Hacker News

Technical Design Document

[**1. Project Overview**](#_62qaskwk0v2p) **3**

[1.1 Objective](#_30j0zll) 3

[1.2 Assumptions](#_ao533ivhbzz7) 4

[**2. Technical Specifications**](#_3znysh7) **4**

[2.1 Drupal Architecture Diagram](#_2et92p0) 4

[2.2 Web and Database Servers](#_1t3h5sf) 6

[2.3 Drupal Specifications](#_17dp8vu) 6

[2.4 Theming](#_3rdcrjn) 6

[2.5 Modules](#_t6be0hxlhum8) 7

[**3. Implementation**](#_4so2ifsi94ug) **7**

[3.1 Setup Drupal](#_o0t4agfnx4xr) 7

[3.2 Create custom module](#_wsauh8r7o7jl) 7

[3.2.1 Custom stylesheets and javascripts are also developed for this module.](#_x02z5ib8ayly) 7

[3.2.2 Other files in the module](#_7xj0s46djj09) 8

[3.2.3 Controller - NewsController.php](#_glvkboftu7z) 8

[3.2.4 Templates - For the view layer we have used our own template for module](#_jbchj1gjgzwu) 10

[3.5 Controller](#_ti8ubt7fuofo) 10

[3.6 Module](#_lfdtgfmbv5i3) 11

[3.7 View](#_hx95ncw8f7en) 12

[3.8 Database](#_57lqm1xjy0xc) 13

[3.9 Screens](#_nhbq03g08sy7) 15

[3.9.1 Home Screen](#_w1it13cw3q7w) 16

[3.9.2 Top Stories](#_5kaz9wqri9dv) 16

[3.9.3 New Stories](#_l9ayb4mh9l59) 18

[3.9.4 Best Stories](#_kgyfrnop73h3) 19

[3.9.5 Comments](#_6p1j4ahd5pqa) 20

[3.9.6 User Details](#_4qjfp6fye0na) 21

[3.9.7 User Stories](#_2ytksde2uebq) 21

[3.9.8 User Comments](#_s7wj92c59fzi) 22

[**4. Logs**](#_d3x9nsjhmqm7) **23**

[**5. Repositories**](#_kdkqgpys5x58) **23**

[**6. Security**](#_46r0co2) **24**

[6.1 Risks](#_111kx3o) 25

# 1. Project Overview

In this assessment, we are going to build a Hacker News clone.

We will be required to write a custom site that periodically fetches the latest news items using the hacker news API.

The documentation for the API can be found here: https://github.com/HackerNews/API

## 1.1 Objective

Our application should:

● Fetch the news items periodically for top stories, new stories and best stories.

● Create and store the news item in a SQL database

● You should also fetch all the comments associated with the news item and store them in the

database

● Create a view that displays the content as per the hacker news site with the associated detail page

that includes the item and its comments.

● MVC/ similar architecture is encouraged.

● You may use any PHP framework of your choice. Drupal 8 is preferred however use what you are most familiar with

Styling, documentation and design will award bonus points, as the core focus of the assessment will be the quality of the code.

## 

## 

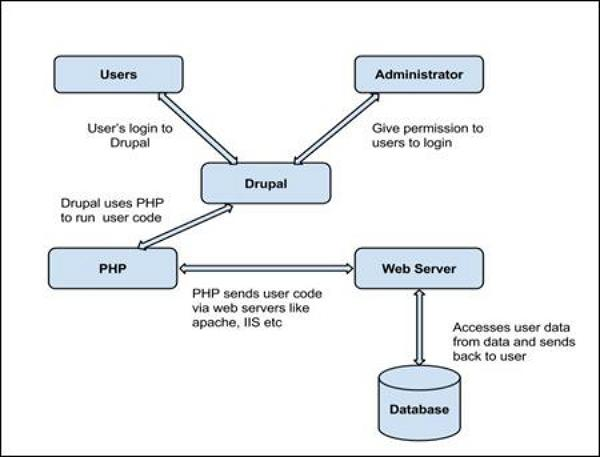
## 1.2 Assumptions

Our site should only fetch the top 500 stories and build from there, there is no need to go much further than that.

# 2. Technical Specifications

## 2.1 Drupal Architecture Diagram

Drupal is a platform for web content management which is a powerful tool for building simple and complex sites. The following diagram shows the architecture of Drupal:



**Users**: The user sends a request to a server using a web browser acting like clients.

**Administrator** − Administrator can provide access permission to authorized users and will be able to block unauthorized access. Administrative account will be having all privileges for managing content and administering the site.

**Drupal** − Drupal is a free and open source Content Management System (CMS) that allows organizing, managing and publishing content and is built on PHP based environments. Drupal CMS is very flexible and powerful.

**PHP** − Drupal uses PHP in order to work with an application which are created by users. It takes the help of a web server to fetch data from the database.

**Web Server** − Web server is a server where the user interacts and processes requests via HTTP/HTTPS (Hyper Text Transfer Protocol) and serves files that form web pages to web users. The communication between the user and the server takes place using HTTP/HTTPS.

**Database** − Database stores the user information, content and other required data of the site. It is used to store the administrative information to manage the Drupal site. Drupal uses the database to extract the data and enables to store, modify and update the database.

## 2.2 Web and Database Servers

We will use XAMPP for total development of this project. That will have Apache as our application server and we will use MYSQL for the database.

Drupal 8 Database is a highly normalized database schema. For every new entity, two tables are created in the database (actual and revision).

* Database - MariaDB
* Database version - 5.5.5-10.3.16

## 2.3 Drupal Specifications

The website is using the Drupal version, Database version, PHP and NGIX versions specified below:

|  |  |
| --- | --- |
| **Drupal** | **8.8.6** |
| **Database System** | **MariaDB** |
| **Database System Version** | **5.5.5-10.3.16** |
| **PHP** | **7.1.30** |
| **Web Server** | **apache/2.4.39** |

## 2.4 Theming

Businessplus\_lite is a contributed theme which will be used in this website. Businessplus\_lite is using Symfony’s twig template engine. Many regions are defined in this theme and twig files are using them to show the contents.

Below is a list of some of the template files: -

Path - **/themes/**Businessplus\_lite

* block.html.twig
* page.html.twig
* page-title.html.twig
* node.html.twig

## 2.5 Modules

**2.5.1 Contributed Modules**

* Admin Toolbar

**2.5.2 Custom Modules**

* Custom

# 3. Implementation

## 3.1 Setup Drupal

We have used drupal 8 to implement the hacker news website. We have used the traditional approach to install drupal where we download the drupal core from drupal.org, create blank database and then run the setup.

Drupal makes changes in files and databases while installation.

Site name - Hacker News

Database name - hacker\_news

Admin username - wunderman

Admin password - thompson (not recommended to put in TDD, we are adding because this is an assessment)

## 3.2 Create custom module

After setting up drupal the next step is to create a custom module which will do the full implementation of business logic.

