IMY 220 Project 2023

Article sharing website

The project must be completed by Thursday **26 October**.

We recommend that you test it on the server throughout the semester.

Contents

Objective

Create a website that allows users to write and share articles. You must use the technologies and techniques as specified in the project spec section to write your own code.

Your website must have a unique visual design that fits with the theme of the website (i.e., an article-sharing website). You are not allowed to simply use another website's design or create a "knockoff" of another website when designing the visual theme of your site. Note that you also don't have to use the terminology used in this document to describe functionality, such as "article", "friend", etc. You may name these however you want based on the theme of your website.

The purpose of the project is for you to demonstrate the skills that you have learned in IMY 220 - not anything else. Therefore, you will not get marks for proving your proficiency with any other skills OR for using code that is not your own (e.g., from other libraries or templates).

Submission Instructions

- ***Late projects will not be accepted.***
- You have to upload your files to the **imy.up.ac.za server** as well as **ClickUP**.
- *IMPORTANT: Please do not move your files around to different directories (in order to fix the directory structure) after you have finished writing your code.*

imy.up.ac.za server

Uploads to the server will be **blocked at exactly 23:59 on Thursday 26 October**.

- Please make sure that you start uploading long before the deadline.
- Upload and test your project on the **imy.up.ac.za** server throughout the semester. This is very important as your local server might ignore bugs in your project that the server would not.

ClickUP

The submission link on ClickUP will close at **exactly 23:59 on Thursday 26 October**.

- Do not wait until just before 09:30 to upload because your submission might take longer than usual to upload.
- Instructions:
    - Place all your project files, along with your exported MySQL database (in the form of an SQL text file) in a folder named with your Position_Surname.
    - Compress the FOLDER using a zip archive format.
    - Upload the compressed folder to ClickUP to the relevant project link.

IMPORTANT: Do not expect the lecturer to immediately respond to communications around the submission time. It is your responsibility to make sure you submit all the relevant files well before the submission time.

Project demo

You will have to demo your project from the **imy.up.ac.za** server in order to get marks for it.

- You will have to demo using either Google Chrome or Mozilla Firefox.
- When you demo your project, you will be required to show how you have implemented all the instructions by demonstrating the functionality on the website itself (not in the database).
- Demo sessions will be made available on ClickUP closer to the deadline.
- Demo sessions will take place on **Monday 30 October** and **Tuesday 31 October**.
- You will receive marks for achieving working functionality according to the instructions. For example, if you are required to implement functionality that modifies and displays database data on the website itself, you will receive no marks for simply modifying the database.
- Project demos will take place in the Virtual Reality and Interaction lab on the 4th floor of the IT building (IT 4-62).
- ***You will receive no marks if you do not demo your project.***

You will lose marks in the following conditions

- If your system does not have all the required functionality.
- If you use any other technology or technique that is not part of the course.
- If you use code that is not your own, i.e., if you use code generators or templates, you will not get marks for it.

*You will get **no marks** in the following conditions*

- If you submit work that is not your own.
- If you do not upload to imy.up.ac.za
- If you do not upload to ClickUP.

Main pages

All pages must clearly display the name of the website creatively and have a unique visual design.

Splash page

- When an unregistered user or a user that is **not logged in** visits the website, they must only be allowed to see a “splash page”.
- A splash page is the webpage that the user sees first before being given the option to continue to the main content of the site.
- Splash pages are used to inform new users about the services provided by the website and to promote it. You must thus include a tagline or other information that explains the contents and purpose of the website.
- The splash page is the only page that does not have the same consistent layout as the rest of the webpages.
- The splash page must contain at least the name of the website as well as a log in and registration form.

- *BONUS: catch new users' attention and show off your design skills by creating a well-designed landing page that explains the goal of your website. Divide this page into well-designed sections and use the page scroll to trigger visual effects when the user scrolls down. How you make use of page scroll is up to you.*

Home page

- When a user is **logged in**, the home page must contain the **local activity feed** for that user in reverse chronological order.
- A user should then be able to switch between their local activity feed (that displays the activity for all that user's friends) and a global activity feed that contains the activity for all users. Also display this in reverse chronological order.

