmark_id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
84     "bookmark_name" TEXT,
85     "link" TEXT UNIQUE,
86     "description" TEXT,
87     "creator" INTEGER,
88     "last_updated" TEXT,
89     "report_status" INTEGER DEFAULT 2,
90     "rating" INTEGER DEFAULT 0,
91     "num_ratings" INTEGER DEFAULT 0
92 );
93 SQL

```

```

93 SQL
94
95 DB.execute <<-SQL
96 CREATE TABLE IF NOT EXISTS "users" (
97   "user_id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
98   "username" TEXT NOT NULL UNIQUE,
99   "first_name" TEXT,
100  "last_name" TEXT,
101  "password" TEXT NOT NULL,
102  "email" TEXT NOT NULL,
103  "active_status" INTEGER NOT NULL DEFAULT 0,
104  "admin" INTEGER NOT NULL DEFAULT 0,
105  "new" INTEGER NOT NULL DEFAULT 1
106 );
107 SQL
108
109 DB.execute <<-SQL
110 CREATE TABLE IF NOT EXISTS "feedback" (
111   "feedback_id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
112   "user_id" TEXT NOT NULL,
113   "date_time" TEXT NOT NULL,
114   "feedback_topic" TEXT NOT NULL,
115   "feedback" TEXT NOT NULL,
116   "resolved" INTEGER NOT NULL DEFAULT 0,
117   "bookmark_id" DEFAULT NULL,
118   FOREIGN KEY("user_id") REFERENCES "users"("user_id")
119 );
120 SQL

```



```

124 csvTaggedBookmarks.each do |row|
125   DB.execute "insert into tagged_bookmarks values ( ?, ?)", row.fields
126 end
127
128 csvBookmarks.each do |row|
129   DB.execute "insert into bookmarks values ( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", row.fields
130 end
131
132
133 csvBookmarkHistory.each do |row|
134   DB.execute "insert into bookmark_history values ( ?, ?, ?)", row.fields
135 end
136
137 csvUsers.each do |row|
138   DB.execute "insert into users values ( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", row.fields
139 end
140
141 csvTags.each do |row|
142   DB.execute "insert into tags values ( ?, ?)", row.fields
143 end
144
145 csvFeedback.each do |row|
146   DB.execute "insert into feedback values ( ?, ?, ?, ?, ?, ?, ?, ?)", row.fields
147 end
148
149 csvLoginHistory.each do |row|
150   DB.execute "insert into login_history values ( ?, ?)", row.fields
151 end

```



```

152
153 csvFavourites.each do |row|
154   DB.execute "insert into favourites values ( ?, ?)", row.fields
155 end
156

```

C. app/history.rb

```
1 module History
2
3   def History.new(user_id, bookmark_id, date)
4     result = false
5     insert = "INSERT INTO bookmark_history (user_id, bookmark_id, date) VALUES (?, ?, ?)"
6     begin
7       $db.execute insert, user_id, bookmark_id, date
8       result = true
9       puts "Insertion success"
10    rescue SQLite3::ConstraintException
11      puts "Insertion Failed"
12    end
13    return result
14  end
15
16  def History.get_last_visit_date(bookmark_id)
17    result = Array.new
18    query = "SELECT date FROM bookmark_history WHERE bookmark_id = ?;"
19    result = $db.execute query, bookmark_id
20    return result[result.length-1]
21  end
22
23  def History.update_date(bookmark_id, date)
24    query = "UPDATE bookmark_history SET date = ? WHERE bookmark_id = ?;"
25    begin
26      $db.execute query, date, bookmark_id
27      result = true
28      puts "Last visit date updated"
29    rescue SQLite3::ConstraintException
30      puts "Insertion Failed"
31    end
32    return result
33  end
34
35 end
```

D. app/signup.rb

