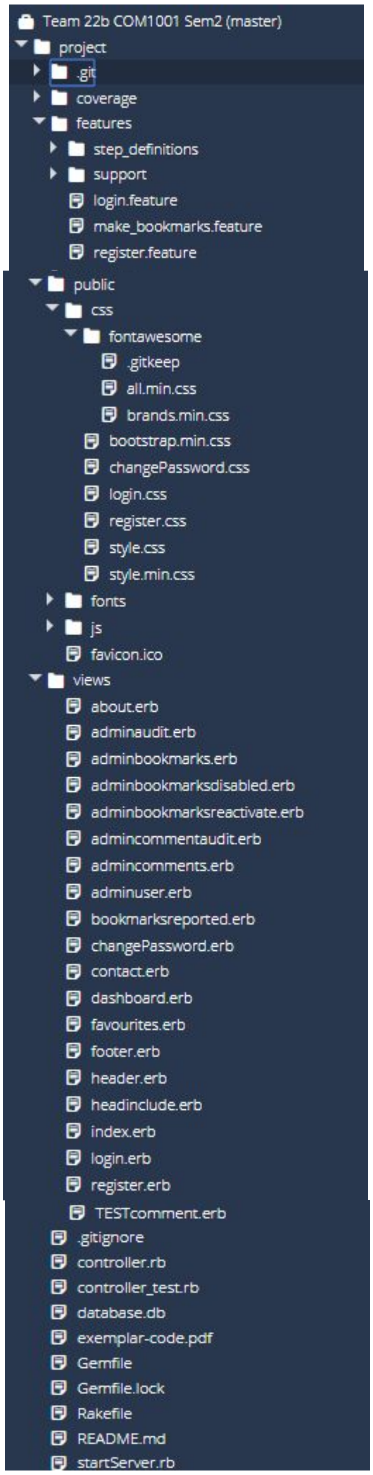
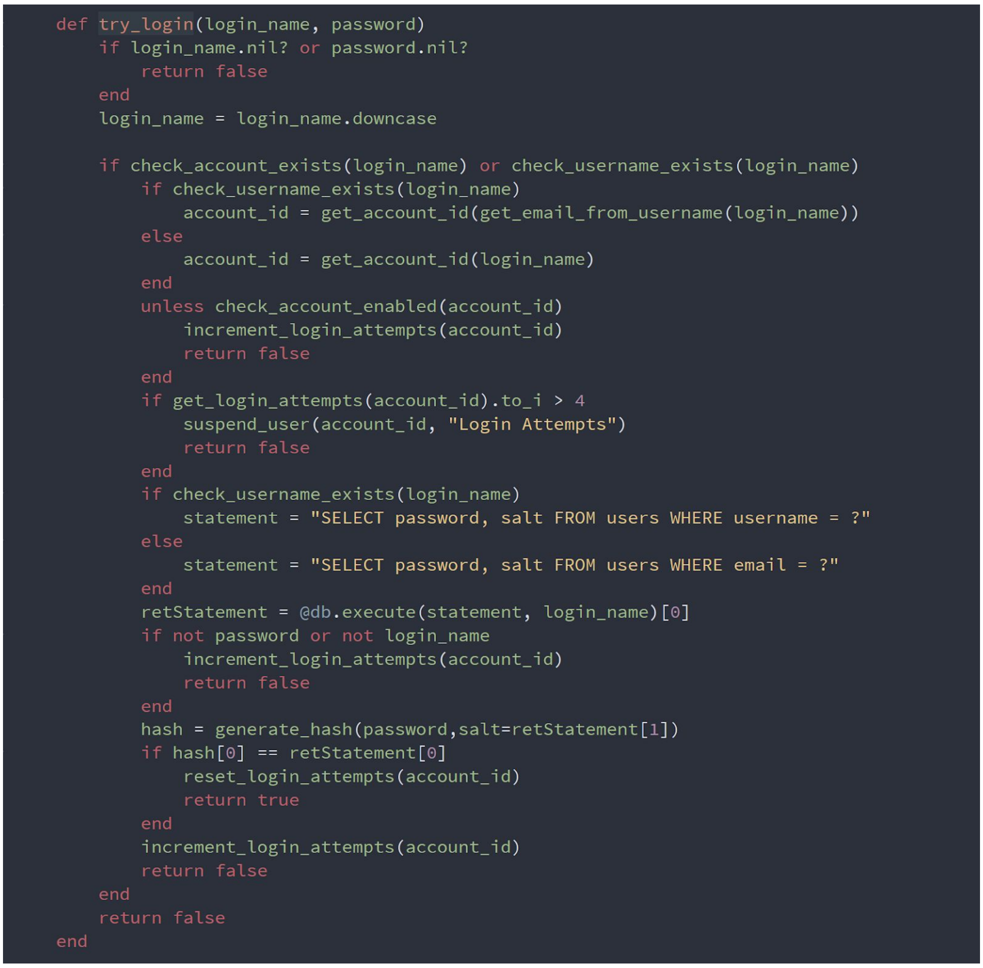


```
Team 22b COM1001 Sem2 (master)
  project
    .git
    coverage
    features
      step_definitions
      support
      login.feature
      make_bookmarks.feature
      register.feature
    public
      css
        fontawesome
          .gitkeep
          all.min.css
          brands.min.css
        bootstrap.min.css
        changePassword.css
        login.css
        register.css
        style.css
        style.min.css
      fonts
      js
      favicon.ico
    views
      about.erb
      adminaudit.erb
      adminbookmarks.erb
      adminbookmarksdisabled.erb
      adminbookmarksreactivate.erb
      admincommentaudit.erb
      admincomments.erb
      adminuser.erb
      bookmarksreported.erb
      changePassword.erb
      contact.erb
      dashboard.erb
      favourites.erb
      footer.erb
      header.erb
      headinclude.erb
      index.erb
      login.erb
      register.erb
      TESTcomment.erb
    .gitignore
    controller.rb
    controller_test.rb
    database.db
    exemplar-code.pdf
    Gemfile
    Gemfile.lock
    Rakefile
    README.md
    startServer.rb
```