The only used module in this project is Custom module

Path - **modules/custom**

### 3.2.1 Custom stylesheets and javascripts are also developed for this module.

Stylesheets

Path - **modules/custom/css/custom.css**

1. Custom.css

JavaScript

Path - **modules/custom/js/custom.js**

1. Custom.js

### 3.2.2 Other files in the module

1. custom.info.yml
2. custom.module
3. custom.routing.yml
4. custom.libraries.yml

### 3.2.3 Controller - NewsController.php

Path - **modules/custom/src/controller/NewsController.php**

**namespace Drupal\custom\Controller;**

**class NewsController {**

**public function stories( $type ) {**

**$output = get\_hacker\_stories($type);**

**return array(**

**'#theme' => 'stories',**

**'#items' => $output,**

**'#title' => 'Our '.str\_replace("\_"," ",$type)**

**);**

**}**

**public function comments( $id ) {**

**$story\_details = get\_story\_details($id);**

**$output = get\_nested\_comments($id);**

**return array(**

**'#theme' => 'comments\_list',**

**'#items' => (array)$output,**

**'#story\_data' => (array)$story\_details**

**);**

**}**

**public function homepage() {**

**$output = array(**

**'top\_stories' => array\_slice(get\_hacker\_stories('top\_stories'), 0, 5, true),**

**'new\_stories' => array\_slice(get\_hacker\_stories('new\_stories'), 0, 5, true),**

**'best\_stories' => array\_slice(get\_hacker\_stories('best\_stories'), 0, 5, true),**

**);**

**return array(**

**'#theme' => 'homepage',**

**'#top' => $output['top\_stories'],**

**'#new' => $output['new\_stories'],**

**'#best' => $output['best\_stories']**

**);**

**}**

**public function author( $name ) {**

**$output = get\_author\_details($name);**

**return array(**

**'#theme' => 'author',**

**'#items' => $output,**

**);**

**}**

**public function authorStories( $name ) {**

**$output= get\_author\_stories($name);**

**return array(**

**'#theme' => 'stories',**

**'#items' => $output,**

**'#name' => $name**

**);**

**}**

**public function authorComments( $name ) {**

**$output= get\_author\_comments($name);**

**return array(**

**'#theme' => 'comments\_list',**

**'#items' => $output,**

**'#name' => $name**

**);**

**}**

**}**

### 3.2.4 Templates - For the view layer we have used our own template for module

List of files:-

1. stories.html.twig
2. comments-list.html.twig
3. homepage.html.twig
4. author.html.twig

## 3.5 Controller

We have used only one controller to perform all the actions as the website contains less than 10 pages and it's easy to maintain.

Controller name - NewsController.php

Path - **modules/custom/src/controller/NewsController.php**

One object is created for each page which takes the data from the custom.module file and sends it to template files(twig).

## 3.6 Module

Custom.module is the file where all hooks are written.

Below is the snippet of hook\_theme.

/\*\*

\* Implementing hook\_theme().

\*/

function custom\_theme($existing, $type, $theme, $path) {

return array(

'stories' => array(

'variables' => array('items' => array(), 'title' => '', 'name' => '')

),

'comments\_list' => array(

'variables' => array('items' => array(), 'title' => '', 'story\_data' => array(), 'name' => '')

),

'homepage' => array(

'variables' => array('top' => array(), 'new' => array(), 'best' => array())

),

'author' => array(

'variables' => array('items' => array())

),

);

}

This function declares the theme variables.

All other functions can be checked in the file.

path - **modules/custom/custom.module**

## 3.7 View

For the view layer we have used our own template for module

List of files:-

1. stories.html.twig
2. comments-list.html.twig
3. homepage.html.twig
4. author.html.twig

We have used 4 twig files for the theming and have written code in a way that we reuse the files. For example the stories.html.twig file is used for 5 different pages in our project

//////////////// stories.html.twig starts /////////////////

{% if name %}

<h3>Stories by {{ name }} </h3>

{% endif %}

<table id="story-table">

<thead>

<tr>

<th>Title</th>

<th>Points</th>

<th>Comments</th>

<th>Author</th>

</tr>

</thead>

<tbody>

{% for news in items %}

<tr>

<td><img src="{{ url('<front>') }}themes/businessplus\_lite/images/forum-icon.png" width="24" height="24">&nbsp;&nbsp;&nbsp;&nbsp;

<b><a href="{{ news.url }}" target="\_blank">{{news.title}}</a></b>

</td>

<td>{{ news.score }}

</td>

<td><a href="{{ url('<front>') }}comments/{{ news.story\_id }}">{{ news.count }} </a>

<td>

<span><a href="{{ url('<front>') }}author/{{ news.author }}">By {{ news.author }}</a></span></br>

<span>Time {{ news.time |date('d/m/Y H:i:s')}}</span>

</td>

</tr>

{% endfor %}

</tbody>

</table>

//////////////// stories.html.twig ends /////////////////

All other files can be checked in the folder.

path - **modules/custom/template**

## 3.8 Database

We have created 4 custom tables in our drupal database.

* hacker\_stories
* top\_stories\_comments
* new\_stories\_comments
* best\_stories\_comments

Creating custom tables is not considered to be a very good practice but it depends on the requirement, for the current project the comments data is very large and if we do a content type approach it will create revision tables for every record, which will slow down our website going forward. but in the current implementation the performance of the website will not affect even with large amounts of data.

Below are the structures of the custom tables.