Profile page

- A user should be able to go to their profile page when they are logged in.
- The profile must have a profile image. If a user has not uploaded a profile image, there should be a placeholder image instead.
    - *BONUS: allow users to add an image using drag-and-drop.*
    - *BONUS: come up with a way to randomise the placeholder image so that each one is different in some way. This should go beyond simply having a few images to select from.*
- A profile must display appropriate personal information about a user, e.g., birthday, work, contact info, relationship, etc.
  Note: Since all this information will be publicly visible, no private information must be shown.
- Each user's list of friends should appear on their profile as well but should only be visible if the user who is logged in is friends with the user whose profile they are viewing. (These details should also be visible if a user is viewing their own profile.)
    - *BONUS: in this friend list, show which friend(s) are currently online.*
- A user should be able to edit appropriate information in a logical manner.
- When you visit the profile page of another user, it must clearly indicate if you are friends on the site.
- There must be some logical way to connect with a user, i.e., send them a friend request.
- When you visit the profile page of another user you are friends with there must also be an option to send messages to the user.
- A list of all the articles created or read by the user must appear on their profile page. You must come up with a sensible and visually appealing way to display this. Keep in mind that there might be many created and read articles which means you must come up with an intuitive and visually appealing way to provide access to each.
    - If an article that a user has read has since been deleted, this must be visually indicated in the list of read articles.

Activity feeds and articles

The website should show two activity feeds: a local activity feed and a global one. Activity in this case refers to creating new articles or sharing articles created by someone else.

The local activity feed consists of all the activity for the current user as well as for all their friends, while the global activity feed consists of all the activity for all users on the website. The activity on the activity feed must only contain the name, image, and summary of an article and accompanying details, e.g., if an article was newly created or read by someone in your friend list.

Activity must be displayed in an aesthetically appealing way. Simply listing the pieces of activity in Bootstrap cards underneath each other is not sufficient; find inspiration from other websites/applications that use news feeds, activity feeds, etc.

Articles

An article in this context refers to a page with a name, summary, a cover image, the article text itself, and one or more hashtags. The following applies with regards to creating articles:

- Any user can create an article.
- An article must contain a cover image and a text summary that summarises what the article is about.
    - *BONUS: allow users to add images using drag-and-drop.*
- An article must also contain the date that the article was created, which must be automatically added from a timestamp, i.e., the user should not be able to edit this manually.
- An article must also contain a category which is selected from a defined list of categories on the website, e.g., “recipe”, “business”, etc. (Only admins should be able to manage categories.)
- The summary of an article must also be able to contain hashtags. You can decide whether users should be able to add them to the text description or in a separate input.
  Note: A hashtag is an interactive piece of text that is created by prefixing a predefined symbol to it, e.g., #cocktailrecipe. When a user clicks on a hashtag, it must be the same as performing a search for that hashtag, i.e., it must display all other posts that have the same hashtag as part of their summary.
- When viewing an article by clicking on its page - for example, from the newsfeed or search results - it must have its own article page, which is functionally similar to a profile page, but shows the details (name, summary, cover image) and also allows a user to read the full article.
- The main text of the article should be long-form text that the user can edit in an easy and intuitive manner.
    - *BONUS: implement basic WYSIWYG (what you see is what you get) functionality that allows users to add formatting, such as bold and italic, font size, etc. to the article text. The mark for this will depend on the level of complexity at which this is implemented, e.g., setting font size for all text is relatively simple whereas adding markdown parsing is much more complex.*
    - *BONUS: allow users to add images anywhere into the article text.*
- The user who created an article should also be able to update all the information associated with the article. This includes deleting other users’ comments including the associated images (details for comments below).
- Users should be able to give an article a rating, e.g., out of 5. The overall rating should be displayed along with each article. This should be the average of all ratings given by users. The creator of an article should not be able to edit ratings.
    - *BONUS: along with the article itself, also display an appropriate review in the newsfeed using tooltip/hover-over functionality. The review must be selected based on the rating that was*

also provided by the same user. For example, if the article has an overall rating of 3.7/5 then the review of the user who rated it most closely should be displayed. (If there is more than one appropriate review in this case, you may select any one.) This should be loaded dynamically, i.e., they should not load along with the rest of the page, only when the user hovers over the article in the newsfeed.

