Exemplar code Iteration 2

A. models/bookmarks.rb

```ruby
module Bookmark

    def Bookmark.new(bookmark_name, link, description, creator, last_updated)
        result = false
        insert = "INSERT INTO bookmarks (bookmark_name, link, description, creator, last_updated, report_status)
        begin
            $db.execute insert, bookmark_name, link, description, creator, last_updated, 1
            result = true
            puts "Insertion success"
        rescue SQLite3::ConstraintException
            puts "Insertion Failed"
        end
        return result
    end

    def Bookmark.find_by(search, tags)
        # something is wrong, wont filter if 2 selected, only works for 1 at a time
        # issue probably to do with dynamic generation of query
        # the sql statement works perfectly in db browser
        searchTerm = '%' + search + '%'
        if tags == 0
            query = "SELECT DISTINCT * FROM bookmarks WHERE bookmark_name LIKE (?) OR link LIKE (?);"
            result = $db.execute query, searchTerm, searchTerm
        else
            tagsString = ""
            tags.each_value do |tag|
                tagsString =  tagsString + tag.to_s + ', '
            end
            #get rid of extra ", " at the end of string
            tagsString = tagsString.chop
            tagsString = tagsString.chop

            #build the query
            query = "SELECT DISTINCT bookmarks.bookmark_id, bookmark_name, link, description,
                    creator, last_updated, report_status, rating, num_ratings
                    FROM bookmarks
                    JOIN tagged_bookmarks ON bookmarks.bookmark_id = tagged_bookmarks.bookmark_id
                    JOIN tags ON tagged_bookmarks.tag_id = tags.tag_id
                    WHERE tags.name IN ((?))
                    AND (bookmark_name LIKE (?) OR link LIKE (?));"

            result = $db.execute query, tagsString, searchTerm, searchTerm
        end

        return result
    end

    def Bookmark.find_by_id(bookmark_id)
        result = Array.new
        query = "SELECT * FROM bookmarks WHERE bookmark_id = (?);"
        result = $db.execute query, bookmark_id
        return result[0]
    end

    def Bookmark.update(bookmark_name, link, description, last_updated, bookmark_id, active_status)
        result = false
        query = "UPDATE bookmarks SET bookmark_name = ?, link = ?,
                description = ?, last_updated = ?, report_status = ? WHERE  bookmark_id = ?;"
        begin
            $db.execute query, bookmark_name, link, description, last_updated, active_status, bookmark_id
            result = true
            puts "Insertion success"
        rescue SQLite3::ConstraintException
```

```
            puts "New rating submitted"
        rescue SQLite3::ConstraintException
            puts "Insertion Failed"
        end
        return result
    end

    #method to get the truncated url without subpages
    def Bookmark.getHost(url)
       uri = URI(url)
       return uri.host
    end

    #returns the tags for a specific bookmark_id
    def Bookmark.getTags(id)
       query = "SELECT tag_id FROM tagged_bookmarks WHERE bookmark_id = ?"
       result = $db.execute query, id
       if (result.count != 0)
          tag_list = Array.new
          for i in 0..(result.count-1)
             query = "SELECT name FROM tags WHERE tag_id = ?"
             list = ($db.execute query, result[i])
             tag_list[i] = list[0]
          end
          output = tag_list.join(", ")
       else
          output = "No Tags"
       end
       return output
    end
end
```