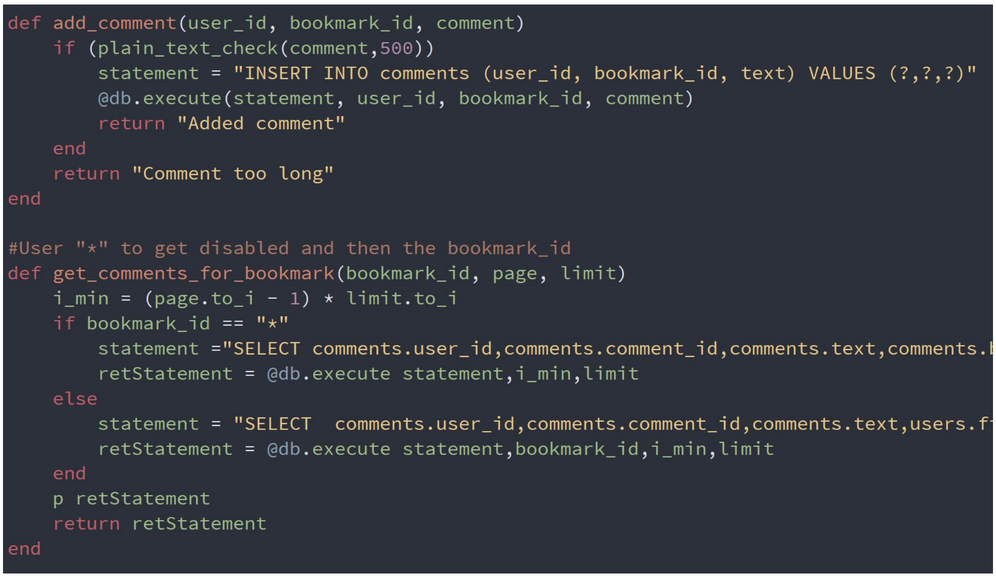
Listing A (above) - This is the File Structure for our project



```
    def try_login(login_name, password)
        if login_name.nil? or password.nil?
            return false
        end
        login_name = login_name.downcase

        if check_account_exists(login_name) or check_username_exists(login_name)
            if check_username_exists(login_name)
                account_id = get_account_id(get_email_from_username(login_name))
            else
                account_id = get_account_id(login_name)
            end
            unless check_account_enabled(account_id)
                increment_login_attempts(account_id)
                return false
            end
            if get_login_attempts(account_id).to_i > 4
                suspend_user(account_id, "Login Attempts")
                return false
            end
            if check_username_exists(login_name)
                statement = "SELECT password, salt FROM users WHERE username = ?"
            else
                statement = "SELECT password, salt FROM users WHERE email = ?"
            end
            retStatement = @db.execute(statement, login_name)[0]
            if not password or not login_name
                increment_login_attempts(account_id)
                return false
            end
            hash = generate_hash(password,salt=retStatement[1])
            if hash[0] == retStatement[0]
                reset_login_attempts(account_id)
                return true
            end
            increment_login_attempts(account_id)
            return false
        end
        return false
    end
```

Listing B (above) - controller.rb - Login

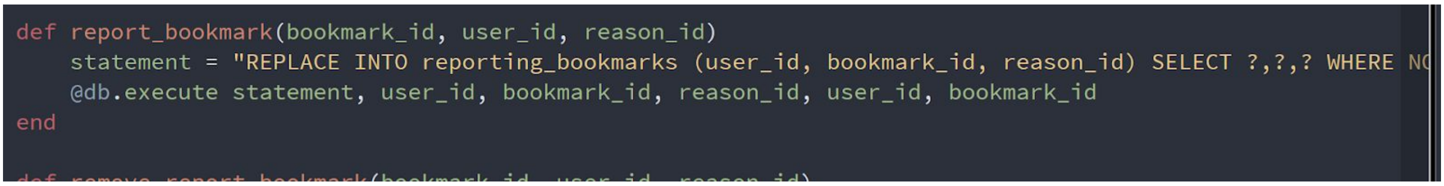
This function tests if the username exists or the email exists based on a string the user posts to the server, if it does then it tries logging in with the password that they provide. There is some validation on the passwords as well as the login name to avoid errors.



```
def add_comment(user_id, bookmark_id, comment)
    if (plain_text_check(comment,500))
        statement = "INSERT INTO comments (user_id, bookmark_id, text) VALUES (?,?,?)"
        @db.execute(statement, user_id, bookmark_id, comment)
        return "Added comment"
    end
    return "Comment too long"
end

#User "*" to get disabled and then the bookmark_id
def get_comments_for_bookmark(bookmark_id, page, limit)
    i_min = (page.to_i - 1) * limit.to_i
    if bookmark_id == "*"
        statement ="SELECT comments.user_id,comments.comment_id,comments.text,comments.
        retStatement = @db.execute statement,i_min,limit
    else
        statement = "SELECT  comments.user_id,comments.comment_id,comments.text,users.f
        retStatement = @db.execute statement,bookmark_id,i_min,limit
    end
    p retStatement
    return retStatement
end
```

