

Assignment

Overview

This assignment will allow you to apply the concepts and technologies covered in class to a “real” project situation. In this project, you will design and implement a web application for a fictional site that sells reproductions of famous art classics. Some parts of the site will be available to the public; other parts in the site will only be available to site administrators. By the end of the assignment, you will have gained experience implementing a non-trivial web application.

If working in a group, each member needs to take responsibility for and complete an appropriate amount of the project work. **Be sure to consult the instructor at least one week prior to the due date if your group is experiencing serious problems in this regard.**

Deliverables Summary

You must implement the use cases as described below in the use case section. As well, the following nonfunctional requirements **must** be met in this milestone:

- 1) The use cases must be implemented in PHP using the XAMPP environment (www.apachefriends.org). You will be using the Art database supplied in Moodle.
- 2) Be sure to make use of the appropriate features of PHP. Be sure to use appropriate programming design patterns.
- 3) Do not use any external libraries or template engines without further authorization.

Source code:

- Write comprehensive inline comments using PHPDoc style of (at least) all public members!
- Use only one language (German or English) during your project.
- Be sure to make the home page file name `index.php`.
- The code is NOT a part of the written documentation.

Contents of the written documentation (no more than ten pages!):

- progress for all Use Cases as a table (Use Case#, Who did it? Progress? Problems?)
- Software design (used technologies, architecture, UML class diagram, sitemap, and directory structure etc.)
- Database design (ER diagram, remarkable SQL queries/functions/triggers [if any])
- Commented screenshots (for a quick overview, no more than five)
- Distribution of work between the team members
- Experiences, problems
- Please do NOT write a handbook.
- **The documentation is to be printed (in s/w) onto A4 paper. Please keep that in mind! (Readability etc.)**

- Hint: A written documentation is like a regular term paper. Therefore, it should contain a cover page with your names and group insignia, a table of contents, page numbers, proper styling etc.

You will lose points if you do not follow these instructions for the deliverables.

Submitting

The complete solution consisting of the source code (without the supplied image assets) and a written documentation (as PDF file in the root of the archive) shall be submitted as a single ZIP file using Moodle.

The name of the archive file shall follow this pattern:

`Assignment_Group_[GROUP-INSIGNIA].zip`

Example:

`Assignment_Group_B.zip`

You will lose points if you do not follow these submission instructions.

Grading

You can collect a maximum of 100 points in total in the following categories (the maximum reachable number of points is shown):

Software

- | | |
|--|----|
| - Degree of Completion (All Use Cases covered?) | 20 |
| - Software Design (Patterns, Creativity) | 20 |
| - Basic Security concerns | 5 |
| - Inline-Documentation (at least all public members) | 5 |
| - Code Structure and Formatting | 5 |

Documentation

- | | |
|---|----|
| - Functionality (no Handbook!) | 10 |
| - Architecture & Database (ER-Diagram etc.) | 10 |
| - Layout (including references) | 5 |
| - Content / Number of Pages | 5 |

Overall impression (including demo scenario)	15
--	----

There may be bonus points granted for optional workloads if announced.

Presentation

For the end of the term there is a presentation planned. Each team shall present their results in a working fashion using a self-developed scenario in a time frame of about 20 minutes.

This presentation is not a test. Its goal is to give each team an opportunity to show their results in action, to highlight special features and to showcase their layout. In short, to present all aspects which cannot be adequately represented in a written documentation.

Database Schema

The database supplied to you contains a complete dataset for this assignment. Additionally, to the ER model in Figure 1 it contains further tables with information about galleries and orders.