- Users should also be able to leave a review for each article and should be able to include an image along with their review. This should be displayed in a logical way, for example, if there are 50 reviews, only a few of them should be displayed on the article's page and the rest should be accessible through some other mechanism.
    - *BONUS: allow users to like reviews which visually indicates the number of likes per review.*
    - *BONUS: allow the creator of an article to pin a review, in other words, this review would always display first and should be visually indicated as being pinned.*
- The creator of an article should also be able to delete their article. Users who have read this article should still have it listed on their profile page, but deleted articles must be indicated visually and should not be clickable links.
    - *BONUS: when creating an article, allow users to make the article private in which case only their friends should be able to read and comment on the article. If someone who should not have access tries to view the article page (e.g., by typing the page URL into their browser), they should only see a well-designed page telling them they do not have access.*

The following applies with regards to reading articles:

- Any user should be able to "read" an article by navigating to its page and scrolling down to the end of the article.
  Note 1: for the purposes of this project, you can assume that if a user scrolls to the end of the article, they have read it.
  Note 2: if you are not able to implement this functionality, you should still implement a way for a user to mark an article as "read" somehow, for the sake of other functionality that relies on this.
    - *BONUS: in the newsfeed, visually indicate if the current user has already read an article.*

- The article page should contain a visual indication of how far the user has scrolled down the page, similar to either of these examples:
    - https://blog.hubspot.com/website/award-winning-websites?hubs_content=blog.hubspot.com
    - https://techcrunch.com/2023/07/31/google-assistant-reportedly-pivoting-to-generative-ai

Lists

A user should be able to create a list of articles, e.g., different lunch recipes. The following applies with regards to creating and managing lists:

- There should be a sensible and intuitive way to add an article to a list.
- A list should have a name and description which can be changed by the user who created the list.
- How a user accesses their own lists is up to you, it just needs to be user-friendly and well-integrated into the rest of the website.

- *BONUS: create functionality for recommending articles for a specific list based on the hashtags of articles they have previously attended to said list. Hashtags that appear more often in a list should be given preference when recommending other articles. You must come up with a sensible and aesthetically pleasing way to integrate this into the list functionality.*

User Roles

The unregistered user

An unregistered user can do the following:

- Register as a new user on the splash page. The splash page does not offer any functionality other than allowing the user to register or login.
  Note: An unregistered user cannot interact with the website in any other way.

The registered user

A registered user can do the following:

- Log in and log out.
- See all activity from themselves and all their friends (local activity feed) as well as from all users (global activity feed).
- Search for articles or users and view other users' profile pages.
- Edit all appropriate fields on their own profile.
- Add new articles.
- Manage (edit/delete) their own articles (this is discussed further under Articles).
- Delete their own profile. Deleting their profile removes their user account from the database.
  Note: This should automatically delete all articles that were added by that user. The same applies as when deleting an individual article.

User interaction

A registered user can do the following:

- Connect with other users by following them or sending a friend request to connect.
- View the complete user profiles of users that they are friends with.
  Note: When a user is not friends with another user, they can only see that user's name, profile picture, and their latest read article.
- Unfriend other users.
- Privately send messages to other users.
  Note: When a user sends a private message to another user, that user must get a notification when they log into the system and the notification must go away when the user opens the message.

