ANNALES MEMUM

By Samuel Pearce and Julian Werner

CONTENTS

1 - PROJECT CONCEPT

The idea behind the Annales Memum is to be a chronicle of “memes”. These are images, videos, stories, etc. That are shared throughout the internet virally. We believe that, due to them being often misunderstood, it is important to keep a record online of the history of these strange, but entertaining ideas. We can use the site to preserve history for future generations to understand internet culture, to keep statistics records, or just to reflect on an idea’s impact to society.

The website will be a custom wiki that allows users to create/edit pages themselves with no planned moderation. Should the project become successful, a long-term moderation plan should be created mimicking that of Wikipedia. Each page should document three main aspects: a meme’s origin/history, a meme’s popularity i.e. what media it was popular on and how many people were exposed to it, and a meme’s cultural impact. The impact should describe whether the meme had a lasting effect, e.g. “Pepe” had an extremely potent effect and lasted far past its due expiration date, but “Sirenhead” had a pretty standard lifespan.

Another aspect of meme-culture to be explored and documented on the site are the theories and sciences pertaining to the study of memes. This would encompass things like the theory of meme lifespan which seek to allow the philosophical discussion about memes.

2 - REQUIREMENTS

2.1 - FUNCTIONAL REQUIREMENTS

- Users can create accounts.
- A user with an account can create new pages.
- A user with an account can edit pages.
- Each page should document the history and public opinion about a meme.
- Meta-pages may exist.
- The users may use markdown to style their pages.

2.2 – NON-FUNCTIONAL REQUIREMENTS

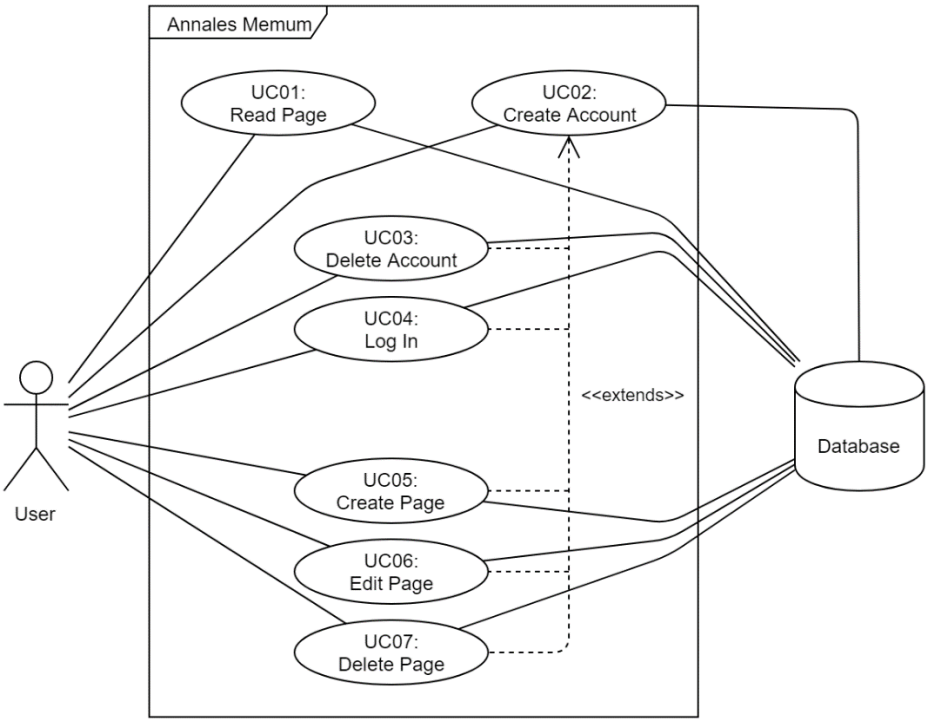
- The project will use a Spring backend.
- The project may be deployed using Docker containers.
- The project will be able to use multiple Databases (5-Tier Architecture)
- The front-end will use HTML, CSS, and JS.

2.3 – NICE-TO-HAVES

- Users can translate a page into multiple languages.
- Users can write a short description about themselves on a personal page.
- Users can earn the privilege to higher moderation rights and only higher-level users may edit protected pages.
- Pages can be protected by higher-level users to prevent griefing via deletion or modification.
- Admins can restore deleted pages.

