Web Technologies for Artificial Intelligence

Webtechnieken voor KI

Project short name (CD)

Coin Desk

Group [18]

1. [Gerson de Kleuver 12054674]
2. [Jason Lam 11844035]
3. [Ingur Veken 11886366]
4. [Scipio Akova 12062774]

Teaching Advisor: [Mathias Kirkeng]

FNWI/Faculty of Science

1 February 2019

1 Introduction

In this technical report we will aim to justify our decisions and elaborate on our goals and design choices, as well as giving a clear overview of what our site has to offer.

Our initial goal was to replicate the successful old forum model and enhance it. We aimed at creating a modern looking forum over the course of a month that avoids many pitfalls other forums have. The pitfalls are as followed; an unclear web design that causes visual clutter and does not aid the user in navigation of the site, posts and topics being lost in the forum due to maze-like navigation with no good way of finding posts, and unclear display and order of information. On top of these we also wanted to implement some nice features, like a night mode for example.

Our forum was going to have a tech related topic. We wanted to create a forum that people can easily use at home or in a hurry.

As it stands we can now present a fletched out forum with clean design and a good user experience. The Coin Desk forum is easy to navigate, has clear instructions and responses and features various additional functionalities. Furthermore we will structurally go over our design choices which includes, the use cases, general requirements and web design. Then we will explain our implementation, show our results after which we will finish with the discussion where we reflect on how we worked and how it worked out.

2 Design

Our goal was to make a forum that was clearer and more navigable than older forums. Further required functionalities included a distinct difference between the guest, user, and admin interfaces and their own abilities. Guests are the unregistered visitors; they can read posts, use all navigation tools and further enhance their search and user experience by making use of other features such as the search bar and Night Mode. Guests are however restricted from most other functionalities such as upvoting, creating a thread or commenting. This keeps the forum cleaner as anonymous people cannot clutter the forum with unrelated content, resulting in a better user experience. For a guest to become a member he must first register; This must be done by entering a valid email address and a strong password to login with, as well as a username which functions as a changeable nickname visible to everyone on the website. The user can use the forum as is intended, i.e. post, upvote, comment freely, and pick a profile icon. In case of a forgotten password, the user can use his email to reset it and enter a new one. The user can also change his icon and username at any time. Finally the admin, in addition to being a normal user, has an enhanced interface with some extra functionalities, e.g. to lock threads, so people can no longer post. Admins can also directly delete posts and entire threads with a click of a button to keep the forum even more clean.

2.1 Use case(s)

Our website has been developed for people interested in cryptocurrency and gives a platform to people who want to speculate, promote or simply want to talk with people about cryptocurrency. Since the cryptomarket is characteristically volatile our website stays up to date by making a system that enables users to dynamically add new threads with new topics.

2.2 General requirements

- Users classify their own submissions to the discussion
  A functionality that is required to make our forum as aforementioned dynamic. It is one of the most important features as it would enable us to keep the forum up to date without needing admins to constantly create new threads with new classifications.

- Users can filter user-submitted content based on these tags
  Needed to give the users a tool to let them see coins, or types of discussions they are interested in. This provides users with a more comfortable experience browsing around the website, finding threads that are interesting to them.

- User system and the ability to upvote, reply, use text markup and quote
  Provides all the basic necessities to create a forum people would actually want to use. The users system is needed to login, have distinct profiles and grant access to the features, such as quoting, that are widespread among other forums and thus are also included in ours.
  Users can sort on new, hot and top post along with the tag system.
  This (again) provides users with a more comfortable experience browsing around the website, finding threads that are interesting to them.

- Web api, that tracks popular coins, fastest rising/declining coins and provides basic information to use alongside the forum.
  On a forum that is centered around cryptocurrency, it would be a logical decision to provide users with simple news about the topic the site revolves around. Giving users a live overview makes sure users aren't constantly forced to [switch websites].

- Good admin system to keep the threads clean and enforce proper usage of the forum.
  Needed for moderation of the forum, deleting posts and locking threads.
  Administrators play a vital role in any forum as they keep it focused and make sure inappropriate content gets removed.