Listing C (right) - controller.rb - Commenting

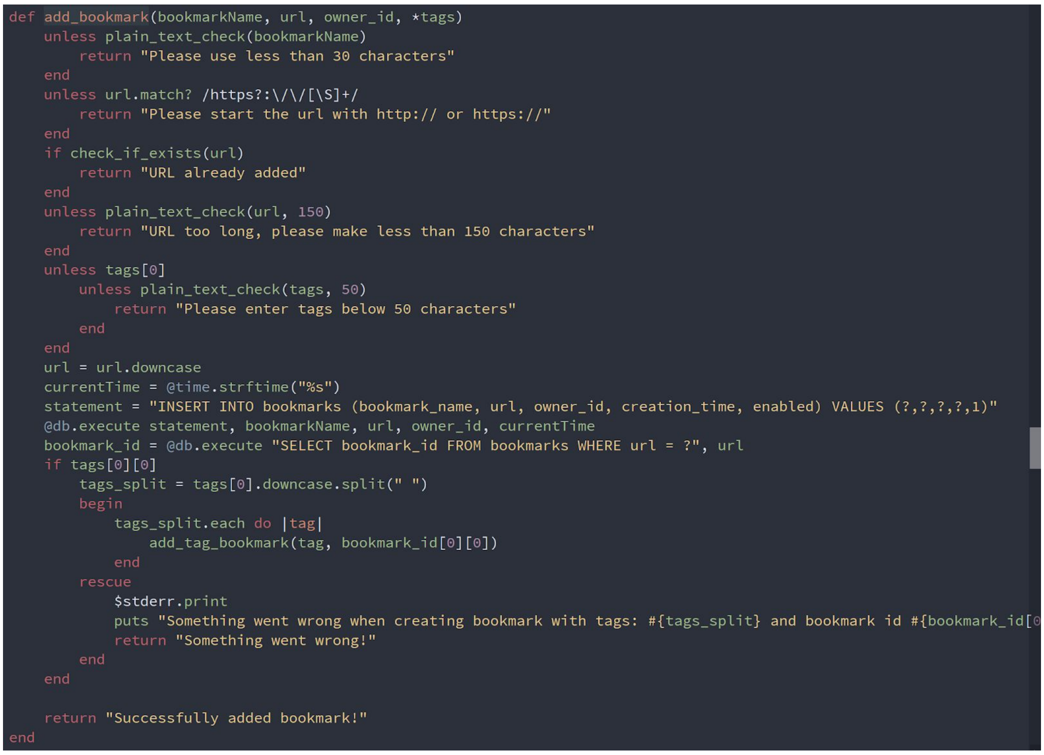
These functions add a comment to the database if it's not too long, then the get_comments_for_bookmark function acquires all of the comments for a specific bookmark based on the bookmarks id, this also then is filtered down so it can display x amount per page.



```
def report_bookmark(bookmark_id, user_id, reason_id)
    statement = "REPLACE INTO reporting_bookmarks (user_id, bookmark_id, reason_id) SELECT ?,?,? WHERE NO
    @db.execute statement, user_id, bookmark_id, reason_id, user_id, bookmark_id
end

def remove_report_bookmark(bookmark_id, user_id, reason_id)
```