3 – USE-CASES

3.1 – USE CASE DIAGRAM



3.2 – USE CASES

ID	UC01
Title	Read Page
Preconditions	- The user is on the website's main page.
Main Flow	1. The user enters a search term into the search box. 2. The user selects one of the found pages. 3. The page content is displayed to the user.
Possible Exceptions	- **Page not found:** 1. The user is informed that no results were found. 2. Proceed as above.
Postconditions	- The user is on the page they wanted.

ID	UC02
Title	Create Account
Preconditions	- The user is on the website.
Main Flow	1. The user clicks on the "Create Account" button. 2. The user enters the required details. 3. The user is redirected to the main page and a success message is displayed.
Possible Exceptions	- **Invalid input Data:** 1. The user is informed their details are invalid. 2. Proceed as above.
Postconditions	- The user's account has been created.

ID	UC03
Title	Delete Account
Preconditions	- The user is on the website. - The user is logged in.
Main Flow	1. The user selects the dropdown next to their name. 2. The user selects the “Delete Account” option. 3. The user enters their password and confirms they wish to permanently remove their account. 4. The user is redirected to the main page and a success message is displayed.
Possible Exceptions	- **Invalid Password:** 1. The user is informed their credentials were invalid. 2. Proceed as above.
Postconditions	- The user’s account has been permanently deleted.

ID	UC04
Title	Log In
Preconditions	- The user is on the website. - The user has an account. - The user is logged out.
Main Flow	1. The user clicks the “Log In” Button and enters their account credentials. 2. The user is redirected to the main page and a success message is displayed-
Possible Exceptions	- **Invalid Password:** 1. The user is informed their credentials were invalid. 2. Proceed as above.
Postconditions	- The user is logged in

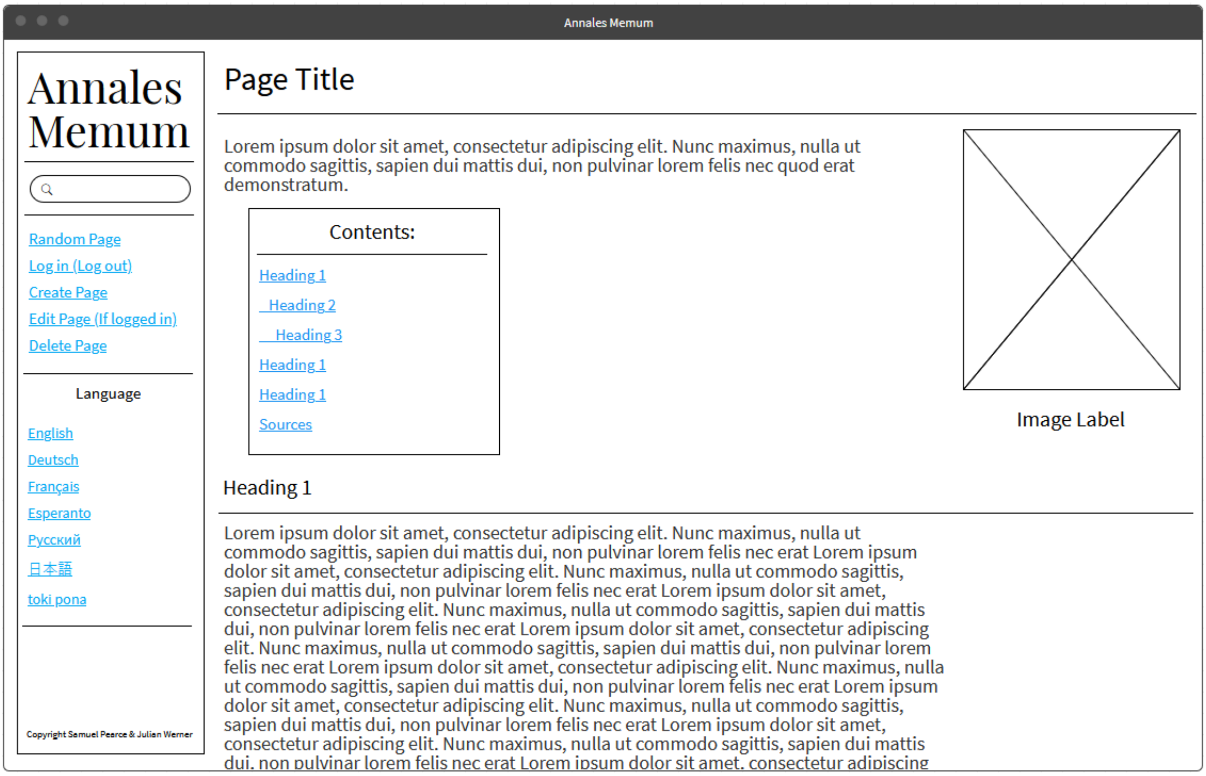
ID	UC05
Title	Create Page
Preconditions	- The user is on the website. - The user is logged in.
Main Flow	1. The user selects the dropdown next to their name. 2. The user selects the “Create Page” option. 3. The user enters the title and content for the new page. 4. The page is created, and the user is redirected to the new page.
Possible Exceptions	- **Invalid Page Data:** 1. The user is informed that their chosen data is invalid and asked to try again. 2. Proceed as above.
Postconditions	- The new page is created and publicly visible.