B. database/createDatabase.rb

```
require 'sqlite3'
require 'csv'

csv_text_feedback = File.read("public/csv/feedback.csv")
csvFeedback = CSV.parse(csv_text_feedback, :headers => true)

csv_text_bookmark_history = File.read("public/csv/bookmark_history.csv")
csvBookmarkHistory = CSV.parse(csv_text_bookmark_history, :headers => true)

csv_text_bookmarks = File.read("public/csv/bookmarks.csv")
csvBookmarks = CSV.parse(csv_text_bookmarks, :headers => true)

csv_text_users = File.read("public/csv/users.csv")
csvUsers = CSV.parse(csv_text_users, :headers => true)

csv_text_favourites = File.read("public/csv/favourites.csv")
csvFavourites = CSV.parse(csv_text_favourites, :headers => true)

csv_text_login_history = File.read("public/csv/login_history.csv")
csvLoginHistory = CSV.parse(csv_text_login_history, :headers => true)

csv_text_tags = File.read("public/csv/tags.csv")
csvTags = CSV.parse(csv_text_tags, :headers => true)

csv_text_tagged_bookmarks = File.read("public/csv/tagged_bookmarks.csv")
csvTaggedBookmarks = CSV.parse(csv_text_tagged_bookmarks, :headers => true)

inputs = ARGV
if inputs[0] == "testdb"
  DB = SQLite3::Database.new 'database/acme_test_db.sqlite'
```

```ruby
  DB = SQLite3::Database.new 'database/acme_db.sqlite'
end

# Create tables
DB.execute <<-SQL
CREATE TABLE IF NOT EXISTS "tagged_bookmarks" (
    "bookmark_id"   INTEGER,
    "tag_id"    INTEGER,
    FOREIGN KEY("tag_id") REFERENCES "tags"("tag_id"),
    FOREIGN KEY("bookmark_id") REFERENCES "bookmarks"("bookmark_id")
);
SQL

DB.execute <<-SQL
CREATE TABLE IF NOT EXISTS "tags" (
    "tag_id"    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    "name"  TEXT
);

SQL

DB.execute <<-SQL
CREATE TABLE IF NOT EXISTS "bookmark_history" (
    "user_id"   INTEGER,
    "bookmark_id"   INTEGER,
    "date"  INTEGER,
    FOREIGN KEY("user_id") REFERENCES "users"("user_id"),
    FOREIGN KEY("bookmark_id") REFERENCES "bookmarks"("bookmark_id")
);
SQL

```

```ruby
DB.execute <<-SQL
CREATE TABLE IF NOT EXISTS "login_history" (
    "user_id"   INTEGER,
    "date"  INTEGER,
    PRIMARY KEY("user_id","date")
);
SQL

DB.execute <<-SQL
CREATE TABLE IF NOT EXISTS "favourites" (
    "user_id"   INTEGER,
    "bookmark_id"   INTEGER,
    FOREIGN KEY("user_id") REFERENCES "users"("user_id"),
    PRIMARY KEY("user_id","bookmark_id"),
    FOREIGN KEY("bookmark_id") REFERENCES "bookmarks"("bookmark_id")
);
SQL

DB.execute <<-SQL
CREATE TABLE IF NOT EXISTS "bookmarks" (
    "bookmark_id"   INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    "bookmark_name" TEXT,
    "link"  TEXT UNIQUE,
    "description"   TEXT,
    "creator"   INTEGER,
    "last_updated"  TEXT,
    "report_status" INTEGER DEFAULT 2,
    "rating"    INTEGER DEFAULT 0,
    "num_ratings"   INTEGER DEFAULT 0
);
SQL
```

```
SQL

DB.execute <<-SQL
CREATE TABLE IF NOT EXISTS "users" (
    "user_id"   INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    "username"  TEXT NOT NULL UNIQUE,
    "first_name"    TEXT,
    "last_name" TEXT,
    "password"  TEXT NOT NULL,
    "email" TEXT NOT NULL,
    "active_status" INTEGER NOT NULL DEFAULT 0,
    "admin" INTEGER NOT NULL DEFAULT 0,
   "new"    INTEGER NOT NULL DEFAULT 1
);
SQL

DB.execute <<-SQL
CREATE TABLE IF NOT EXISTS "feedback" (
    "feedback_id"   INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    "user_id"   TEXT NOT NULL,
    "date_time" TEXT NOT NULL,
   "feedback_topic" TEXT NOT NULL,
    "feedback"  TEXT NOT NULL,
    "resolved"  INTEGER NOT NULL DEFAULT 0,
    "bookmark_id" DEFAULT NULL,
    FOREIGN KEY("user_id") REFERENCES "users"("user_id")
  );
SQL
```

```
csvTaggedBookmarks.each do |row|
    DB.execute "insert into tagged_bookmarks values ( ?, ?)", row.fields
end

csvBookmarks.each do |row|
    DB.execute "insert into bookmarks values ( ?, ?, ?, ?, ?, ?, ?, ?, ?)", row.fields
end


csvBookmarkHistory.each do |row|
    DB.execute "insert into bookmark_history values ( ?, ?, ?)", row.fields
end

csvUsers.each do |row|
    DB.execute "insert into users values ( ?, ?, ?, ?, ?, ?, ?, ?,?)", row.fields
end

csvTags.each do |row|
    DB.execute "insert into tags values ( ?, ?)", row.fields
end

csvFeedback.each do |row|
    DB.execute "insert into feedback values (?, ?, ?, ?, ?, ?, ?)", row.fields
end

csvLoginHistory.each do |row|
    DB.execute "insert into login_history values ( ?, ?)", row.fields
end
```

```

csvFavourites.each do |row|
     DB.execute "insert into favourites values ( ?, ?)", row.fields
end