Listing D (above) - controller.rb - Report

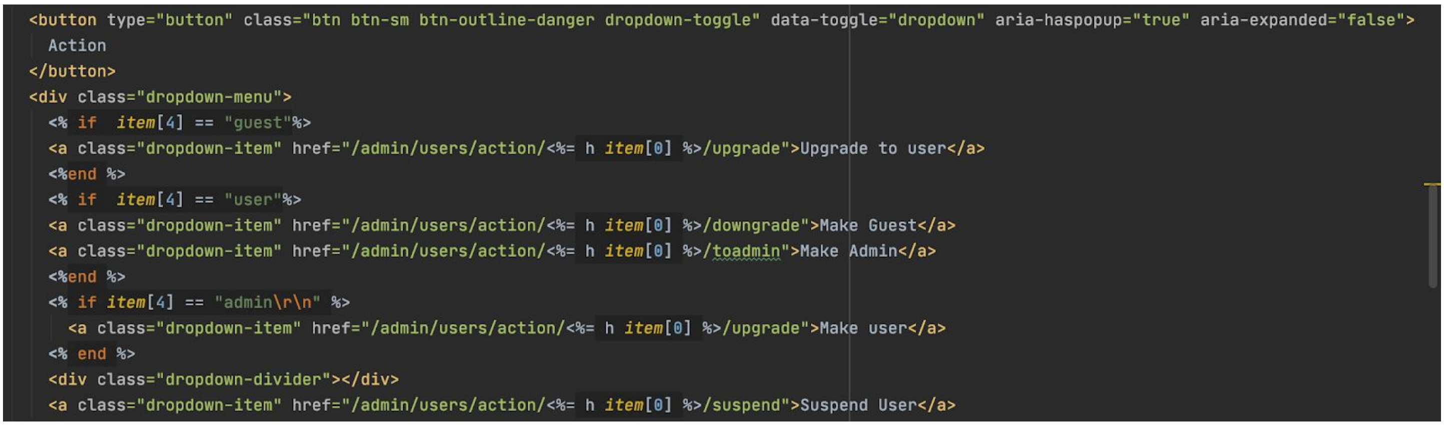
This function adds an entry if not exists into the reporting_bookmarks table to show that a user has reported a bookmark



```
def add_bookmark(bookmarkName, url, owner_id, *tags)
    unless plain_text_check(bookmarkName)
        return "Please use less than 30 characters"
    end
    unless url.match? /https?:\/\/[\S]+/
        return "Please start the url with http:// or https://"
    end
    if check_if_exists(url)
        return "URL already added"
    end
    unless plain_text_check(url, 150)
        return "URL too long, please make less than 150 characters"
    end
    unless tags[0]
        unless plain_text_check(tags, 50)
            return "Please enter tags below 50 characters"
        end
    end
    url = url.downcase
    currentTime = @time.strftime("%s")
    statement = "INSERT INTO bookmarks (bookmark_name, url, owner_id, creation_time, enabled) VALUES (?,?,?,?,1)"
    @db.execute statement, bookmarkName, url, owner_id, currentTime
    bookmark_id = @db.execute "SELECT bookmark_id FROM bookmarks WHERE url = ?", url
    if tags[0][0]
        tags_split = tags[0].downcase.split(" ")
        begin
            tags_split.each do |tag|
                add_tag_bookmark(tag, bookmark_id[0][0])
            end
        rescue
            $stderr.print
            puts "Something went wrong when creating bookmark with tags: #{tags_split} and bookmark id #{bookmark_id[0
            return "Something went wrong!"
        end
    end

    return "Successfully added bookmark!"
end
```

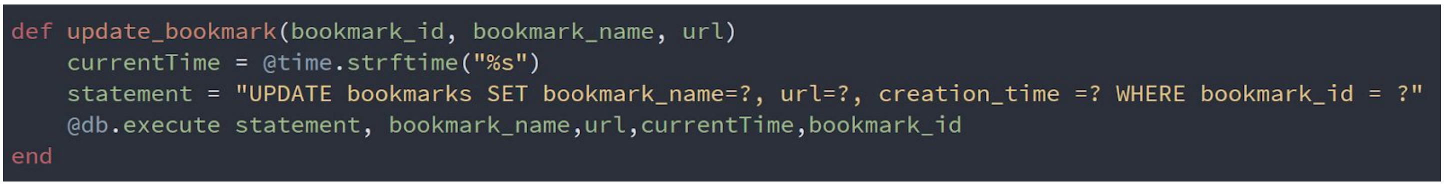
Listing E (left) - controller.rb - Add bookmark

This function adds a bookmark to the bookmarks table and also iterates over the tags for the bookmarks to add them to the corresponding bookmark_tags table. This doesn't add the tag if it already exists.



```
<button type="button" class="btn btn-sm btn-outline-danger dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
  Action
</button>
<div class="dropdown-menu">
  <% if  item[4] == "guest"%>
  <a class="dropdown-item" href="/admin/users/action/<%= h item[0] %>/upgrade">Upgrade to user</a>
  <%end %>
  <% if  item[4] == "user"%>
  <a class="dropdown-item" href="/admin/users/action/<%= h item[0] %>/downgrade">Make Guest</a>
  <a class="dropdown-item" href="/admin/users/action/<%= h item[0] %>/toadmin">Make Admin</a>
  <%end %>
  <% if item[4] == "admin\r\n" %>
    <a class="dropdown-item" href="/admin/users/action/<%= h item[0] %>/upgrade">Make user</a>
  <% end %>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="/admin/users/action/<%= h item[0] %>/suspend">Suspend User</a>
```