Table - hacker\_stories

Description - Stores total 500 stories

200 Top Stories | 150 New Stories | 150 Best Stories

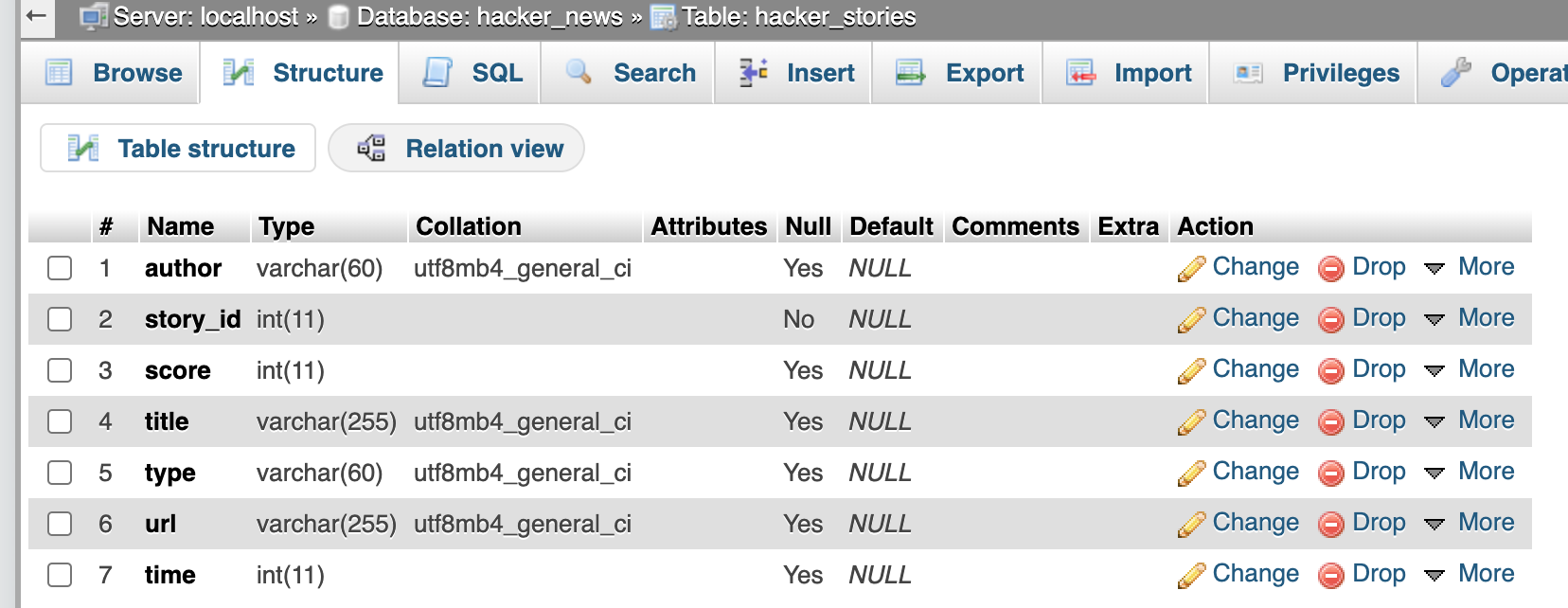
****

Table - new\_stories\_comments