```

C. app/history.rb

```ruby
module History

    def History.new(user_id, bookmark_id, date)
        result = false
        insert = "INSERT INTO bookmark_history (user_id, bookmark_id, date) VALUES (?,?,?)"
        begin
            $db.execute insert, user_id, bookmark_id, date
            result = true
            puts "Insertion success"
        rescue SQLite3::ConstraintException
            puts "Insertion Failed"
        end
        return result
    end

    def History.get_last_visit_date(bookmark_id)
        result = Array.new
        query = "SELECT date FROM bookmark_history WHERE bookmark_id = ?;"
        result = $db.execute query, bookmark_id
        return result[result.length-1]
    end

    def History.update_date(bookmark_id, date)
        query = "UPDATE bookmark_history SET date = ? WHERE bookmark_id = ?;"
        begin
            $db.execute query, date, bookmark_id
            result = true
            puts "Last visit date updated"
        rescue SQLite3::ConstraintException
            puts "Insertion Failed"
        end
        return result
    end

end
```

D. app/signup.rb

```ruby
get '/signup' do
    Bookmark.getTags(2)
    erb :signup
end

post '/signup' do

    @first_name = params[:first_name].strip
    @last_name = params[:last_name].strip
    @username = params[:username].strip
    @email = params[:email].strip
    @confirmed_email = params[:email_confirm].strip
    @confirmed_password = params[:password_confirm].strip
    @password = params[:password].strip
    @email_used = User.find_email(@email)
    @username_used = User.find_username(@username)

    if @email_used
        flash[:info] = "Sorry, this email is already registered"
        redirect '/signup'
    elsif @username_used
        flash[:info] = "Sorry, this username is already used by a different user"
        redirect '/signup'
    elsif @email == @confirmed_email && @password == @confirmed_password
        User.new(@username, @first_name, @last_name, @password, @email)
        flash[:info] = "Your account is waiting for admin verification"
        redirect '/login'
    else
        redirect '/signup'
    end

end
```