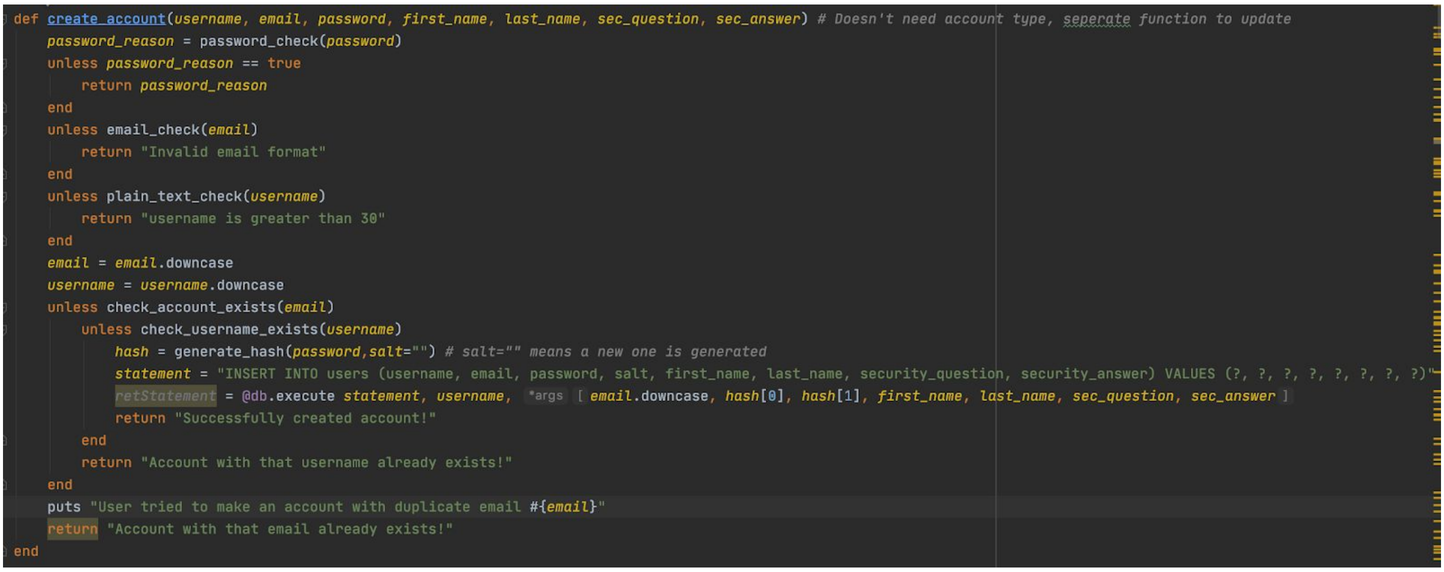
Listing F (above)- adminuser.erb - Admin Panel - Shows the button for each possible action based on the users roles -



```
def update_bookmark(bookmark_id, bookmark_name, url)
    currentTime = @time.strftime("%s")
    statement = "UPDATE bookmarks SET bookmark_name=?, url=?, creation_time =? WHERE bookmark_id = ?"
    @db.execute statement, bookmark_name,url,currentTime,bookmark_id
end
```

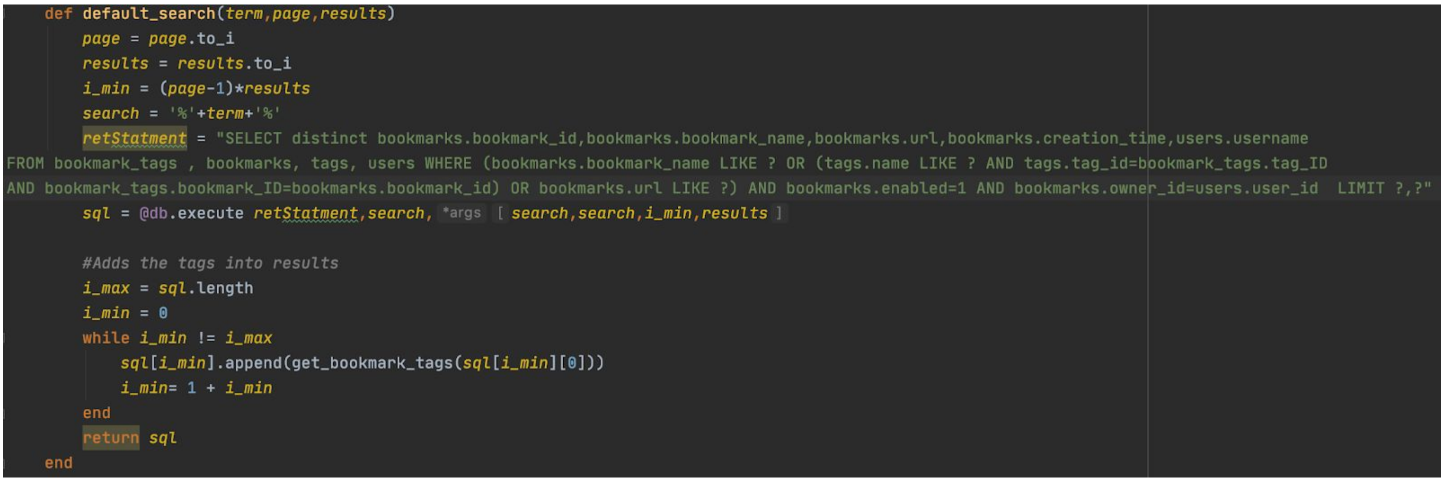
Listing G (above) - controller.rb - Edit bookmark

Simply takes in the id name and url to update the bookmark in the bookmarks table