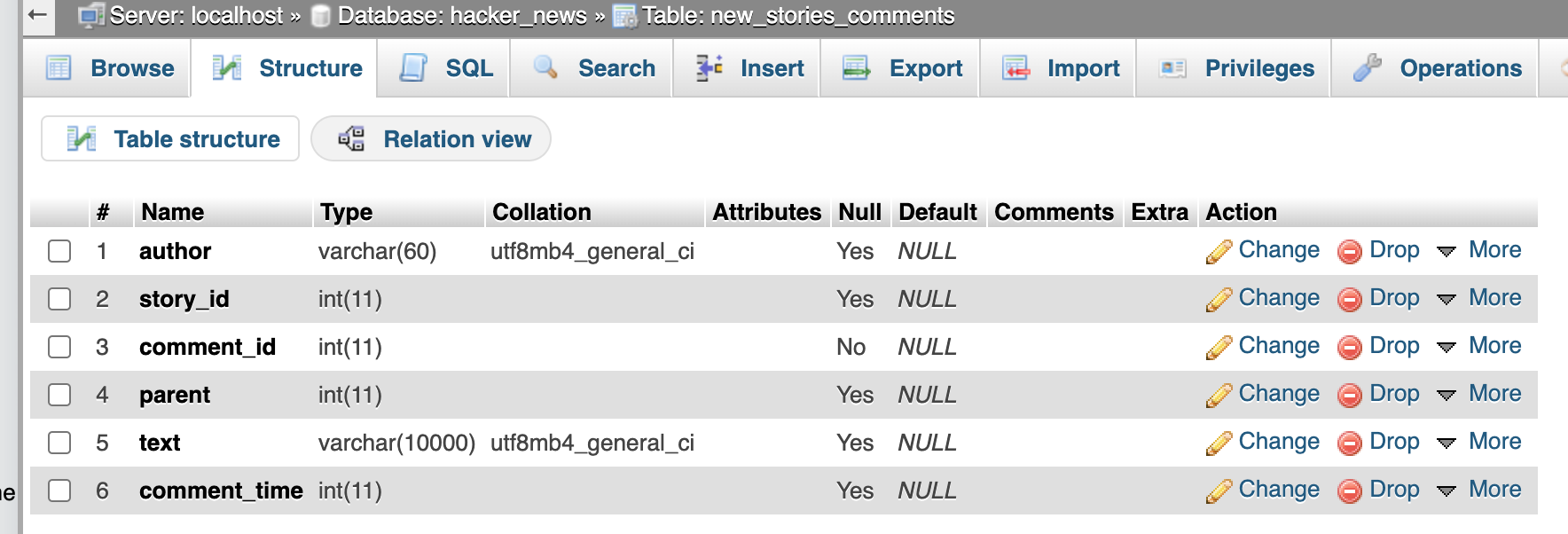
****

Table - top\_stories\_comments

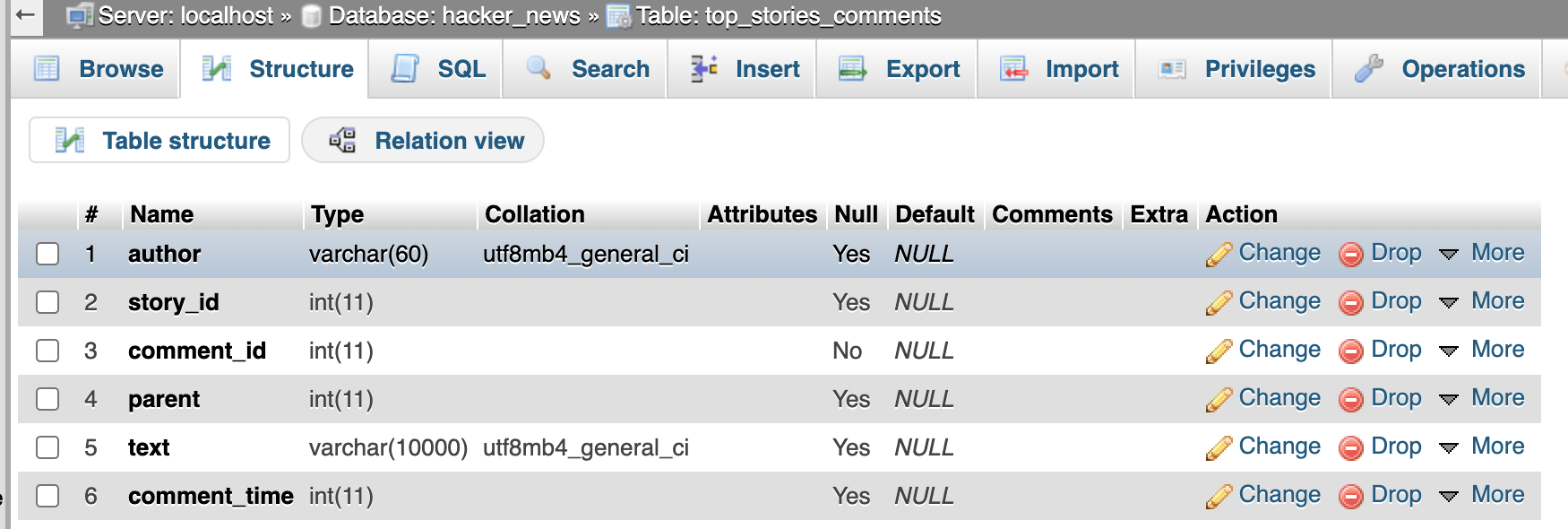
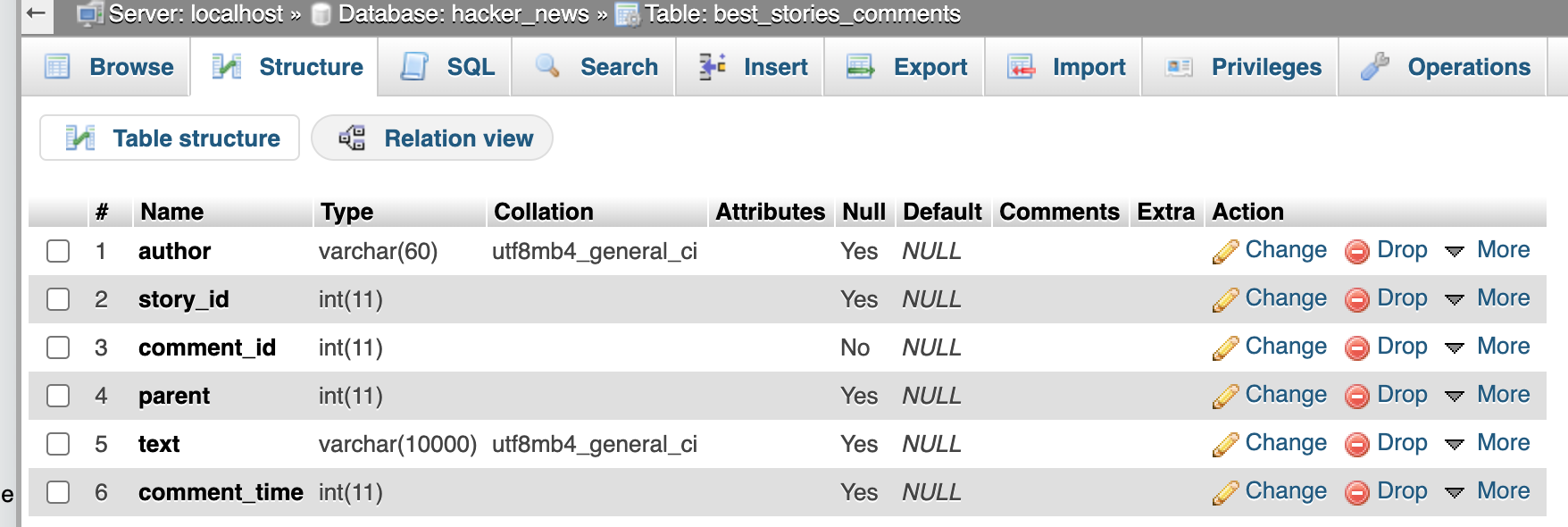
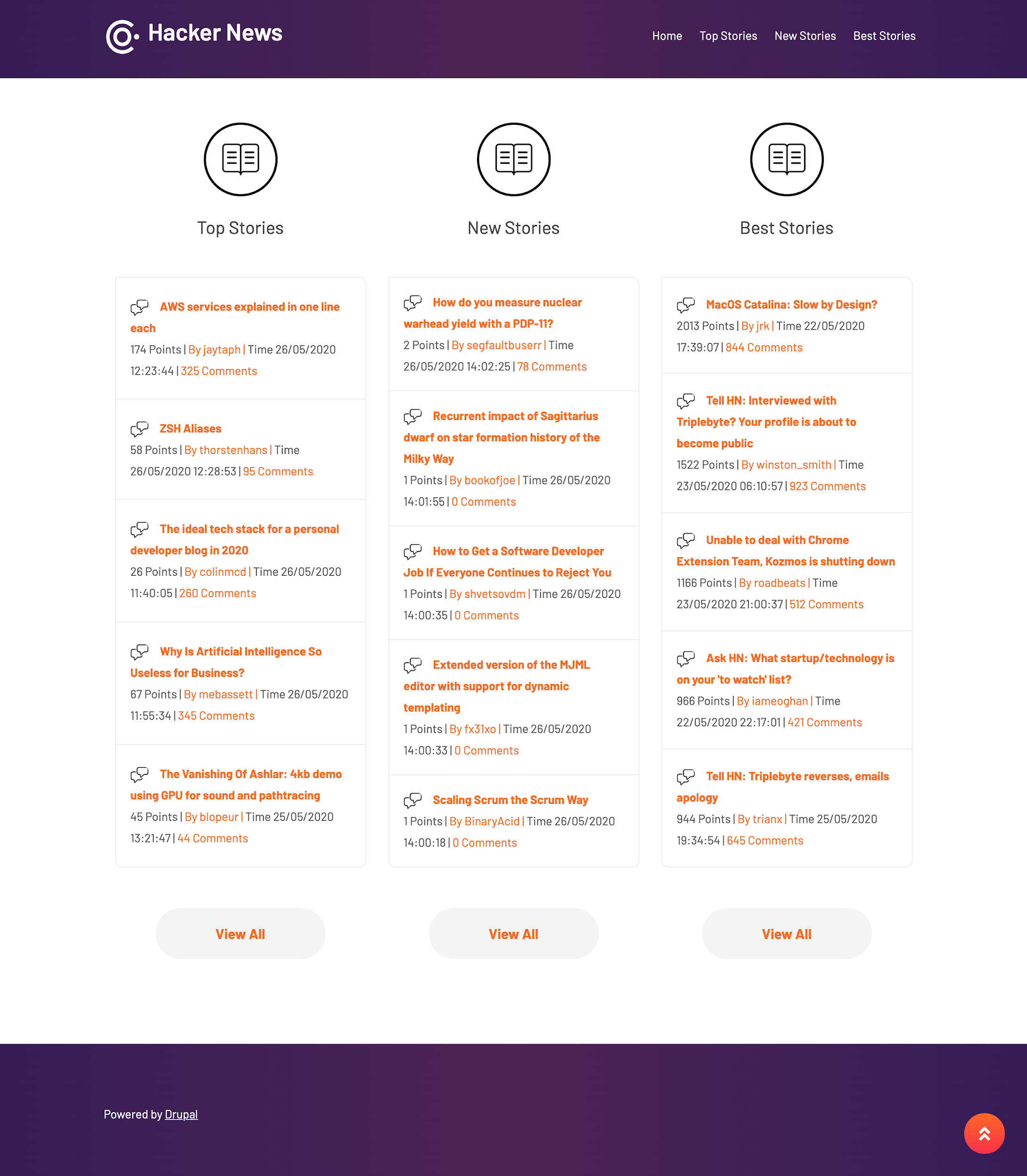
****