```
1 get '/signup' do
2   Bookmark.getTags(2)
3   erb :signup
4 end
5
6 post '/signup' do
7
8   @first_name = params[:first_name].strip
9   @last_name = params[:last_name].strip
10  @username = params[:username].strip
11  @email = params[:email].strip
12  @confirmed_email = params[:email_confirm].strip
13  @confirmed_password = params[:password_confirm].strip
14  @password = params[:password].strip
15  @email_used = User.find_email(@email)
16  @username_used = User.find_username(@username)
17
18  if @email_used
19    flash[:info] = "Sorry, this email is already registered"
20    redirect '/signup'
21  elsif @username_used
22    flash[:info] = "Sorry, this username is already used by a different user"
23    redirect '/signup'
24  elsif @email == @confirmed_email && @password == @confirmed_password
25    User.new(@username, @first_name, @last_name, @password, @email)
26    flash[:info] = "Your account is waiting for admin verification"
27    redirect '/login'
28  else
29    redirect '/signup'
30  end
31
32 end
```


E. views/bookmark_table.erb

```
1 <table class="table">
2   <thead>
3     <tr>
4       <th>No.</th>
5       <th>Bookmark Name</th>
6       <th>Link</th>
7       <th>Rating</th>
8       <th>Last Updated</th>
9       <th>Status</th>
10      <th>More Info</th>
11    </tr>
12  </thead>
13
14  <tbody id="bookmark_table">
15
16    <% @bookmark_list.each do |bookmark| %>
17      <tr>
18
19        <td><%= @bookmark_list.find_index(bookmark) + 1 %></td>
20        <td><%= bookmark[1] %></td>
21
22        <td>
23          <form class="history" method="post" action="/add_to_history" target="_blank">
24            <input type="hidden" name="bookmark_id" value="<%= bookmark[0] %>" />
25            <input class="btn btn-info btn-sm" type="submit" value="<%= Bookmark.getHost(bookmark[2]) %>" />
26            <input type="hidden" name="bookmark_url" value="<%= bookmark[2] %>" />
27          </form>
28        </td>
29
30        <td>
31          <form class="rating" method="post" action="/submit_rating" style="justify-content: left;">
32
33            <input type="hidden" name="bookmark_id" value="<%= bookmark[0] %>" />
34            <input type="hidden" name="num_ratings" value="<%= bookmark[8] %>" />
35            <input type="hidden" name="current_rating" value="<%= bookmark[7] %>" />
36            <div class="stars">
37              <% case bookmark[7].to_i when 0 %>
38                <button id="rating" type="submit" name="stars" value="5">&#9733;</button>
39                <button id="rating" type="submit" name="stars" value="4">&#9733;</button>
40                <button id="rating" type="submit" name="stars" value="3">&#9733;</button>
41                <button id="rating" type="submit" name="stars" value="2">&#9733;</button>
42                <button id="rating" type="submit" name="stars" value="1">&#9733;</button>
43              <% when 1 %>
44                <button id="rating" type="submit" name="stars" value="5">&#9733;</button>
45                <button id="rating" type="submit" name="stars" value="4">&#9733;</button>
46                <button id="rating" type="submit" name="stars" value="3">&#9733;</button>
47                <button id="rating" type="submit" name="stars" value="2">&#9733;</button>
48                <button id="rating" type="submit" name="stars" value="1" style="color: #fed136;">&#9733;</button>
49              <% when 2 %>
50                <button id="rating" type="submit" name="stars" value="5">&#9733;</button>
51                <button id="rating" type="submit" name="stars" value="4">&#9733;</button>
52                <button id="rating" type="submit" name="stars" value="3">&#9733;</button>
53                <button id="rating" type="submit" name="stars" value="2" style="color: #fed136;">&#9733;</button>
54                <button id="rating" type="submit" name="stars" value="1" style="color: #fed136;">&#9733;</button>
55              <% when 3 %>
56                <button id="rating" type="submit" name="stars" value="5">&#9733;</button>
57                <button id="rating" type="submit" name="stars" value="4">&#9733;</button>
58                <button id="rating" type="submit" name="stars" value="3" style="color: #fed136;">&#9733;</button>
59
60            </div>
61          </form>
62
63          <td>
64            <% case bookmark[6] when 0 %>
65              <img src='images/red.png' class="icon" alt="broken link" />
66            <% when 1 %>
67              <img src='images/wrench.png' class="icon" alt="link under review" />
68            <% when 2 %>
69              <img src='images/green.png' class="icon" alt="working link" />
70            <% else %>
71              <img src='images/green.png' class="icon" alt="working link" />
72            <% end %>
73          </td>
74
75          <td>
76            <case bookmark[6] when 2 %>
77              <a style="color:gray; text-decoration:underline; font-size:13px" href="/report/<%= bookmark[0] %>">Report</a>
78            <% end %>
79          </td>
80
81          <td id="more_info">
82            <a tabindex="0" class="btn btn-sm btn-info" role="button" data-html="true" data-toggle="popover" data-trigger="focus"
83              title="<%= bookmark[1] %>"
84              data-content="<div><b>Bookmark ID: <%= bookmark[0] %></b></div>
85                <div><b>Description: </b><%= bookmark[4] %> </div>
86                <div><b>Creator: </b><%= ($db.execute "SELECT email FROM users WHERE user_id = ?",bookmark[4])[0][0] %></div>
87                <div><b>Department tags: </b><%= Bookmark.getTags(bookmark[0]) %></div>
88                ">More Info</a>
89          </td>
90        </tr>
91      </tbody>
92    </table>
```

F. app/admin.rb