- All account functionality located in the navigation bar.
  As the site is mainly centered around discussion, and not users, the amount of profile options must be minimized, as they will only clutter the screen. Therefore all profile management must be done in the navigation bar.

2.3 Website/webapplication design

In the forum we wanted to build there are about 5 things that need to be represented on the database; threads, posts, upvotes, users, tags. On the homepage the user is greeted with a list of threads, these threads contain 1: a title, 2: an author, 3: text, 4: the date they have been submitted. Inside of a thread users can submit reactions (posts), these posts contain 1: a reference to the thread they are in, 2: an author, 3: text and 4: a date they have been submitted.

Since threads contain the same information that posts do (author, text, date), we gave threads a reference to a post (their OP). This way, 2,3,4 at threads could be replaced with 1 foreign key towards a post. Using a foreign key to an op makes it easy (and fast) to access the information that is needed to display threads on the homepage, without storing any information twice.

The authorID that is stored inside a post is a foreign key towards a user. The text is submitted on posting and the date is fetched upon inserting the post in the database. The threadID is a foreign key towards the thread that post is located in. Although it is a little unorthodox to have 2 tables cross-referencing each other with a foreign key, storing the threadID inside a post makes fetching all posts that are contained in a thread extremely simple and clear.

Upvotes are not an attribute from a post or a user itself, but are a combination of a reference towards both. After all: a user can make multiple upvotes across different posts, a post can have multiple upvotes across different users. However, a post can not have multiple upvotes from the same user. Therefore the primary key of an upvote (the combination of the authorID and postID), is always unique.

Users have an email address, a username, a BCRYPT protected password and a reference towards their user icon.

This table contains user ids as primary and foreign key, with it a BCRYPT protected 128 bit token is stored with an expire_time timestamp. This makes it impossible for attackers to bruteforce a password reset link. The expire_time makes it possible to ensure send password reset links are only valid for 24 hours.

Tags are the topics you can bind to a certain thread when creating it. Their primary key is the combination of their name and the foreign towards the thread they are attributed too. This way a tag can be contained across multiple threads and a thread can contain multiple tags, whilst keeping a unique primary key.

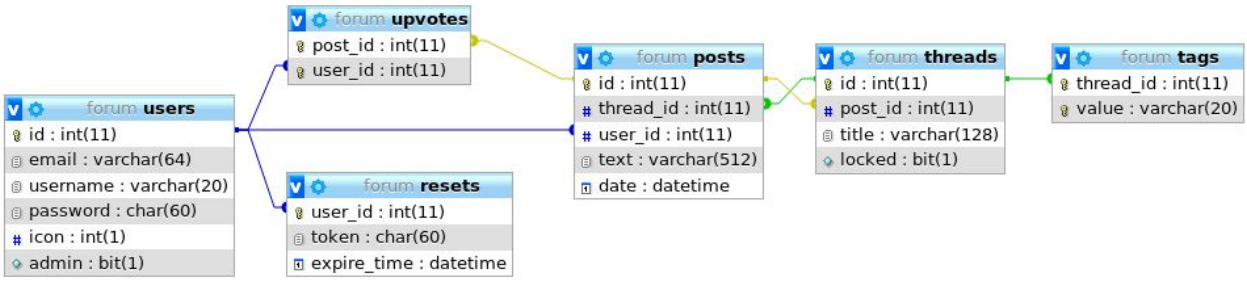
For the addition of admin roles, we also store 2 extra columns in the database. 1 column in the users table that stores whether a user is an admin or not. The other column

is in the threads table and keeps a record of whether a thread is locked by an admin.

Posts, threads and users obtain an auto-incremented value once new ones are made. This, always unique, auto-incremented value is their primary key.

The frequent usage of foreign keys makes for a stable database, as you can't insert new rows in tables that are missing data. I.e.; you can't create a thread without a post, you can't make a post without a user, you can't make an upvote without a user and a post and so on. The use of foreign keys also makes it easy to delete data that is dependent on each other. I. e.; when you delete a thread all of it's posts get delete, when you delete a post all of it's upvotes get deleted, when you delete an OP the thread gets deleted and so on.