Table - best\_stories\_comments

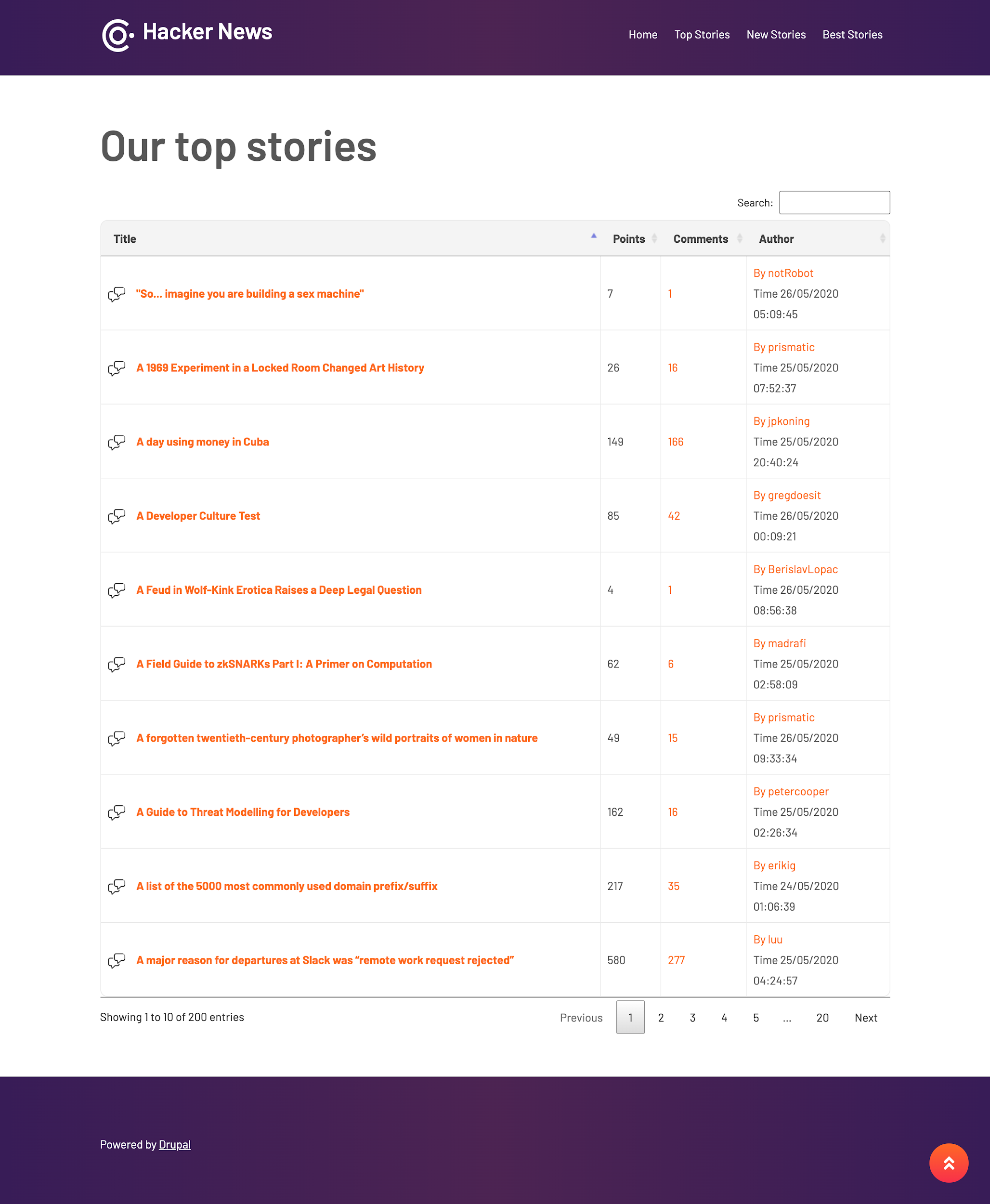
****

## 3.9 Screens

### 3.9.1 Home Screen

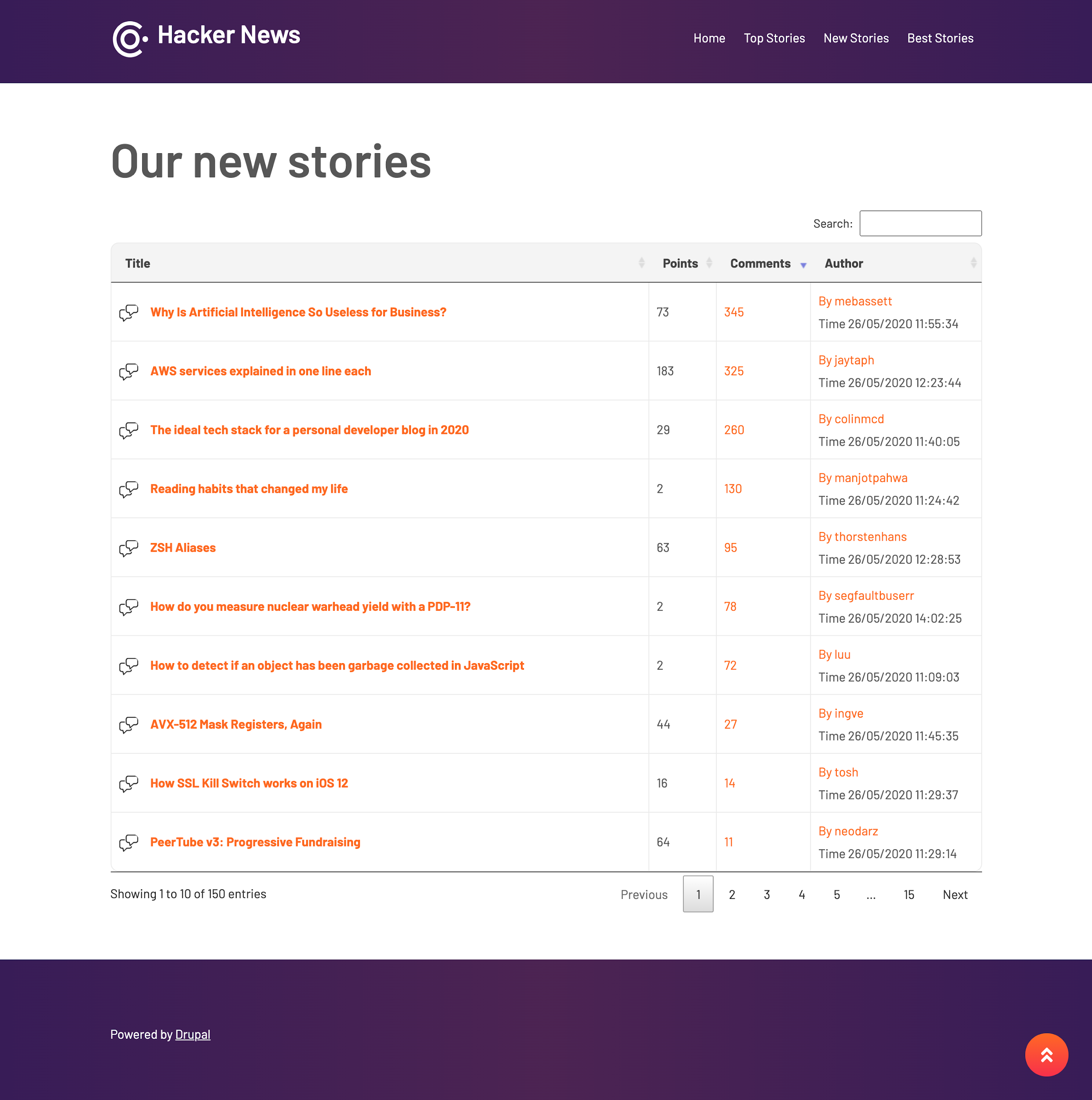
****

### 3.9.2 Top Stories

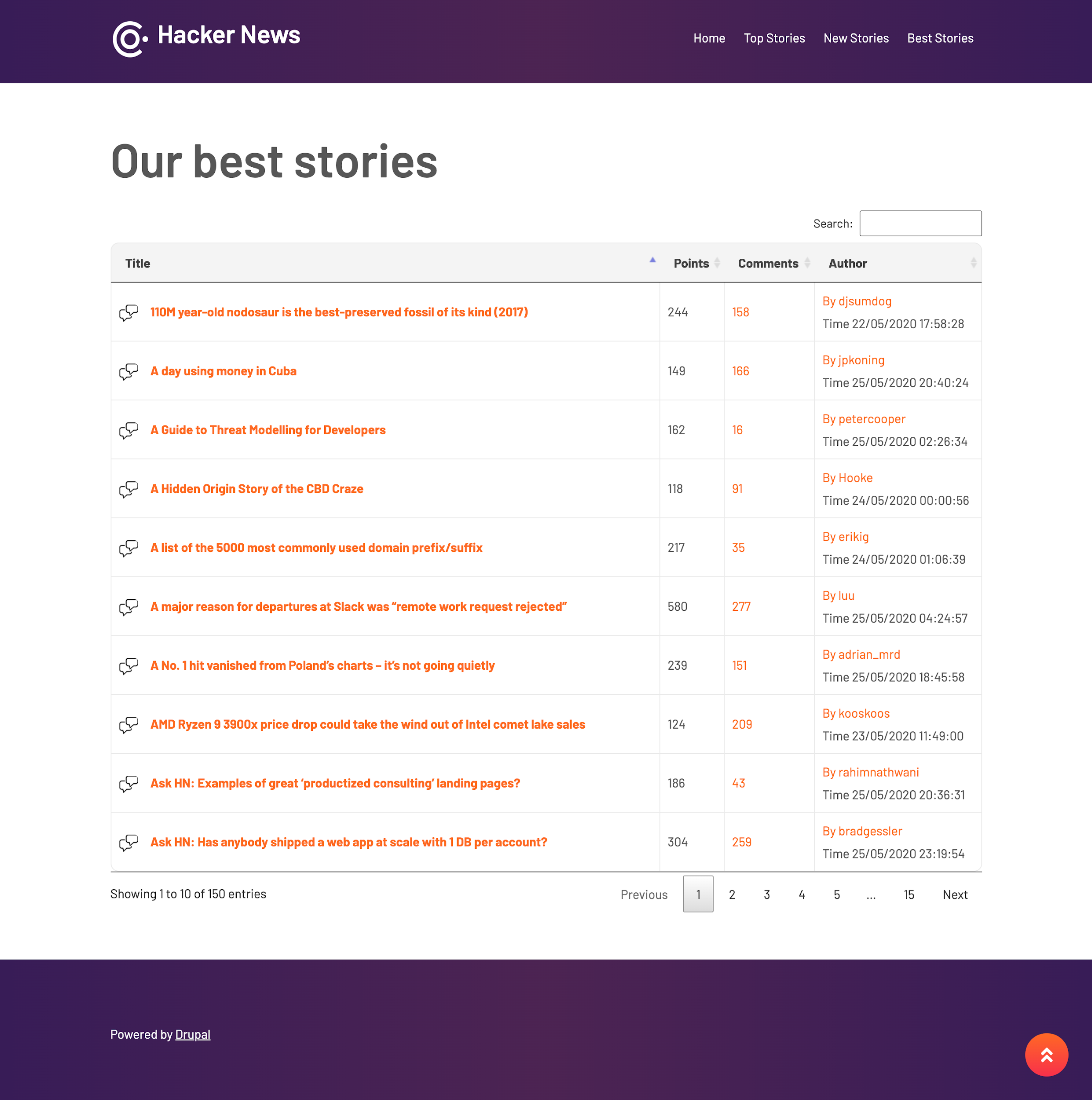
****

### 

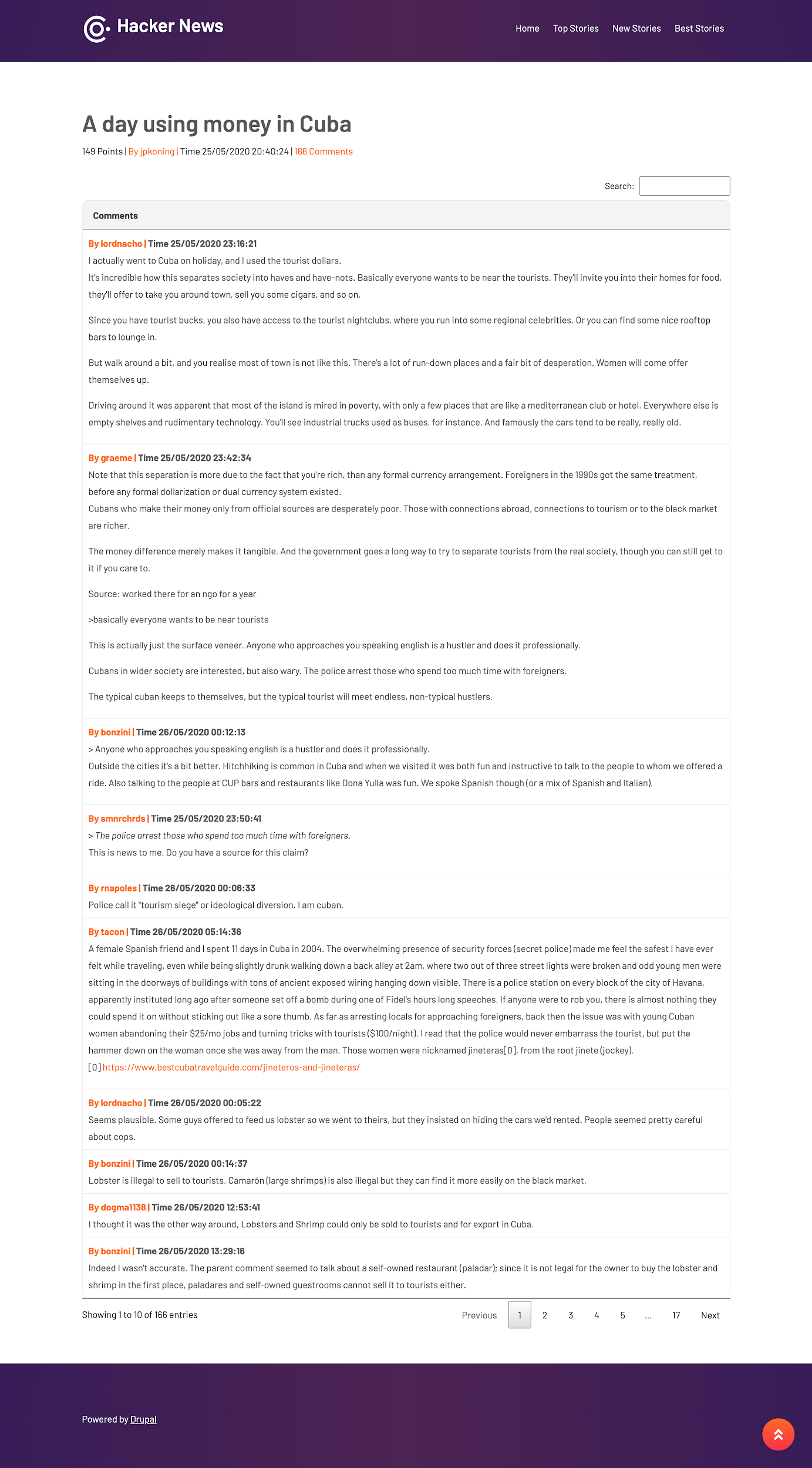
### 3.9.3 New Stories

****

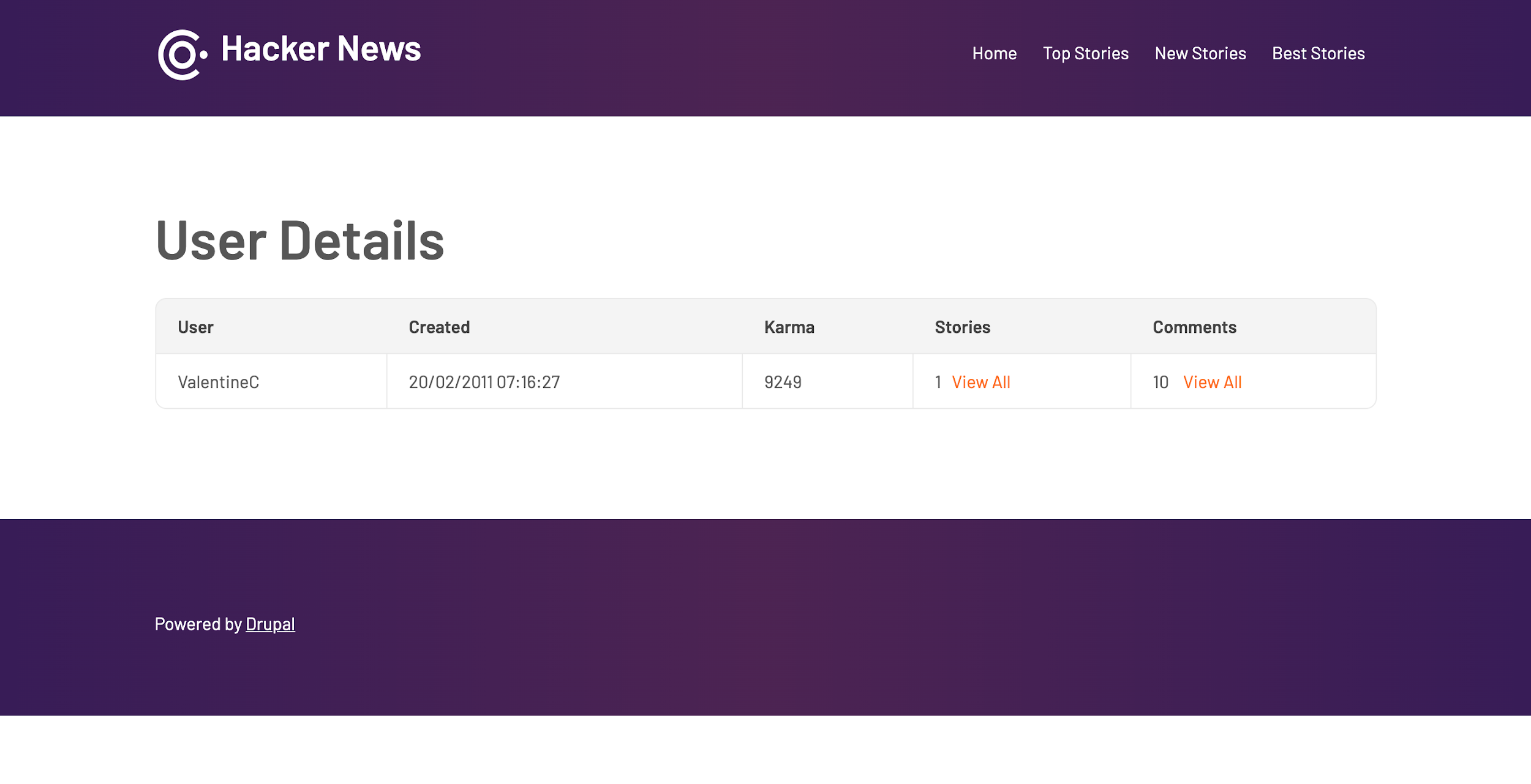
### 3.9.4 Best Stories

****

### 3.9.5 Comments

****

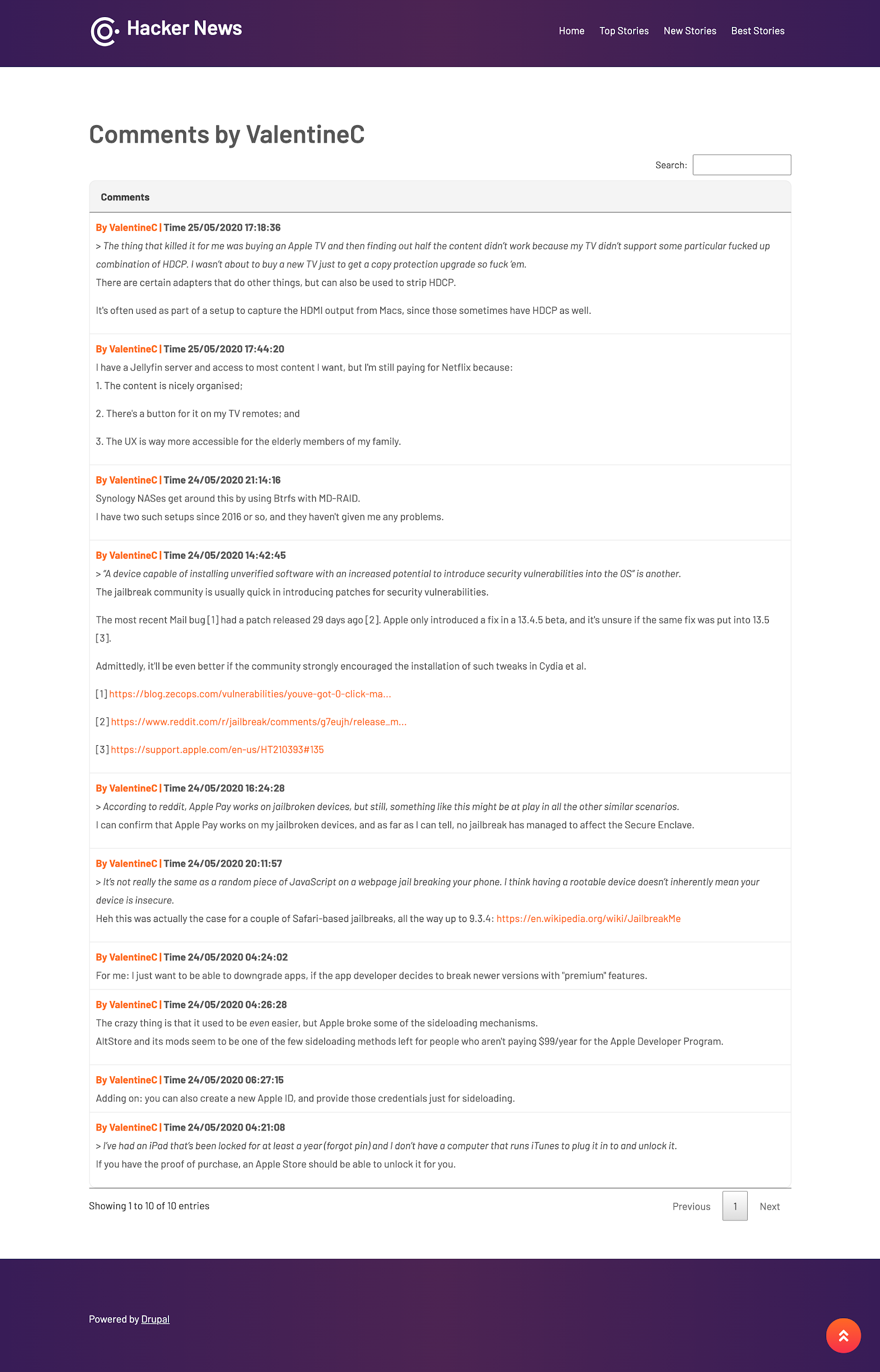
### 3.9.6 User Details

****

### 3.9.7 User Stories

****

### 3.9.8 User Comments

****

# 4. Logs

Drupal store logs in the watchdog table in the database.

The Database log lists all database queries performed by the server process. The database log shows all events of a pool of several active connections to the database.

A log line can contain information on the following:

* Connection that has been requested.
* Query that started to run, including the actual query, the system time and thread that requested the query.
* Connection that has been refused.

# 5. Repositories

|  |  |
| --- | --- |
| Repository name | URL |
| hacker-news | <https://github.com/shubho8/hacker-news> |