```
1 ▽ get '/admin' do
2 ▽   if !session[:admin]
3 ▽     flash[:warning] = "You wandered into protected space, please login to continue"
4 ▽     redirect '/login'
5 ▽   else
6 ▽     erb :admin_dashboard
7 ▽   end
8 ▽ end
9
10 #get method for admin users, retrieves users from db and stores it in an array
11 ▽ get '/admin_users' do
12 ▽   if !session[:admin]
13 ▽     flash[:warning] = "You wandered into protected space, please login to continue"
14 ▽     redirect '/login'
15 ▽   else
16 ▽     @user_list = $db.execute "SELECT * FROM users ORDER BY user_id ASC"
17 ▽     @new_user_list = $db.execute "SELECT * FROM users WHERE new = 1 ORDER BY user_id ASC"
18 ▽     erb :admin_user_mgmt
19 ▽   end
20 ▽ end
21
22 #get method for admin bookmarks, retrieves bookmarks from db and stores it in an array
23 ▽ get '/admin_bookmarks' do
24 ▽   if !session[:admin]
25 ▽     flash[:warning] = "You wandered into protected space, please login to continue"
26 ▽     redirect '/login'
27 ▽   else
28 ▽     @admin_bookmark_list = $db.execute "SELECT * FROM bookmarks ORDER BY bookmark_id ASC"
29 ▽     erb :admin_bookmark_mgmt
30 ▽   end
31 ▽ end
32
33 #get method for feedback, retrieves feedback from db and stores it in an array
34 ▽ get '/admin_feedback' do
35 ▽   if !session[:admin]
36 ▽     flash[:warning] = "You wandered into protected space, please login to continue"
37 ▽     redirect '/login'
38 ▽   else
39 ▽     @admin_feedback_list = $db.execute "SELECT * FROM feedback ORDER BY feedback_id ASC"
40 ▽     erb :admin_feedback
41 ▽   end
42 ▽ end
43
44 #get method to edit specific user
45 ▽ get '/admin_edit_user/:id' do
46 ▽   if !session[:admin]
47 ▽     flash[:warning] = "You wandered into protected space, please login to continue"
48 ▽     redirect '/login'
49 ▽   end
50
51   redirect '/index' if !session[:admin]
52
53   session[:editing_id] = params["id"]
54
55   redirect '/admin_edit_user'
56 ▽ end
57
58 ▽ get '/admin_edit_user' do
59 ▽   if !session[:admin]
60 ▽     flash[:warning] = "You wandered into protected space, please login to continue"
61 ▽     redirect '/login'
62 ▽   else
63 ▽     @admin_editing_user = User.admin_find_user(session[:editing_id])
```

```
64         erb :admin_edit_user
65     end
66 end
67
68 ##-----Post Methods-----#
69
70
71 ▼ post '/admin_add_user' do
72     User.approve(params[:userid])
73     redirect '/admin_users'
74 end
75
76 ▼ post '/admin_update_user' do
77     @user_id = params[:user_id]
78     @first_name = params[:fname].strip
79     @last_name = params[:lname].strip
80     @username = params[:username].strip
81     @email = params[:email].strip
82     @active_status = params[:active_status].strip
83
84
85     User.update(@user_id,@username, @first_name, @last_name, @email, @active_status)
86     redirect '/admin_users'
87 end
88
89 ▼ post '/admin_resolve_feedback' do
90     Feedback.approve(params[:feedbackid])
91     redirect '/admin_feedback'
92 end
```