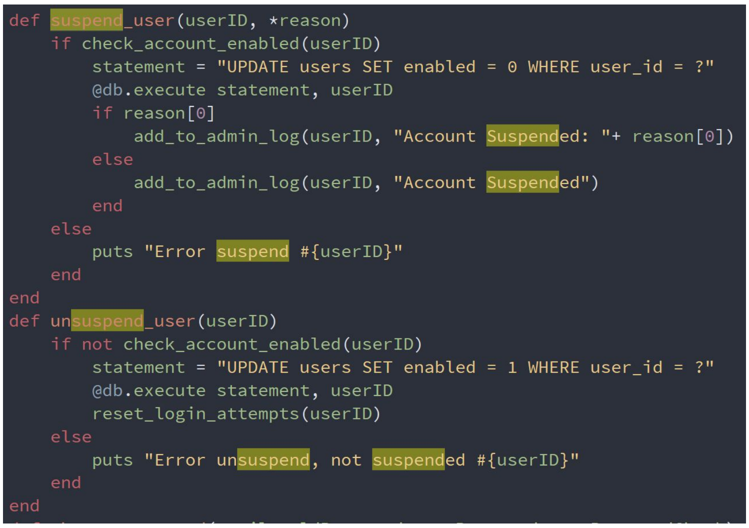
```ruby
def create_account(username, email, password, first_name, last_name, sec_question, sec_answer) # Doesn't need account type, seperate function to update
    password_reason = password_check(password)
    unless password_reason == true
        return password_reason
    end
    unless email_check(email)
        return "Invalid email format"
    end
    unless plain_text_check(username)
        return "username is greater than 30"
    end
    email = email.downcase
    username = username.downcase
    unless check_account_exists(email)
        unless check_username_exists(username)
            hash = generate_hash(password,salt="") # salt="" means a new one is generated
            statement = "INSERT INTO users (username, email, password, salt, first_name, last_name, security_question, security_answer) VALUES (?, ?, ?, ?, ?, ?, ?, ?)"
            retStatement = @db.execute statement, username, *args [ email.downcase, hash[0], hash[1], first_name, last_name, sec_question, sec_answer ]
            return "Successfully created account!"
        end
        return "Account with that username already exists!"
    end
    puts "User tried to make an account with duplicate email #{email}"
    return "Account with that email already exists!"
end
```

<u>Listing H </u>(above) - controller.rb - Create account function - with validation to ensure nothing unwanted is pushed to db.



```ruby
    def default_search(term,page,results)
        page = page.to_i
        results = results.to_i
        i_min = (page-1)*results
        search = '%'+term+'%'
        retStatment = "SELECT distinct bookmarks.bookmark_id,bookmarks.bookmark_name,bookmarks.url,bookmarks.creation_time,users.username
FROM bookmark_tags , bookmarks, tags, users WHERE (bookmarks.bookmark_name LIKE ? OR (tags.name LIKE ? AND tags.tag_id=bookmark_tags.tag_ID
AND bookmark_tags.bookmark_ID=bookmarks.bookmark_id) OR bookmarks.url LIKE ?) AND bookmarks.enabled=1 AND bookmarks.owner_id=users.user_id  LIMIT ?,?"
        sql = @db.execute retStatment,search, *args [ search,search,i_min,results ]

        #Adds the tags into results
        i_max = sql.length
        i_min = 0
        while i_min != i_max
            sql[i_min].append(get_bookmark_tags(sql[i_min][0]))
            i_min= 1 + i_min
        end
        return sql
    end
```

<u>Listing I </u>(above) - controller.rb - search function showing all results as well as appending all tags into result - controller.rb



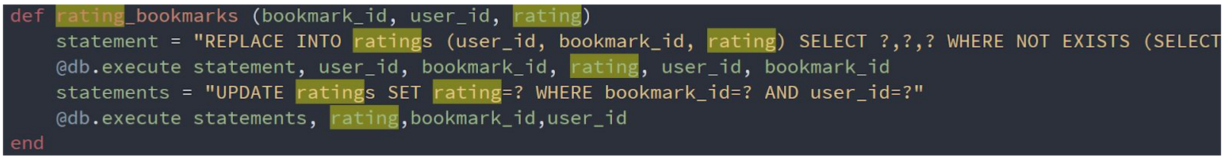
```ruby
def suspend_user(userID, *reason)
    if check_account_enabled(userID)
        statement = "UPDATE users SET enabled = 0 WHERE user_id = ?"
        @db.execute statement, userID
        if reason[0]
            add_to_admin_log(userID, "Account Suspended: "+ reason[0])
        else
            add_to_admin_log(userID, "Account Suspended")
        end
    else
        puts "Error suspend #{userID}"
    end
end
def unsuspend_user(userID)
    if not check_account_enabled(userID)
        statement = "UPDATE users SET enabled = 1 WHERE user_id = ?"
        @db.execute statement, userID
        reset_login_attempts(userID)
    else
        puts "Error unsuspend, not suspended #{userID}"
    end
end
```

<u>Listing J </u>- controller.rb - Suspend/unsuspend user -

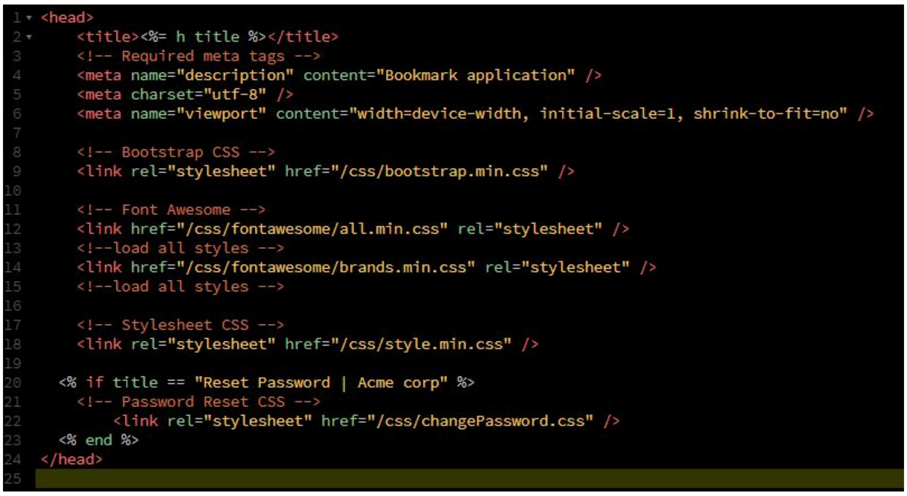
These functions change the enabled field in the users table based on the user's id and then places a log into the audit log so that admins can see what has been done

<u>Listing K</u> (below)- - Rating -

This function adds a rating element if not exists into the table and then sets the rating for the bookmark from the user into the database, this is then queried later on in the code.