    - *BONUS: allow users to create group chats, similar to groups in WhatsApp. Users should be able to invite their friends to groups, give the group a name, and message sent would be visible by all members in the group.*

- *BONUS: allow users to send a YouTube video instead of a text message. When a YouTube video link is pasted into the text box, this must be automatically detected as being a YouTube video and the resulting content must be replaced with an embedded video. This must work for both the normal URL format (e.g., https://www.youtube.com/watch?v=dQw4w9WgXcQ) and the shortened format (e.g., https://youtu.be/dQw4w9WgXcQ).*

The admin (user account)

Some users should be able to log in as admin. The administrator has all the powers of a registered user.

Additionally, they can do the following:

- Manage (edit/delete) all activity.
- Manage (edit/delete) all articles.
- Manage (edit/delete) all user accounts.
- Add new article categories.
- *BONUS: add functionality for admins to verify accounts. A (non-admin) user should only be able to request that their account be verified if their account is more than one week old, if they have created at least one article, and read at least one. Admins should have access to a list of verification requests that they can approve/deny on a case-by-case basis. Once an account is verified, it should be visually indicated on the user's profile somehow.*

Search

Add search functionality that allows a user to search for the following:

- Other users by name, username, or email address.
- Articles by name, date, or summary
- Hashtags
- *BONUS: The search functionality must be able to find something even if the search term is a little incomplete or spelled incorrectly, i.e., fuzzy string searching: the technique of finding strings that match a pattern approximately (rather than exactly).*

  Note: this does not refer to simply adding wildcards to your searches. A fuzzy search should be able to find words that have letters missing, swapped around, etc.
- *BONUS: Add autocomplete functionality to suggest results as the user is typing. You would need to come up with an intuitive design for this.*

Search results must be interactive, i.e., where appropriate they must return links, for example, to user profile pages, article pages, etc.

Usability

- Take care to design a website that is intuitive and easy to use. It is your own responsibility to come up with usable and intuitive interactions for the website's functionality.
- For example:
    - There must not appear a log out button if a user is not logged in, or a login button must not be visible if a user is already logged in.
    - When a user creates an account, they must automatically be logged in with that account.
    - All form fields must have working labels, i.e., when you click on the label, focus is given to its accompanying input.
    - Well-designed error messages should appear when appropriate, e.g., when a user enters incorrect login details.
    - Navigation must stay consistent throughout pages and must provide a logical way to access the core functionality.
    - Text that is interactive must look interactive (buttons, links, hashtags).
    - Clicking on the website logo/name must take the user to the home page.
- *Tip: Test the usability of your site by asking other people (especially non-IT students) to interact with it to find out how easy it is for them to navigate.*
- You will be penalised for sloppy design that is detrimental to the user experience, such as debugging alerts (or any unnecessary alerts).
- *BONUS: allow the user to toggle between a standard (light mode) and a dark theme (night mode).*

Visual design

- Your website must have a design that is up to professional standards. Refer to well-designed websites for inspiration, e.g., https://www.awwwards.com, https://dribbble.com/shots/popular/web-design, etc.
- Use colour schemes that match, e.g. browse the colour lovers website (https://www.colourlovers.com/) or use Adobe Color (https://color.adobe.com/) or Coolors (https://coolors.co/) to find colour palettes and patterns or search online for colour scheme creators.
- You **must** redesign all form elements to fit with your website's visual theme, in other words, you are **not** allowed to have **any default** CSS/Bootstrap input boxes, buttons, etc.
- You must include at least two fonts as part of your website's design. These may not be the default fonts for HTML or Bootstrap. Use fonts that look good and fit the overall design of the site. You must embed fonts for your website.
  Note: Fonts must be legible. Do not use more than three fonts for the entire website. Do not use stylised fonts for body text. Do not use standard Windows fonts. Embed fonts with CSS.