The primary keys of all tables are unique and there are no double columns in any tables, making the database compliant with 1NF standards. All subsets of data that apply to multiple rows of a table are placed in separate tables, these tables are referenced using foreign keys, making the database compliant with 2NF standards. All columns in tables are dependent on their primary key, removing any transitive dependency. This makes the entire database compliant with 3NF standards.



an overview of the database

3 Implementation:

How is the code structured? Where do we find the various developed functionality?

The main ‘container’ files are stored in the main directory. We have stored the .css and .js files in /css and /js respectively. The database connect.php file and other authentication related files are stored in the /auth directory. The files related to resetting your password using an email link are stored in the /reset directory. The top navigation bar also functions as the account configuration settings and also has its own directory /nav. All other files related to displaying and processing content on the forum such as threads, posts and search results are stored in the /content directory, which in itself is also divided in multiple subcategories to maintain structure.

How is it prevented that the same code is repeated in several places?

Connections to the database are only made using the /auth/connect.php file. The file dbio.php contains all database related functions that occur in multiple files. Wherever these functions are needed, said php file is included before calling those functions. All head metadata is stored in the head.php file. This way we made sure we didn’t repeat code in several places.

Which queries are most important for the operation of your website?

The pdo sql queries used in the /auth/* files and /content/pdio.php contain the most important queries related to inserting/deleting items in the forum database, but other sql queries used in the /content directory are also important to ensure a good functionality.

How is the website made user-friendly?

The site has been made responsive using html and css. The main layout could be divided into 3 layouts; A desktop, tablet and mobile phone layout. It’s also possible to resize your window or rotate your phone to seamlessly switch between the layouts. This contributes a lot to the user-friendliness of the forum.

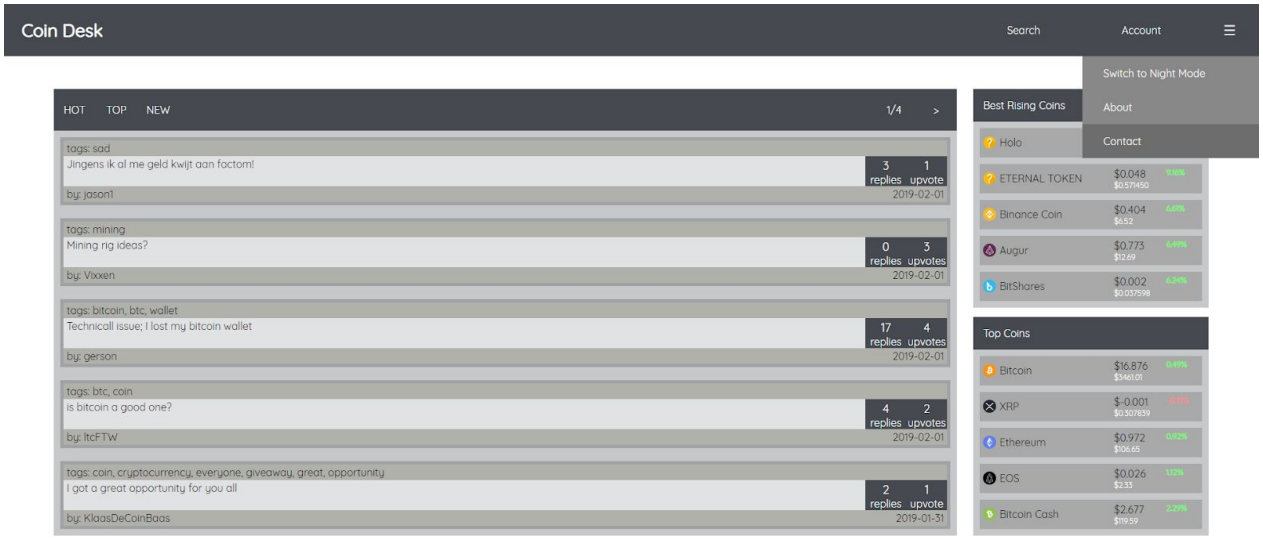
Another feature is our form.js redirecting, made using .js (and JQuery). This makes it possible to run form action files without redirecting the user, and even allows us to display error/success messages on button click. This is also used to redirect a user to the thread he just created for example.

