Exemplar code Iteration 1

A. login.rb

```ruby
VALID_EMAIL_REGEX = /\A([\w+\-].?)+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i

#get logout request and redirect to index page
get '/logout' do
    session.clear
   flash[:info] = "You have been successfully logged out"
    redirect '/index'
end


#get login request and redirect to login page
get '/login' do
    redirect '/index' if session[:logged_in]
    erb :login
end

post '/login' do
    session.clear
    @login = true


    @email = params[:email]
    @pass = params[:password]
    @userFound = User.find_user(@email, @pass)

    @email_ok = !@email.nil? && @email != "" && @email =~ VALID_EMAIL_REGEX
    @pass_ok = !@pass.nil? && @pass != ""
    @all_ok = @email_ok && @pass_ok

    if @userFound
    #Gather all user data
    query = "SELECT * FROM users WHERE email = ?;"
    @userInfo = $db.execute query, @email
    @active = @userInfo[0][5]
        puts @active
    end

    if  @active == 1
        session[:logged_in] = true #User now logged in
        session[:admin] = false #User is not an admin until confirmed

    #   store user info in session data
        session[:email] = @userInfo[0][4]
        session[:user_id] = @userInfo[0][0]
        session[:first_name] = @userInfo[0][1]

        if @userInfo[0][6] == 1
            session[:admin] = true #User is an admin
            redirect'/admin'
        end

        flash[:notice] = "You have successfully signed in"
        redirect '/index'
    else
        session[:logged_in] = false
        erb :login
    end

end
```

B. user.rb

```ruby
module User

    def User.find_user(email, pass)
       result = false
       query = "SELECT password FROM users WHERE email = ?;"
       rows = $db.execute query, email
       if (rows.count == 1)
           hash_check = BCrypt::Password.new(rows[0][0])
           result = hash_check.is_password?(pass)
       end
       return result
    end


    def User.logged_in?(session)      #not sure if we need separate function for this
        session.nil? ? false : true   #we can verify if user is logged using function find_user
    end

    def User.new(user_id, first_name, last_name, email, password)
        result = false
        query = "INSERT INTO users (user_id, first_name, last_name, password, email, active_status,admin) VALUES(?, ?, ?, ?, ?,?,?);"
        begin
            $db.execute query, user_id, first_name, last_name, BCrypt::Password.create(password),email,0,0
            result = true
        rescue SQLite3::ConstraintException
            puts "Insertion Failed"
        end
        return result
    end

end
```

C. 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) VALUES (?,?,?,?,?,?)"
        begin
            $db.execute insert, bookmark_name, link, description, creator, last_updated, 0
            result = true
            puts "Insertion success"
        rescue SQLite3::ConstraintException
            puts "Insertion Failed"
        end
        return result
    end

    def Bookmark.find_by(search)
        result[]
        db = Sqlite::Database.new '../database/acme_db.sqlite'
        query = "SELECT"
    end

end
```

D. new_bookmark.rb

```ruby
get '/new_bookmark' do
    redirect '/index' if !session[:logged_in]
    erb :new_bookmark
end

post '/new_Bookmark' do
    redirect '/index' unless session[:logged_in]

    #date-time format (YYYY/MM/DD HH:MM)
    @last_updated = Time.now.strftime("%Y/%m/%d %H:%M").to_s

    @bookmark_name = params[:bookmark_name]
    @link = params[:link]
    @description = params[:description]
    @report_status=0

    Bookmark.new(@bookmark_name, @link, @description, session[:user_id], @last_updated)

#     add_bookmark = "INSERT INTO bookmarks (bookmark_name, link, description, creator, last_updated) VALUES (?,?,?,?,?)"

#     $db.execute add_bookmark, params[:bookmark_name], params[:link], params[:description], session[:user_id], @last_updated

    redirect '/'
end
```

E. bookmark_table.erb

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

  <tbody>

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

        <td><%= bookmark[0] %></td>
        <td><%= bookmark[1] %></td>
        <td><a class="btn btn-info btn-xs" href="<%= bookmark[2] %>">Link</a></td>
        <td><%= ($db.execute "SELECT email FROM users WHERE user_id = ?",bookmark[4])[0][0] %></td>
        <td><%= bookmark[5] %></td>
        <td><%= bookmark[6] %></td>
        <td><a class="btn btn-info btn-xs" href="<%= bookmark[3] %>"> More Info</a></td>

      </tr>
    <% end %>
    </tbody>
</table>
```

F. createDatabase.rb

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

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 0,
    "rating"    INTEGER DEFAULT 0,
    "num_ratings"   INTEGER DEFAULT 0
);
SQL

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

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

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



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
```