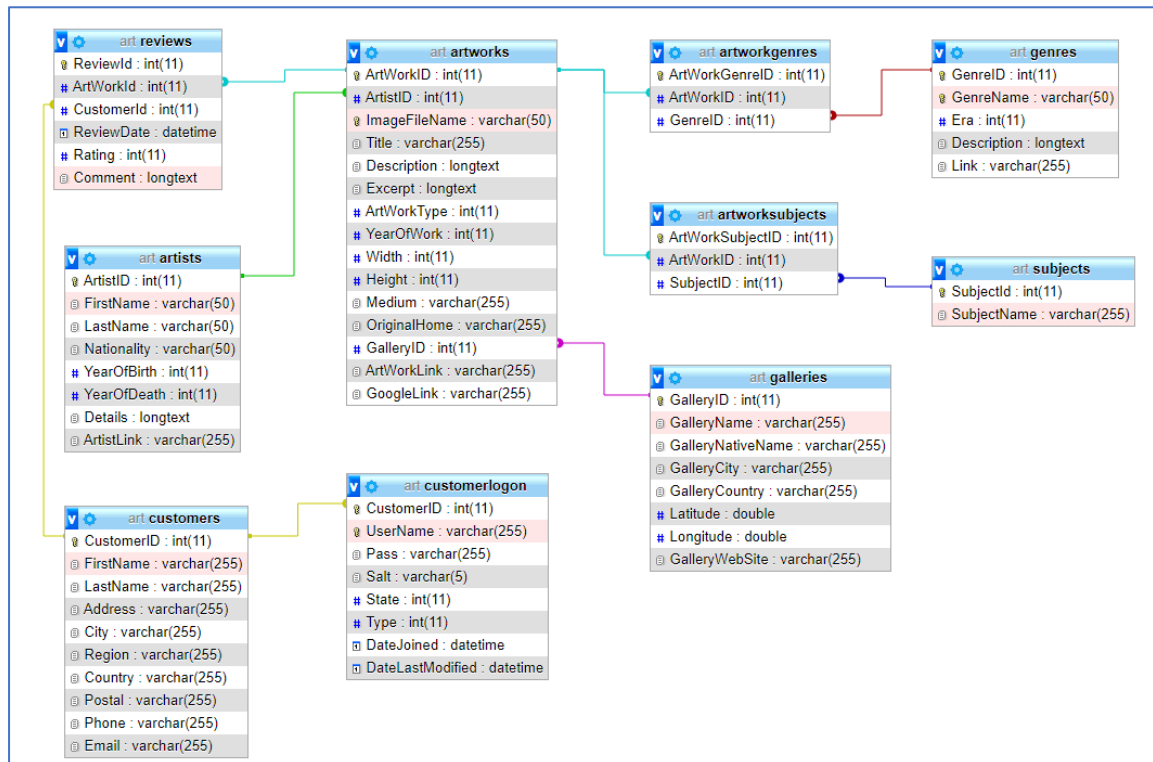


Figure 1: ER model of the supplied Arts database

UI Assets

The Art Image archive contains all the referenced images in multiple sizes and qualities for you to use.

Use Cases

USE CASE NAME:	01 - Bootstrap Theme and Site Design ✓
DESCRIPTION:	Design site's appearance by using Bootstrap
1.	<p>Use Bootstrap to style your site. ✓</p> <p>Your site must use two fonts from the Google Web Fonts collection. ✓</p> <p>Some design choices, such as header and footer treatments, can require additional CSS formatting outside of Bootstrap. ✓ ✓</p> <p>Remember a well thought and consistent design will receive more points than a broad or inconsistent mixture of styles, positioning and/or fonts.</p> <p>You do NOT have to be a designer or genius to achieve this goal!</p>

USE CASE NAME:	02 - Home Page ✓
DESCRIPTION:	Home page for the web site
1.	<p>The system displays a logo and navigation system (see Navigation use case). It should also contain a carousel (see Bootstrap site) with at least three slides. Each slide should act as a promotion of what the site offers (i.e., marketing). If the slides show artworks, then these should be randomly selected.</p> <p>The home page should display some visual boxes that contain the following data:</p> <ul style="list-style-type: none">• the three top works (based on average review rating of at least three reviews) with the rating visible in the form of stars (1 star, 1.5 stars, ..., 5 stars)• the three most-reviewed artists with the number of reviews shown.• the three most-recent reviews. (It should just list the review or perhaps just the first 100 or so characters of it and each should be a link to the according single artwork page). <p>From a design standpoint, these boxes should be encapsulated as functions (in separate files) just to be included and called on the Home Page</p>

USE CASE NAME:	03 - Navigation ✓
DESCRIPTION:	The web site's navigation system.
1.	<p>The system must provide a way to navigate to different sections of the site. ✓ There should be a ✓ primary navigation system with links to Home Page, About Us, Advanced Search, and Browse. ✓ The browse must have a submenu (you can use Bootstrap's dropdown navigation) that provides options for browsing artists, artwork, genres, or subjects. ✓ The primary navigation must be available everywhere in the site. ✓</p> <p>There needs to be a utility menu that is available everywhere in the site. It can have links to View Favorites List, My Account, Manage Users, Register and Login. ✓ ✓ ✓ ✓ ✓</p> <p>The links to use cases that you did not implement can simply be dummy links.</p>

USE CASE NAME:	04 - About Us ✓ Implementiert aber noch nicht richtig ausgefüllt
DESCRIPTION:	Display information about this site for any user.
1.	The system will display information about this site. Be sure to mention that this site is hypothetical and was created as a term project for this lecture. As well, be sure to list the group member names and roughly what parts of the project each member implemented.