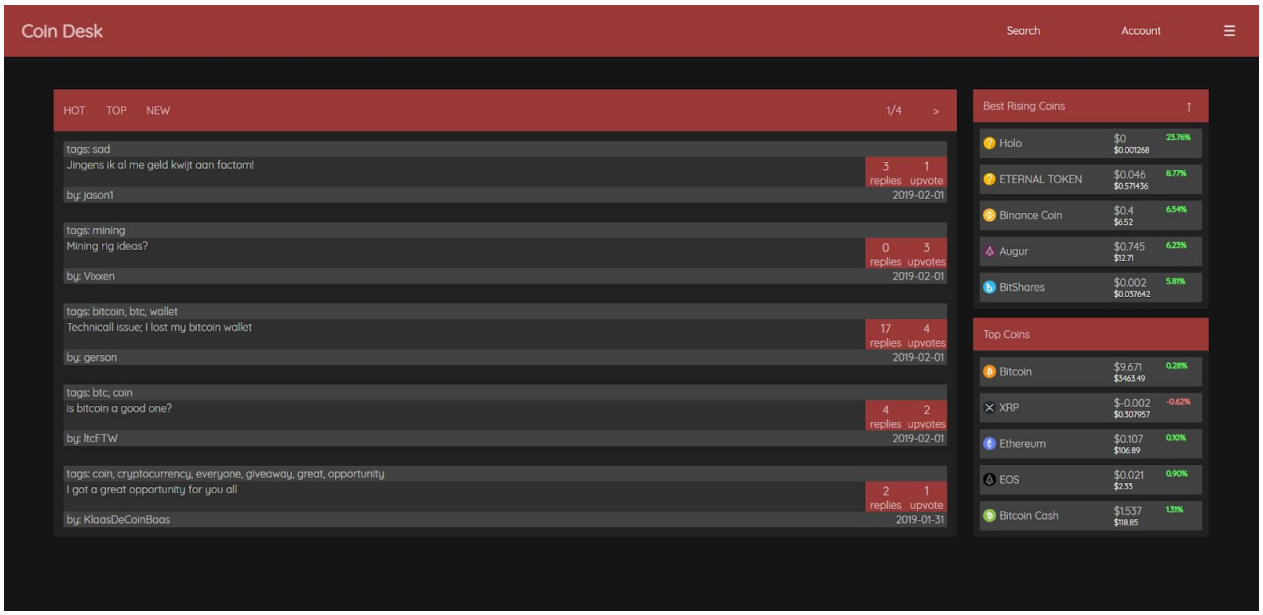
How is the website protected against malicious users?

Every user input gets filtered before it's processed by the server using a filter() function included in /auth/connect.php. Php Data Objects are used to interact with the database where queries are properly prepared and variables properly inserted to avoid sql injection.