**5.1 GIT Push Procedure**

Step 1 – git init

Step 2 – git status

Step 3 – git stash <if we have modified files in git status>

Step 4 – git pull origin <your branch name>

Step 5 – git stash app <only in case we have stashed our changes in Step 3>

Step 6 – git add <your file paths>

Step 7 – git commit -m “your commit message”

Step 8 – git push origin <your branch name>

**5.2 Rollback Plan:** We are using GIT which is a version control system and comes up with its own rollback management. There are mechanisms available with which we can roll back to previous/earlier versions if something goes wrong.

# 6. Security

Drupal is a secure CMS and application framework that stands up to the most critical internet vulnerabilities in the world to prevent the worst from happening. Drupal is mature, stable and designed with robust security in mind.

* Secure Access - Out of the box, Drupal account passwords are encrypted--salted and repeatedly hashed--when they are stored in the database. Drupal can support a wide variety of password policies such as minimum length, complexity, or expiration.
* Granular User Access Control - Drupal can give administrators complete control over who can see and who can modify every part of a site. Drupal operates based on a system of extensible user roles and access permissions. Administrators can create user roles and give them specific, limited permissions.
* Preventing XSS, CSRF, and other malicious data entry - Drupal’s Form API ensures that data is validated and scrubbed before entry in the database. The system tests that user-entered data--and even the form fields themselves--match prescribed, expected formats and values. Tokens are injected into each form as it is generated, to protect against potential CSRF attacks. Drupal’s database abstraction layer performs additional security checks on data as it is written to and retrieved from the database.
* Brute Force Detection - Drupal protects against brute-force password attacks by limiting the number of logins attempts from a single IP address over a predefined period of time. Failed login attempts are logged and visible via the administrative interface. Drupal can also be configured to allow administrators to ban individual IP addresses and address ranges.
* Mitigating Denial of Service (DoS) Attacks - Drupal’s extensible cache layer comes pre-configured with basic page, JavaScript, and CSS caches. Individual components of Drupal are typically cached as well, and granular expiry is a common feature. This multi-layered cache architecture is extremely resistant to high volumes of traffic.
* Addresses OWASP Top 10 Risks - Drupal includes features that address all of the Open Web Application Security Project’s top ten security risks, a list of the most commonly seen risks in practice.

## 6.1 Risks

Drupal being an open source application there can be many vulnerabilities like

1. Less secure than other commercial application - The nature of the open source model is that open source projects make their code available to anybody, hackers can use the publicity of these exploits to their advantage
2. Don’t come with extensive support - Many times we have to rely on the solutions suggested by the community, if any module is not maintained by any developer and we are still using it then it is a high risk to the website.

Author - Shubho Banerjee

Date - 28th May 2020

Email - [shubho.b701@gmail.com](mailto:shubho.b701@gmail.com)

Github link - <https://github.com/shubho8/hacker-news>