ID	UC06
Title	Edit Page
Preconditions	- The user is on the website. - The user is logged in. - The user is on any editable page.
Main Flow	1. The user clicks on the “Edit” Button. 2. The user is shown an editor with the current page contents where they make their changes. 3. The user clicks the “Save Changes” Button. 4. The user is redirected to the page and a success message is displayed.
Possible Exceptions	- **Invalid Page Data:** 1. The user is informed that their chosen data is invalid and asked to try again. 2. Proceed as above.
Postconditions	- The page has been updated with the new content.

ID	UC07
Title	Delete Page
Preconditions	- The user is on the website. - The user is logged in. - The user is on any editable page.
Main Flow	1. The user clicks on the “Delete” Button. 2. The user is prompted for their credentials and asked to confirm they want to delete the page. 3. The user is redirected to the main page and a success message informs them the page was successfully removed.
Possible Exceptions	- **Invalid Credentials:** 1. The user is informed that their credentials invalid. 2. Proceed as above.
Postconditions	- The page has been marked as deleted. (**NOT** Deleted from the database though.)

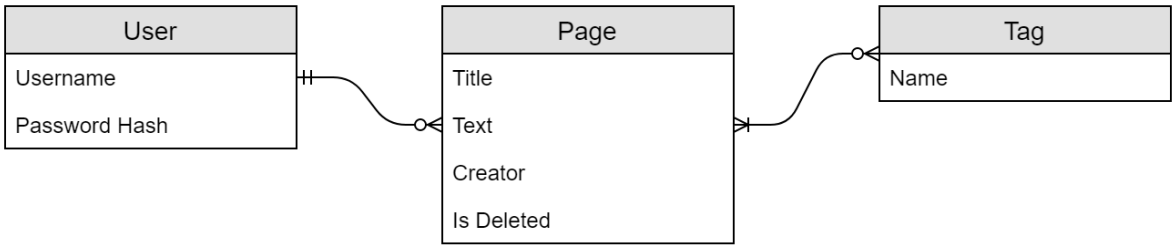
4 – MOCKUPS


Annales Memum
Log in
Annales Memum
Logo
Search
Random Page

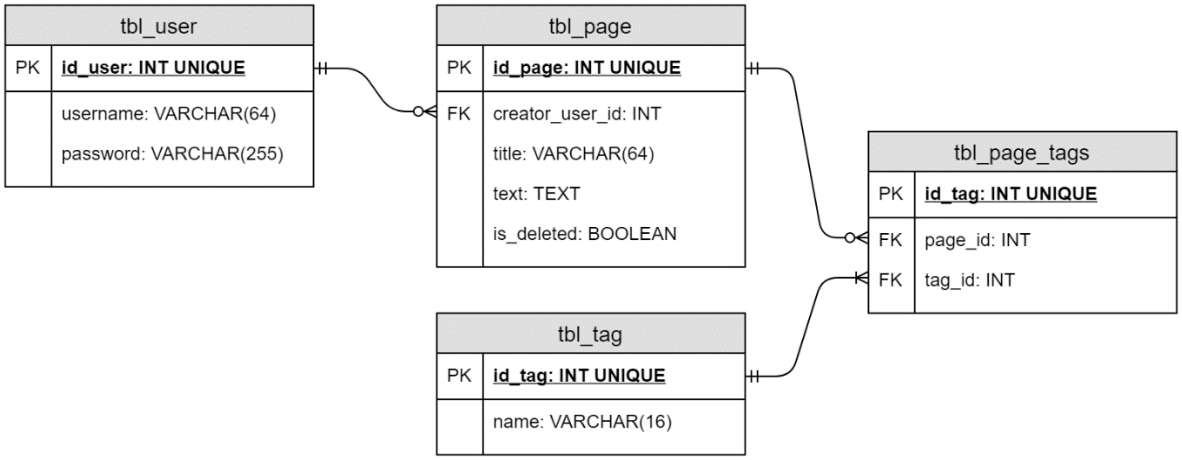

Annales Memum
Annales Memum
Random Page
Log in (Log out)
Create Page
Edit Page (If logged in)
Delete Page
Language
English
Deutsch
Français
Esperanto
Русский
日本語
toki pona
Copyright Samuel Pearce & Julian Werner
Page Title
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec quod erat demonstratum.
Contents:
Heading 1
Heading 2
Heading 3
Heading 1
Heading 1
Sources
Image Label
Heading 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat Lorem ipsum dolor sit amet, consectetur adipiscing

5 – DATABASE

5.1 – ENTITY RELATIONSHIP DIAGRAM



5.2 – RELATIONSHIP MODEL



6 – WHAT WE'VE LEARNED

6.1 – SPRING SECURITY

I think the main issue with us using the spring security framework, is that, if we really wanted to use it correctly, we would have needed to take a full course. It's a powerful framework indeed, but we were not capable of utilising its full capacity and would undoubtedly made serious errors, or worse yet: not gotten the project working at all.