USE CASE NAME:	05 - Browse Artists ✓
DESCRIPTION:	Section home for browsing artists
1.	<p>The system displays a list of all the artists in the database, along with their image (see art-images/artists). Each list entry should be a link to Display Single Artist case. ✓</p> <p>Provide the ability to change the sort order of the artists. Sort by name (last name + first name) but provide option for ascending or descending. The current sort order should be visible.</p>

USE CASE NAME:	06 - Browse Artworks ✓
DESCRIPTION:	Section home for browsing artworks
1.	<p>The system displays a list of all the artworks in the database, along with their image (see art-images/artworks). Each list entry should be a link to Display Single Artwork case. ✓</p> <p>Provide the ability to change the sort order of the artworks. Sort by title (default), artist, or year and provide an option for ascending or descending. The current sort order should be visible.</p>

USE CASE NAME:	07 - Browse Genres ✓
DESCRIPTION:	Section home for browsing genres
1.	<p>The system displays a list of all the genres in the database, along with their genre image (see art-images/genres) sorted by Era then GenreName. Each list entry should be a link to Display Single Genre case.</p> <p>This list does not need to be sortable.</p>

USE CASE NAME:	08 - Browse Subjects ✓
DESCRIPTION:	Section home for browsing subjects
1.	<p>The system displays a list of all the subjects in the database, along with their subject image (see art-images/subjects) sorted by SubjectName. Each should be a link to Display Single Subject case.</p> <p>This list does not need to be sortable.</p>

USE CASE NAME:	09 - Simple Search ✓
DESCRIPTION:	System allows user to search for artists or artworks from a single search box for any user. ✓
1.	This use case is initiated when a user enters text (at least three characters) into search box and hits some type of submit button or link. Note: this search box should be accessible everywhere in the site. ✓
2.	The system will search for results that match an artist last name or an artwork title. The search should be a wildcard search (i.e., like "searchvalue%" in SQL). The system goes to the Search Results use case. ✓

USE CASE NAME:	10 - Search Results
DESCRIPTION:	System displays a listing of search results ✓
1.	This use case is initiated after the user makes a search request. ✓
2.	The system will display, depending on the type of search, matching artists, or artworks, or both. Each result should have either the artist's name or artwork title as a link to the appropriate page (i.e., go to the Display Single Artist or the Display Single Artwork use case). You must also add other information to each search result, such as a small image or add to favorite links. ? ✓
3.	Provide the ability to change the sort order of the artists or artworks. For the artist list, sort by last name but provide option for ascending or descending. For the artworks list, provide option to sort by title, artist last name, or year, and provide option for ascending or descending. ✗

Implementiert aber funktioniert noch nicht wie es soll

USE CASE NAME:	11 - Display Single Artist
DESCRIPTION:	System displays a single artist for view by any user.
1.	This use case is initiated when a user selects a single artist to display (e.g., clicks a link that takes the user to this page).
2.	<p>The system displays information about a single artist (specified via the artist id passed in via a Query String parameter). This page should handle a missing or invalid Query String Id parameters by redirecting to an error page.</p> <p>All the information in the artists table should be displayed. There should be some way to view all the artworks by the artist. The artwork's thumbnail, title, and the View button must be links to that artwork (i.e., go to the Display Single Artwork use case).</p> <p>This page must allow the user to add the artist to a session-based favorites list (i.e., go to the Add To Favorites use case). In this case, it will be added to the favorite artists list.</p>