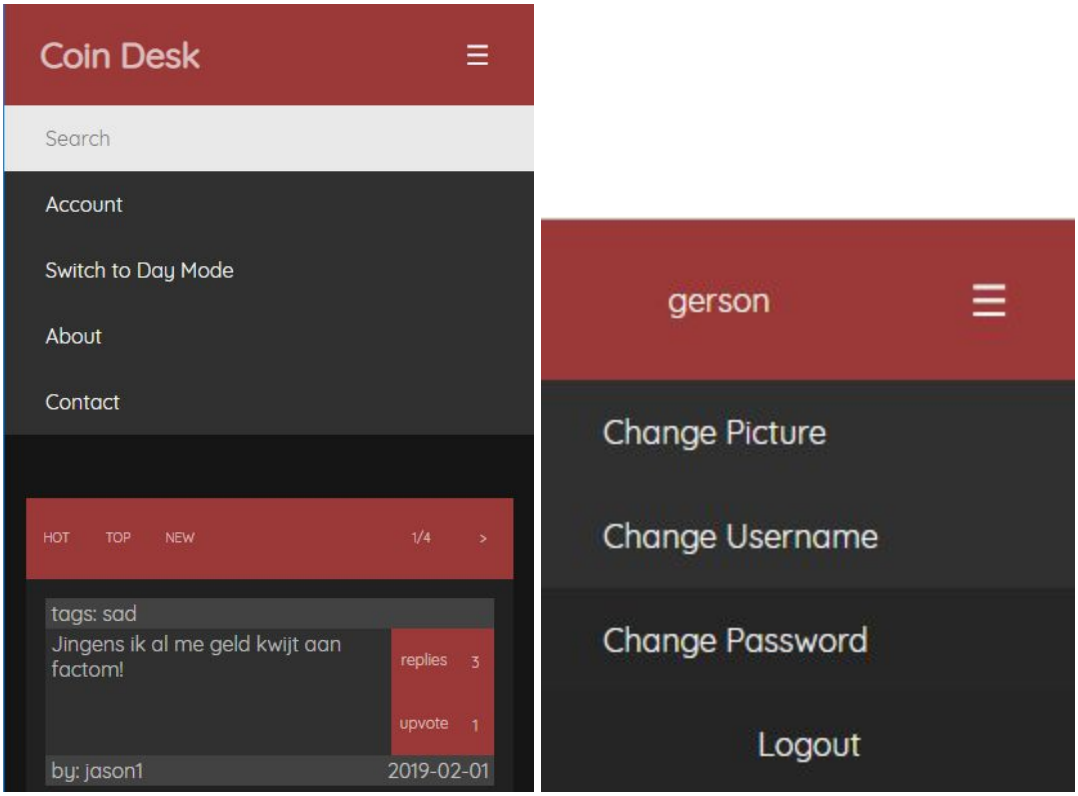
4 Results:

Full size site on Day/Night mode:



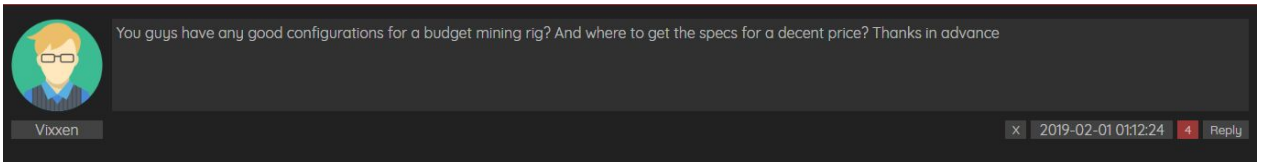


Responsiveness and navbar functions:



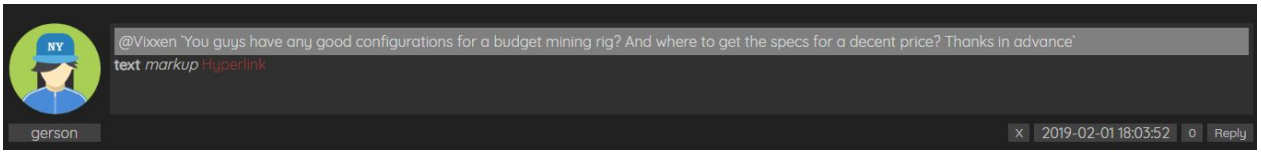
Sorting:

Upvotes:



Clickable tags and admin functions:

Text markup and replying:



5 Discussion

In hindsight we have learned quite a lot, our site's functionalities have been refined a lot. Keeping the site responsive was increasingly difficult as we added more functionalities and boxes. The database was also overhauled multiple times, for example to make it 3NF which initially lead to various issues. Restructuring in general took us a lot of time and we changed our work attitude during the second week. Our CSS also took more time than we liked.

Things we learned included using more *flex* and percentages to create boxes which helped with the responsiveness. We also learned to use global variables better and stored variables in the URL instead of storing it all in the database.

After working out the proper database and the restructuring adding features became much easier. Our functions all worked pretty fast due to usage of standard formats we made. As of now everything works as planned and due to the existing structure it would be very easy to add additional features. The standard formats and current structure make the site all in all very maintainable; changes are easily made and new problems are very visible.

With better prior knowledge we could have definitely started better. Sometimes we put too much time in small details and we did not always cooperate well. This led to different parts of the site and functions initially being very hard to integrate with each other with a chosen database, as we constantly used dummy databases individually, which lead to functions not always connecting well. We should have definitely worked out the database first instead of leaving that for later. Taking all that into account, however, it did not end up being a total failure after all.