The Spring framework's security features include support for other, more well-known, security systems such as OAuth, LDAP, and Kerberos. It utilises these in its two main functions: Authentication and Authorisation. Authentication is primarily concerned with the login functionality. It is possible to link the user-data used by Spring to your existing database, so that all your users can be added normally while automatically allowing them to be used by the security system. Authorization is, as one might expect, handles which pages, and which parts of pages, are visible to whom. This is a far more in-depth topic, which we didn't even get to before being held up. Spring Security's base relies heavily on XML configuration files, which get very opaque quickly.

6.2 – FLASK

After spending some time wrangling with Spring's security system and not making any Progress, it became increasingly clear that we might have to find another method if we were to produce anything functional. With the deadline inching nearer each day, Julian suggested we switch to a different basis. We really didn't want to regress to using PHP, so Julian found a Security framework

using Python and was eager to give it a go. We both did some preliminary research on how to use it. Before we knew it, Julian had whipped together a functional register and log-in system. He touted the framework's lack of boilerplate code; it was a stark contrast to spring which is mostly boilerplate code.

Flask is a microframework that requires very little setup and has many built-in features to make the development lifecycle easier. Certain basic security measures are built in or can be activated with a simple configuration option.

6.3 – DOCKER & DOCKER SECURITY

A while we had been given an introduction into Docker and were also interested in trying out containerising the whole application. The server was also immaterial; Sam had set up a server at his house for experimenting with Docker and other hosting options a while ago and had all the necessary infrastructure, including a domain. First a Database container was set up using the SQL-script created earlier with most external connection closed off, so that only the server backend could access. This guaranteed that no unfiltered requests could make it to the server.

Security in Docker, we learned, is not the greatest to start with. Most of the containers are wide open for setup and testing purposes. This means that you have to make sure all the right settings have been configured to only allow intended requests in and out. Unlike spring security though, this isn't a slog through dense XML; the Dockerfiles are quite manageable, and many tutorials exist online to guide you through how to close up the various images' ports, and many other security holes.