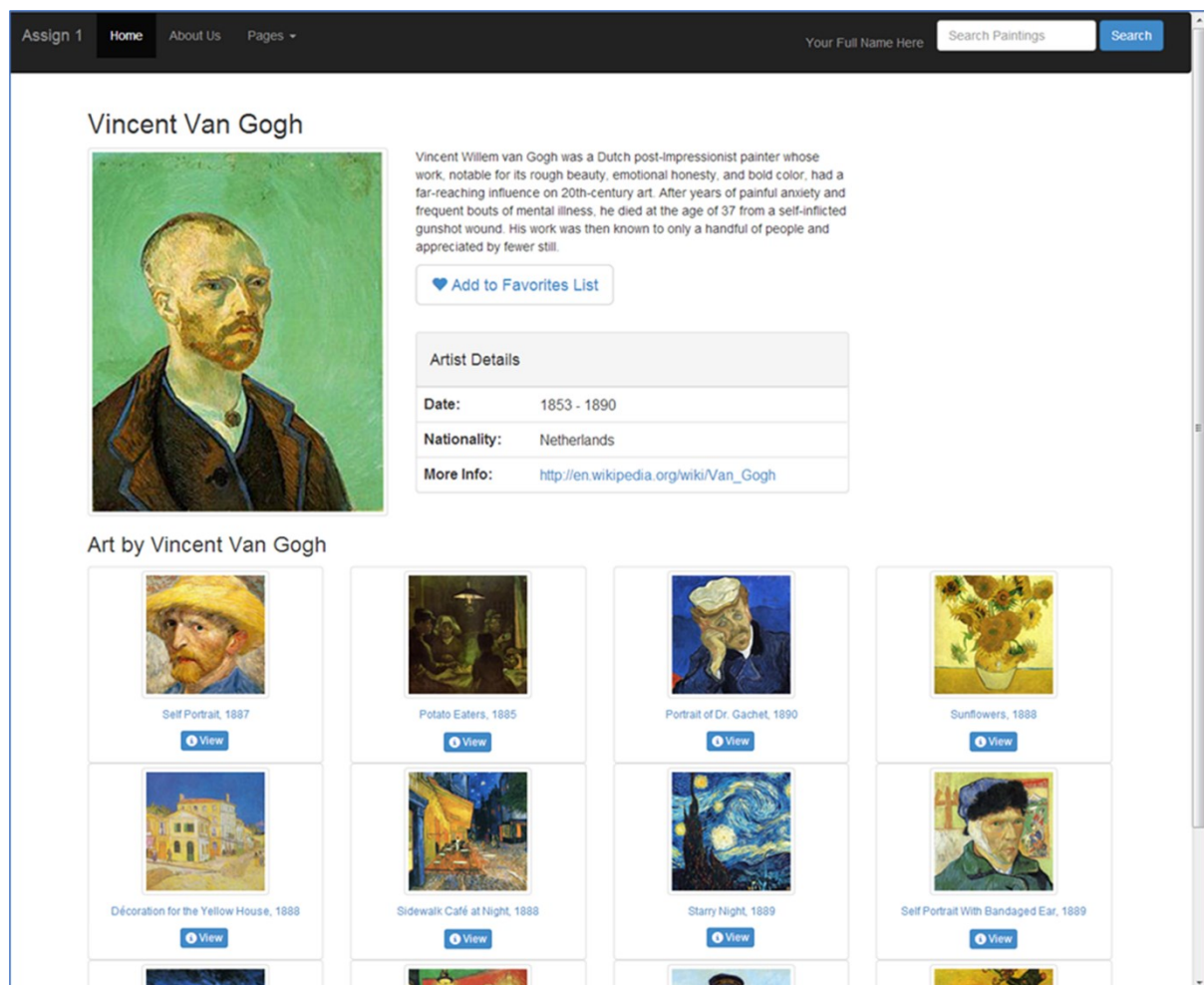


Figure 2: Layout Sample for Singe Artist page

USE CASE NAME:	12 - Display Single Artwork
DESCRIPTION:	System displays a single artwork for view by any user.
1.	This use case is initiated when a user selects a single artwork to display (e.g., clicks a link that takes the user to this page).
2.	<p>The system displays information about a single artwork (specified via the artwork id passed in via a Query String parameter). This page should handle a missing or invalid Query String Id parameter. You must display the painting information (hint: remember that SQL allows you to join information from multiple tables). The artist's name must be a link to the page for that artist. Notice that you must also display genres and subjects for the requested artwork. Be sure to turn data into hyperlinks when appropriate (e.g., artist, genre, etc.) (i.e., go to the Display Single Artist, the Browse Genres or the Browse Genres use case). Most of the information in the Artworks table should be displayed, except Cost.</p> <p>The image must be a link to the large version. Use the Bootstrap Modal dialog to display the larger version.</p> <p>Instead of displaying the OriginalHome field, you must display information about the related museum from the Gallery Table. Put the gallery information within a Bootstrap collapsible accordion element.</p> <p>This page must allow the user to add the artwork to a session-based favorites list (i.e., go to the Add To Favorites use case). In this case, it will be added to the favorite art works list.</p> <p>All users (authenticated and non-authenticated) should be able to view the reviews for this artwork. Each review entry shows the date, the given rating, the comment, and the reviewer city with country (e.g., "Berlin (Germany)").</p> <p>This page must allow the authenticated users – which have not already reviewed this artwork - to add a review for the artwork (i.e., go to the Add A Review use case). After a new review was added this page or the list of reviews should reload.</p> <p>For administrators, there must be a way to delete reviews directly from this page (i.e., go to the Delete A Review use case).</p> <p>The page must display a user rating calculated out of the review ratings (average) as well as the total number of reviews used to calculate this value. If there are no reviews for this artwork a blank rating should be shown.</p>

USE CASE NAME:	13 – Display Single Genre
DESCRIPTION:	Displays information for a single genre
1.	<p>The system displays information from genre table for specified single genre (specified via the genre id passed in via a Query String parameter). This page should handle a missing or invalid Query String Id parameters by redirecting to an error page.</p> <p>It must also display list of all artworks for that genre with the same functionality as the Search Results case.</p>

USE CASE NAME:	14 – Display Single Subject
DESCRIPTION:	Displays information for a single subject