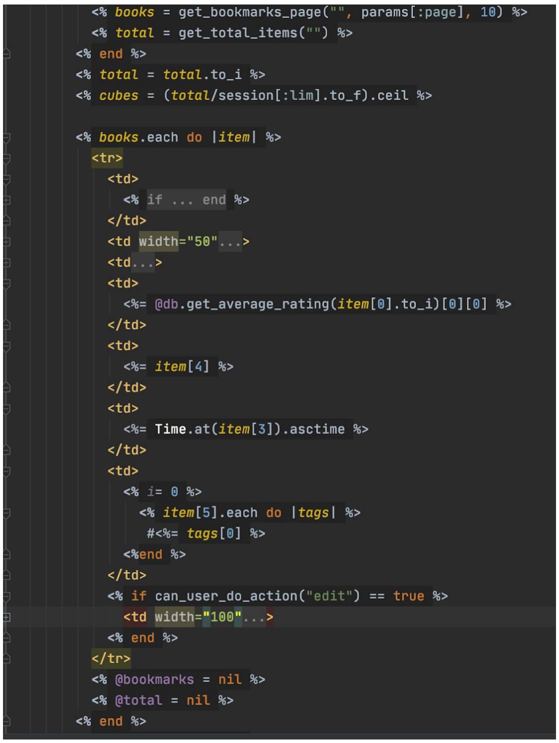


```ruby
def rating_bookmarks (bookmark_id, user_id, rating)
    statement = "REPLACE INTO ratings (user_id, bookmark_id, rating) SELECT ?,?,? WHERE NOT EXISTS (SELECT
    @db.execute statement, user_id, bookmark_id, rating, user_id, bookmark_id
    statements = "UPDATE ratings SET rating=? WHERE bookmark_id=? AND user_id=?"
    @db.execute statements, rating,bookmark_id,user_id
end
```



```
<head>
    <title><%= h title %></title>
    <!-- Required meta tags -->
    <meta name="description" content="Bookmark application" />
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="/css/bootstrap.min.css" />

    <!-- Font Awesome -->
    <link href="/css/fontawesome/all.min.css" rel="stylesheet" />
    <!--load all styles -->
    <link href="/css/fontawesome/brands.min.css" rel="stylesheet" />
    <!--load all styles -->

    <!-- Stylesheet CSS -->
    <link rel="stylesheet" href="/css/style.min.css" />

  <% if title == "Reset Password | Acme corp" %>
    <!-- Password Reset CSS -->
        <link rel="stylesheet" href="/css/changePassword.css" />
  <% end %>
</head>

```

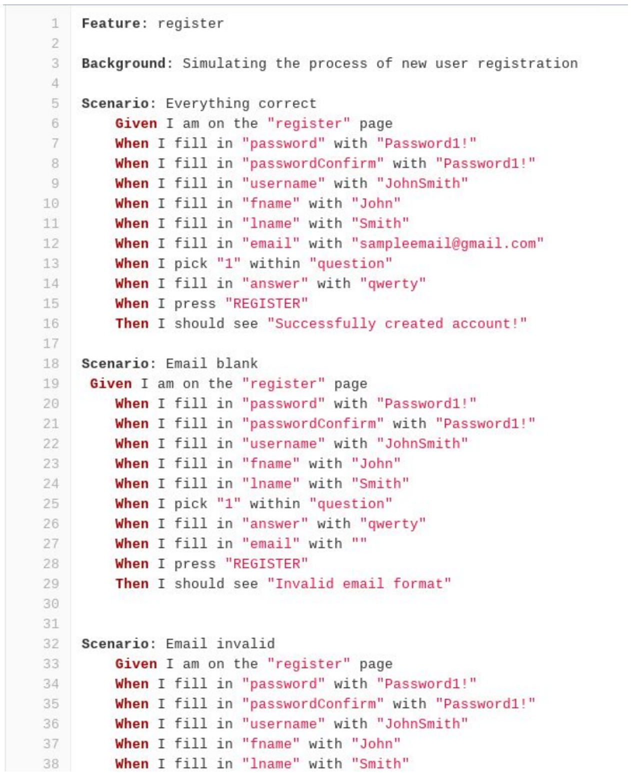
<u>Listing L</u> (left) - headinclude.erb - This is an erb that contains what would be in the header of most of our erbs, this is so that there is less code duplication.



```
      <% books = get_bookmarks_page("", params[:page], 10) %>
      <% total = get_total_items("") %>
    <% end %>
    <% total = total.to_i %>
    <% cubes = (total/session[:lim].to_f).ceil %>

    <% books.each do |item| %>
      <tr>
        <td>
          <% if ... end %>
        </td>
        <td width="50"...>
        <td...>
        <td>
          <%= @db.get_average_rating(item[0].to_i)[0][0] %>
        </td>
        <td>
          <%= item[4] %>
        </td>
        <td>
          <%= Time.at(item[3]).asctime %>
        </td>
        <td>
          <% i= 0 %>
            <% item[5].each do |tags| %>
             #<%= tags[0] %>
          <%end %>
        </td>
        <% if can_user_do_action("edit") == true %>
          <td width="100"...>
        <% end %>
      </tr>
      <% @bookmarks = nil %>
      <% @total = nil %>
    <% end %>
```