E. views/boomark_table.erb

```
<table class="table">
  <thead>
    <tr>
        <th>No.</th>
        <th>Bookmark Name</th>
        <th>Link</th>
        <th>Rating</th>
        <th>Last Updated</th>
        <th>Status</th>
        <th>More Info</th>
    </tr>
  </thead>

  <tbody id="bookmark_table">

        <% @bookmark_list.each do |bookmark| %>
      <tr>

        <td><%= @bookmark_list.find_index(bookmark) + 1 %></td>
        <td><%= bookmark[1] %></td>

         <td>
            <form class="history" method="post" action="/add_to_history" target="_blank">
            <input type="hidden" name="bookmark_id" value="<%= bookmark[0] %>"/>
            <input class="btn btn-info btn-sm" type="submit" value="<%= Bookmark.getHost(bookmark[2]) %>" />
            <input type="hidden" name="bookmark_url" value="<%= bookmark[2] %>">
            </form>
        </td>
```

```

       <td>
          <form class="rating" method="post" action="/submit_rating" style="justify-content: left;">

             <input type="hidden" name="bookmark_id" value="<%= bookmark[0] %>">
             <input type="hidden" name="num_ratings" value="<%= bookmark[8] %>">
             <input type="hidden" name="current_rating" value="<%= bookmark[7] %>">
                <div class ="stars">
                   <% case bookmark[7].to_i when 0 %>
                      <button id="rating" type="submit" name="stars" value="5">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="4">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="3">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="2">&#9733;</button>
                      <button id="rating"type="submit" name="stars" value="1">&#9733;</button>
                   <% when 1 %>
                       <button id="rating" type="submit" name="stars" value="5">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="4">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="3">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="2">&#9733;</button>
                      <button id="rating"type="submit" name="stars" value="1" style="color: #fed136;">&#9733;</button>
                   <% when 2 %>
                      <button id="rating" type="submit" name="stars" value="5">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="4">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="3">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="2" style="color: #fed136;">&#9733;</button>
                      <button id="rating"type="submit" name="stars" value="1" style="color: #fed136;">&#9733;</button>
                   <% when 3 %>
                      <button id="rating" type="submit" name="stars" value="5">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="4">&#9733;</button>
                      <button id="rating" type="submit" name="stars" value="3" style="color: #fed136;">&#9733;</button>
```

```
 <td>
    <% case bookmark[6] when 0 %>
        <img src='images/red.png' class="icon" alt="broken link" />
    <% when 1 %>
        <img src='images/wrench.png' class="icon" alt="link under review"/>
    <% when 2 %>
        <img src='images/green.png' class="icon" alt="working link"/>
    <% else %>
    <% end %>
    <% case bookmark[6] when 2 %>
        <a style="color:grey; text-decoration:underline; font-size:13px" href='/report/<%= bookmark[0] %>'>Report</a>
    <% end %>
  </td>

  <td id="more_info">
      <a tabindex="0" class="btn btn-sm btn-info" role="button"  data-html="true" data-toggle="popover" data-trigger="focus"
        title="<%= bookmark[1] %>"
        data-content="<div><b>Boomark ID: <%= bookmark[0] %></b></div>
                      <div><b>Description: </b><%= bookmark[4] %> </div>
                      <div><b>Creator: </b><%= ($db.execute "SELECT email FROM users WHERE user_id = ?",bookmark[4])[0][0] %></div>
                      <div><b>Department tags: </b><%= Bookmark.getTags(bookmark[0]) %></div>
                      ">More Info</a>

  </td>
</tr>
 end %>
tbody>
e>
```

F. app/admin.rb

```ruby
get'/admin' do
   if !session[:admin]
      flash[:warning] = "You wandered into protected space, please login to continue"
       redirect '/login'
   else
    erb :admin_dashboard
   end
end

#get method for admin users, retrieves users from db and stores it in an array
get '/admin_users' do
   if !session[:admin]
       flash[:warning] = "You wandered into protected space, please login to continue"
       redirect '/login'
   else
     @user_list = $db.execute "SELECT * FROM users ORDER BY user_id ASC"
      @new_user_list = $db.execute "SELECT * FROM users WHERE new = 1 ORDER BY user_id ASC"
    erb :admin_user_mgmt
   end
end

#get method for admin bookmarks, retrieves bookmarks from db and stores it in an array
get '/admin_bookmarks' do
   if !session[:admin]
      flash[:warning] = "You wandered into protected space, please login to continue"
       redirect '/login'
   else
       @admin_bookmark_list = $db.execute "SELECT * FROM bookmarks ORDER BY bookmark_id ASC"
       erb :admin_bookmark_mgmt
   end
end
```

```ruby
#get method for feedback, retrieves feedback from db and stores it in an array
get '/admin_feedback' do
   if !session[:admin]
      flash[:warning] = "You wandered into protected space, please login to continue"
      redirect '/login'
   else
      @admin_feedback_list = $db.execute "SELECT * FROM feedback ORDER BY feedback_id ASC"
      erb :admin_feedback
   end
end

#get method to edit specific user
get '/admin_edit_user/:id' do
    if !session[:admin]
       flash[:warning] = "You wandered into protected space, please login to continue"
       redirect '/login'
    end

    redirect '/index' if !session[:admin]

    session[:editing_id] = params["id"]

    redirect '/admin_edit_user'
end

get '/admin_edit_user' do
    if !session[:admin]
        flash[:warning] = "You wandered into protected space, please login to continue"
        redirect '/login'
    else
        @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
```