1.	<p>The system displays information from subject table for specified single subject (specified via the subject id passed in via a Query String parameter). This page should handle a missing or invalid Query String id parameters by redirecting to an error page.</p> <p>It must also display a list of all artworks for that subject with the same functionality as the Search Results case.</p>

USE CASE NAME:	15 – Missing images
DESCRIPTION:	Show a generic placeholder image for missing images.
1.	<p>Some of the images referenced in the database do not exists or have slightly different file name.</p> <p>In the case of a slightly different file name repair the file names dynamically using PHP or correct the file names in the database.</p> <p>If the referenced image is really missing from the collection than show a generic placeholder image with the right size for the use case.</p>

USE CASE NAME:	16 - Add A Review
DESCRIPTION:	System allows user to add a review for an artwork.
1.	This use case is initiated when an authorized user uses the Add Review function (via button or link) on the page displaying of a single artwork.
2.	<p>Look at the Review table in the database to find information about the fields of the review plus links to the user. All fields must be filled with valid data.</p> <p>Do not make the user enter the review date (you can programmatically set that) and use the appropriate HTML5 control for entering a rating between 1-5. Apply basic validation methods to ensure that only valid reviews (e.g., not empty ones) are stored in the database.</p> <p>A user should be able to review a given artwork only once. X</p>

USE CASE NAME:	17 - Delete A Review
DESCRIPTION:	System allows administrators to remove a review for an artwork.
1.	This use case is initiated when an authorized administrator uses the Delete Review button or link on the page displaying a single artwork.
2.	Show a brief dialog and let the administrator confirm the deletion.
3.	<p>If the deletion is confirmed, remove the selected review from the database.</p> <p>The page to display a single artwork should reload itself after the deletion.</p>

USE CASE NAME:	18 - Add To Favorites List ✓
DESCRIPTION:	Way to add an item to the user's favorites.
1.	Initiated when user clicks the Add item to favorites link/button.
2.	<p>The system will add the item to the appropriate favorites (i.e., artist or artwork) list and then indicate visually somehow that the item has been added to the favorites.</p> <p>The favorites list only needs to be stored in the current web session and must not be stored in the database.</p>


USE CASE NAME:	19 - View Favorites List ✓
DESCRIPTION:	Way to view and modify the contents of user's favorites list.
1.	Initiated when user clicks the view favorites link/button.
2.	The system will display a listing of the contents of each of the two favorites list (with the same behavior as the Search Results use case). Allow the user to modify the favorites list's contents (i.e. remove artist or artwork).