Technologies and Techniques

You may only use the following technologies and techniques:

Technologies:

- Valid HTML5 everywhere (no exceptions). Make sure that when the HTML is returned from the server and viewed on the client (view source) that the HTML is correct, i.e., ONE doctype, ONE head, ONE body. You can use the W3C Markup Validation Service (https://validator.w3.org) to validate your code.
- CSS3 for styling, embedding of fonts, etc. Do not use inline or embedded style sheets.
- Bootstrap CSS framework for layouts etc.
  You **have to** use additional CSS to style elements according to your website's theme. You will lose marks if there is any default Bootstrap styling visible on your website.
- jQuery for layout design, element manipulation, and AJAX usage.
- JSON to transfer your data back and forth between client and server (for example, during AJAX calls).
- AJAX for server calls. Use AJAX only when it is appropriate and would positively affect user experience, such as performing an action that requires database interaction in a case where you do not want to refresh the page.
- At least one instance of a Promise for asynchronous behaviour (this can include the built-in async behaviour of using jQuery AJAX).
- Use SQL and MySQL database.
- *BONUS: use Github for version control. To get the bonus marks, you must have at least 4 commits, each at least two weeks apart.*

Techniques:

- Responsive web design with at least two breakpoints.
  Note: Your website must display correctly on any resolution.
- Error free PHP. Sensible coding, you must only connect to your database in one place. Keep your code DRY (**D**on't **R**epeat **Y**ourself).
- Input validation. Extend the default HTML5 input validation where appropriate (for example, to only let users join if they are above a certain age).
- Session management.

Favicon

- Design and embed a favicon that fits the theme of your website in all your pages.

Database

- There must be **LOGICAL** test data in the database for demo purposes.
- There must be at least the following:
  - 10 existing articles in the database.

    - 5 lists.
    - 10 registered users.

    NB: for general testing purposes, you must have one regular user account with the following login details:
    Email address: *test@test.com*
    Password: *test1234*
    This user must have at least 1 created article, 2 read articles, and 2 friends.
- For the purposes of testing and marking, do not hash your passwords.
- Make sure that image sizes are small enough to stay within the total submission size limit of 50MB.
- Export your complete database to include in your ClickUP submission.

Directory structure

- Structure all your files into a logical directory structure.
  Tip: Start off by figuring out your directory structure instead of moving your files around after you've coded your website, since your references will change if you move files around.
- Name your splash page index.php/index.html.
- Make use of correct file naming conventions.
- Put images, fonts, scripts, css files, etc. in their own folders. Also use folder organisation for your PHP files.
- Use relative file paths for references.
- Make sure the slashes in your references are not backslashes.

Completing the project

You have until 26 October to complete the project.

You have to submit on imy.up.ac.za and ClickUP. Remember to download and double-check your submissions.

To avoid running into problems on 26 October, it is crucial that you test your project on the server throughout the semester. Do not wait until just before the submission deadlines to start testing your project on the server.

Make sure to retest the entire site after submitting it to the server to ensure that what you will be assessed on is fully functional.

FTP – things to look out for

- You will have to put your project on a production server for demo purposes.
- The server will work similar to your local server, but there might be a few differences that you need to be aware of.
- The server has PHP v7 installed on it.

- The server displays all PHP errors and warnings, so it is essential to display all errors and warnings in your PHP code or local server settings while you develop.
- The server does not have file writing permissions switched on by default, so you have to change the file permissions in FileZilla for when you want the user to upload images etc.
- The server is less lenient when it comes to bugs in your code. E.g., on your local server images might still display if the file extension case in your code differs from your image file extensions, but on the server, it would not display! OR back slashes in file references might work on a local server, but it has to be forward slashes in for it to work on a server.
  THUS: Your project would run smoothly on the server if your code is bug free
  AND it will not take you long to FTP if you test your project on the server throughout the semester.
- The only reason it might take long is if you upload and you realise that there are bugs in your code and you have to start debugging.
- A local server (that you used on your laptop) is not as strict as a production server when it comes to bugs. The only way to ensure that your code will work on the production server is to test it throughout the semester so that you can debug as you go.

Mark breakdown

Project Progress (Deliverable 1) 15%

Project Progress (Deliverable 2) 15%

Final Project (Deliverable 3) 70%

Start today!