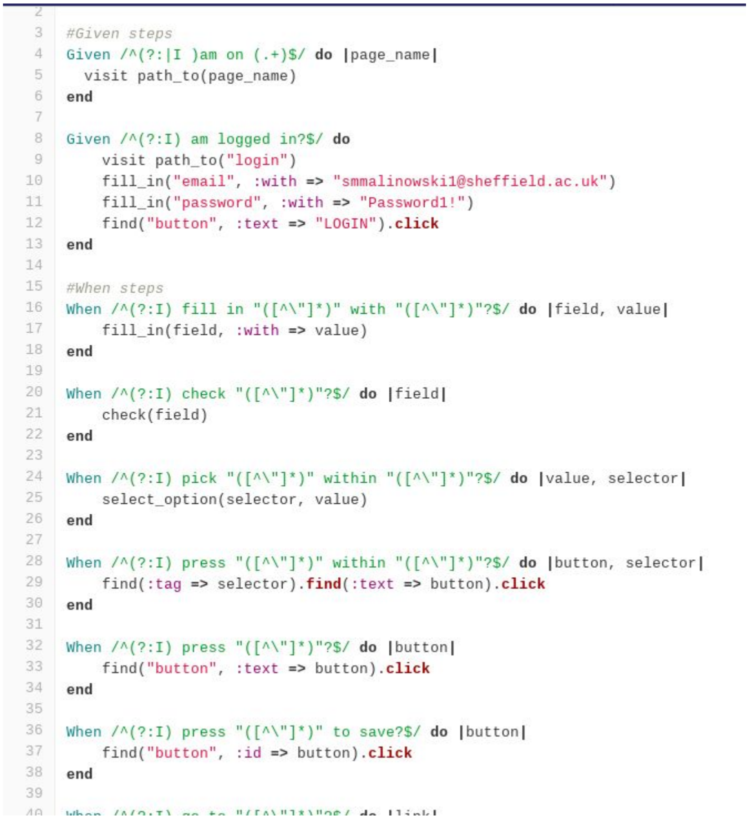
<u>Listing</u> (right) - dashboard.erb - the dashboard - compressed code showing the iterations of each bookmark listing

<u>Testing</u>



```
Feature: register

Background: Simulating the process of new user registration

Scenario: Everything correct
    Given I am on the "register" page
    When I fill in "password" with "Password1!"
    When I fill in "passwordConfirm" with "Password1!"
    When I fill in "username" with "JohnSmith"
    When I fill in "fname" with "John"
    When I fill in "lname" with "Smith"
    When I fill in "email" with "sampleemail@gmail.com"
    When I pick "1" within "question"
    When I fill in "answer" with "qwerty"
    When I press "REGISTER"
    Then I should see "Successfully created account!"

Scenario: Email blank
 Given I am on the "register" page
    When I fill in "password" with "Password1!"
    When I fill in "passwordConfirm" with "Password1!"
    When I fill in "username" with "JohnSmith"
    When I fill in "fname" with "John"
    When I fill in "lname" with "Smith"
    When I pick "1" within "question"
    When I fill in "answer" with "qwerty"
    When I fill in "email" with ""
    When I press "REGISTER"
    Then I should see "Invalid email format"


Scenario: Email invalid
    Given I am on the "register" page
    When I fill in "password" with "Password1!"
    When I fill in "passwordConfirm" with "Password1!"
    When I fill in "username" with "JohnSmith"
    When I fill in "fname" with "John"
    When I fill in "lname" with "Smith"
```

<u>Listing</u> (above) - features/register.feature - Gherkin code for testing the registration process.



```

#Given steps
Given /^(?:|I )am on (.+)$/ do |page_name|
  visit path_to(page_name)
end

Given /^(?:I) am logged in?$/ do
    visit path_to("login")
    fill_in("email", :with => "smmalinowski1@sheffield.ac.uk")
    fill_in("password", :with => "Password1!")
    find("button", :text => "LOGIN").click
end

#When steps
When /^(?:I) fill in "([^\"]*)" with "([^\"]*)"?$/ do |field, value|
    fill_in(field, :with => value)
end

When /^(?:I) check "([^\"]*)"?$/ do |field|
    check(field)
end

When /^(?:I) pick "([^\"]*)" within "([^\"]*)"?$/ do |value, selector|
    select_option(selector, value)
end

When /^(?:I) press "([^\"]*)" within "([^\"]*)"?$/ do |button, selector|
    find(:tag => selector).find(:text => button).click
end

When /^(?:I) press "([^\"]*)"?$/ do |button|
    find("button", :text => button).click
end

When /^(?:I) press "([^\"]*)" to save?$/ do |button|
    find("button", :id => button).click
end

When /^(?:I) go to "([^\"]*)"?$/ do |link|
```

<u>Listing</u> (right) - features/ step_definitions/web_steps.rb - Ruby code for defining the steps to be used by each feature