USE CASE NAME:	20 - Register User ✓
DESCRIPTION:	Ability to register as a new user ✓
1.	Initiated when user clicks the Register link/button on the home page. ✓
2.	<p>The system allows an unauthorized user to register as a new user. ✓</p> <p><i>See the proper tables in the database for the necessary attributes to be collected!</i></p> <p>Appropriate validation (last name, city, address, and country not blank, proper email) should be covered. Be sure to use the best user input types. Ensure that a good password is entered by the user and that a username can be used only once. ✓</p> <p><u>Extend</u> the database to discriminate between regular users and administrators. A self-registered user is always a regular user and can be elevated to administrator only by another administrator. ✓</p>

USE CASE NAME:	21 - Manage Users
DESCRIPTION:	Ability to manage the registered users
1.	Initiated when an administrator clicks on the Manage Users link/button in the utility menu. This link/button should not be visible for regular users. ✓
2.	<p>Administrators can edit the information of all registered users (including other administrators). ✓ They should be able to see a list of all users, and then be able to select a user to update. ✓ Appropriate validation (e.g., last name not blank, or proper email address) should be covered. Be sure to use suitable user input types.</p> <p><i>See the proper tables in the database for the necessary attributes to be maintained!</i></p> <p>This includes the capability to remove a user from the list of currently registered users and to elevate another user to Administrator level. ✓</p> <p>Users must not be deleted in the database as there can be other information (like reviews) related to them. ✓</p> <p>Make sure that the last administrator cannot remove/demote himself. ✓</p>

USE CASE NAME:	22 - Login as user ✓
DESCRIPTION:	Ability to log-in as a registered user
1.	The system displays a login form on the home page.
2.	After submission of the user information, it is validated using the database.
3.	If the supplied credentials are not valid, the system displays a proper error message to the user.
4.	After authentication, every page should display the authenticated user's name and a link/button to log out. After logging out, the current user should be treated as an unauthorized user.

USE CASE NAME:	23 - My Account ✓
DESCRIPTION:	Way to manage the own user account data
1.	Initiated when user clicks the My Account link/button in the utility's menu.
2.	The system will allow the current user to change its own data including the password. Make sure to apply at least basic validation and <u>security techniques</u> when implementing this use case (see the Register User and Manage Users use cases for more information).

USE CASE NAME:	24 - Advanced Search 
DESCRIPTION:	System allows user to search for artists or art works that match a set of criteria
1.	This use case is initiated when a user selects an advanced search link/button.
2.	<p>The system allows the user to specify what kind of result they are looking for: artist or artwork. The system will allow the user to specify different search criteria depending upon what kind of result they are looking for.</p> <p>If they are looking for matching artists, then the search criteria will be: artist name, year of living range (e.g. lived between 1700 and 1800) and nationality.</p> <p>If they are looking for matching artworks, then the search criteria will be: artwork title, year of work range (e.g. 1830 – 1875), and genre.</p> <p>The user can select artwork criteria or artist criteria but not both at the same time.</p> <p>If a criterion was left blank, then it should be ignored by the search. If more than one criterion was entered, then the search results must match the given set of criteria.</p> <p>The system goes to the Search Results use case.</p>

USE CASE NAME:	25 – Map to Museum 
DESCRIPTION:	Display of a map for the location of the gallery for a single artwork.
1.	Initiated when user views information about a single artwork (see the Display Single Artwork use case).
2.	Extend the existing gallery information within a Bootstrap collapsible accordion element with an Open Street Maps map with a marker for the gallery.

Guidance

This is substantial assignment, and your group will likely need to invest 50-70 hours each into it.

I would recommend the following process.

1. Review the provided database model and understand the domain objects.
You do not need to create new tables or fields to implement all these use cases.
2. This is a data-driven site, so I would recommend beginning by constructing the data access layer. To begin, create classes and repositories for Artist, Artwork, Genres, Subjects, and Galleries. You will need to add methods that match the functionality required by the use cases. For instance, the Artworks repository, will need methods such as `GetAll()`, `GetByID()`, `GetForArtist()`, `GetForGenre()`, `GetForSubject()`. All these other methods are simply wrappers for different SQL statements. Some of these methods can be implemented as you need them.
3. It is recommended, that you write test methods in separate pages for each repository so that you know each works as expected before you use them in your use cases.
4. Construct your user interface.
5. Implement the use cases.
6. Add session abilities for favorites lists.
7. Implement the optional use cases as the final cases.

Do not forget to think about basic security procedures! (e.g., "All input is evil!")