```ruby
get '/signup' do
    erb :"Signup/index"
end

#submitting the signup
post '/signup' do

    @password = params[:password]
    @repassword = params[:repassword]
    @username = params[:username]
    @reason = params[:reason]

    #validation checks for the sign up
    if @password != @repassword then
        @signupError = "Passwords don't match"

    elsif User.getByUsername(@username) != nil then
        @signupError = "Username already exists"

    else
      #creating the user
        if User.newUser(@username, @password, 0, 1)


            newAccount = User.getByUsername(@username)
            if SignupRequest.newRequest(newAccount.userId, @reason)
                @signupSuccess = "Account created and sent for approval"
            else
                @signupError = "Sign up request not logged"
            end
        else
            @signupError = "Account unable to be created"
        end
    end

    erb :"Signup/index"

end
```

Controllers/Signup.rb

```ruby
#adding ratings to the bookmarks
post '/ratings/add' do

    #checking the rate has a value
    if params[:rate] == nil then
      redirect "/bookmarks/view/#{params[:bookmarkId]}"
    end

    #adding a rate to the bookmark
    rate = Integer(params[:rate])
    if session[:loggedIn] != true then
      @ratingError = "User not logged in"
      redirect "/bookmarks/view/#{params[:bookmarkId]}"
    else
      result = Rating.newRating(params[:bookmarkId], session[:userId], rate)

      if result then
          redirect "/bookmarks/view/#{params[:bookmarkId]}"
      else
          status 500
      end
    end

end
```

Controllers/Ratings.rb

```ruby
    #searching function for the bookmarks
    def self.searchBy(searchTerm, searchType)
        toReturn = []

        #changing the query dependant on the search
        if searchType == "title" then
            query = "SELECT * FROM bookmarks WHERE title LIKE ? ;"
            result = DB.execute query, "%" + searchTerm + "%"
        elsif searchType == "resource" then
            query = "SELECT * FROM bookmarks WHERE resource LIKE ? ;"
            result = DB.execute query, "%" + searchTerm + "%"

        elsif searchType == "bookmarkId" then
            query = "SELECT * FROM bookmarks WHERE bookmarkId LIKE ? ;"
            result = DB.execute query, searchTerm
        elsif searchType == "userId" then
            query = "SELECT * FROM bookmarks WHERE userId LIKE ? ;"
            result = DB.execute query, searchTerm
        end

        for bookmark in result do
            bookmarkObj = Bookmark.new(bookmark[0], bookmark[1], bookmark[2], bookmark[3], bookmark[4], bookmark[5], bookmark[6])
            toReturn.push(bookmarkObj)
        end

        return toReturn
    end
```

Models/Bookmark.rb

```ruby
    def self.newFavourite(bookmarkId, userId)

        query =  "INSERT INTO favourites('bookmarkId', 'userId') VALUES(?,?);"

        begin
            DB.execute query, bookmarkId, userId
        rescue SQLite3::Exception
            return false
        end

        return true
    end

    # Remove a favourite between a bookmark and a userId
    # Returns: if operation was successful
    def self.removeFavourite(bookmarkId, userId)

        query =  "DELETE FROM favourites WHERE bookmarkId=? AND userId=?;"

        begin
            DB.execute query, bookmarkId, userId
        rescue SQLite3::Exception
            return false
        end

        return true
    end

    #checking if the bookmark is a favowrite for that user
    def self.isFavourite(bookmarkId, userId)

        query = "SELECT * FROM favourites WHERE bookmarkId=? AND userId=?;"

        result = DB.execute query, bookmarkId, userId

        if result.length == 0 then
            return false
        else
            return true
        end

    end
```

Models/Favourite.rb

```ruby
    def self.doesTagExist(tag)

        query = "SELECT * FROM tags WHERE tag=?;"

        result = DB.execute query, tag

        if result == "" || result == nil || result[0] == nil then
            return nil
        else
            return result[0][0]
        end

    end

    # Verifies if a link exists between a tag and a bookmark
    # Returns bool defining if link exists
    def self.doesLinkExist(bookmarkId, tagId)

        query = "SELECT * FROM bookmarks_to_tags WHERE bookmarkId = ? AND tagId = ?;"

        result = DB.execute query, bookmarkId, tagId

        if result == "" || result == nil || result[0] == nil then
            return false
        else
            return true
        end

    end

    def self.trimTagArray(array, maxLength)

        if array.length > maxLength then
            array = array.first(maxLength)
        end

        return array
    end
```

Models/Tag.rb

```css
.rating:not(:checked) > input {
    position:absolute;
    top:-9999px;
}
.rating:not(:checked) > label {
    float:right;
    width:1em;
    cursor:pointer;
    font-size:1.75em;
    color:lightgray;
}
.rating:not(:checked) > label:before {
    content: '★ ';
}
.rating > input:checked ~ label {
    color: #fad728;
}
.rating:not(:checked) > label:hover,
.rating:not(:checked) > label:hover ~ label {
    color: #e8b910;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: black;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
}
.dropdown-content a {
    color: white;
    padding: 12px;
    text-decoration: none;
    display: block;
}
.dropdown:hover .dropdown-content {
    display: block;
}
```

Public/css/style.css

```
    #Adds bookmarks into the database, one that respects the field requiremenets and one that is "archived" and does not have a valid URL
    #Should return true for the first bookmark because it respects the requiremenets and false for the second one
    def test_new_bookmark

        resultOne = Bookmark.newBookmark("Hello", "Hello descriptiption", "https://google.co.uk/", 1, 1);

        assert_equal true, resultOne

        resultTwo = Bookmark.newBookmark("Duplicate resource", "Duplicate description", "something", 0, 1);

        assert_equal false, resultTwo

    end

    #Gets a bookmark that has ID = 1 and a bookmark that has ID = -1
    #Should return true for all the tests made for ID = 1 and ID = -1, because there does not exist a bookmark with a negative ID
    def test_get_by_id

        bookmarkOne = Bookmark.getById(1)

        assert_equal '18 March', bookmarkOne.createdAt
        assert_equal 'apple', bookmarkOne.title
        assert_equal 'something', bookmarkOne.description
        assert_equal 'something', bookmarkOne.resource
        assert_equal true, bookmarkOne.archived

        bookmarkNegative = Bookmark.getById(-1)

        assert_nil bookmarkNegative

    end
```

Tests/Bookmark_test.rb

```
<%= erb :"Shared/header" %>

<h1>All bookmarks</h1>
<hr />
<% if @bookmarks.length > 0 then %>
<div class="container">
    <table>
        <tr>
            <th>Count</th>
            <th>Title</th>
            <th>Description</th>
            <th>Rating</th>
            <th>View</th>
        </tr>
        <% for i in (0..(@bookmarks.length - 1)) %>
            <tr>
                <td><%= (i + 1) %></td>
                <td><%= @bookmarks[i].title %></td>
                <td><%= @bookmarks[i].description %></td>
                <td><%= @ratings[i][0] %> (<%= @ratings[i][1] %>)</td>
                <td class="btn-col"><a href="/bookmarks/view/<%= @bookmarks[i].bookmarkId %>">&#10132;</a></td>
            </tr>
        <% end %>
    </table>
</div>
<% else %>

    <p class="error-message">No bookmarks exist.</p>

<% end %>

<%= erb :"Shared/footer" %>
```

Views/Bookmarks/index.erb