Missile Alert Joke App

Requirements and Specifications Document

URL: grifare.net/examples/jokes/missile-alert/index

Author: Armagan Tekdoner

Created on: February 2018

Requirements

Create an app that

1. displays a deadline in hours, minutes, and seconds in user's local time. The deadline will not be updated on events. That means it will remain the same even if the user refreshes the page or navigates back and forth within the app;
2. incorporates a minutes-seconds countdown clock that reaches zero at the displayed deadline, in local time. The countdown process will not be updated on events. That means it should keep counting down without ever going back to its original value, even if the user refreshes the page or navigates back and forth within the app;
3. provides interactive features to do something before the given deadlines reached. However, buttons should not trigger events unless certain conditions are met;
4. displays user geolocation;
5. emails data to the author about usage statistics;
6. has a story with alternative endings; and
7. is entertaining

Intent

Practising the following topics

Name	Technology
Date	JavaScript
Session Variables	JavaScript
Session Variables	PHP
Validations	HTML5
Validations	JavaScript
Validations	PHP
Captcha	PHP
Geolocation	PHP
Mailer	PHP
Animations	CSS3
Animations	jQuery
Conditional Redirects	JavaScript
Conditional Redirects	PHP
Responsive Design	CSS3

Synopsis

Inspired by the false missile attack alarm issued in Hawaii on January 2018, this mock application now prioritises prevention of false alarms, but overkills. An awfully-designed user interface along with overzealous validations, converts a one-button-task into stomach ache. The legalese found almost in all websites today has also been sarcastically ridiculed by exaggerating the existing nonsensical norms.

Drive-through Outline

The headline in the presentation page reads there is a missile detected, heading to user's city. The user, performing the operator role, is to issue an alert about the situation; but, the unnecessary validations will not be easy to overcome. Moreover, the verbose app is full of threats and warnings. In case of false alarm, the operator would be the sole responsible party for any inconvenience that might be caused. On the other hand, the threat might prove to be real. Distractive and unnecessary images positively contribute to the frustration the app creates.

There are 2 possible outcomes, depending on what action the operator takes.

If the operator decides and succeeds to issue alarm:

The obstacles are numerous. The hard-to-decipher captcha that exists for no reason, prevents the alarm button from working. Furthermore, unless the hidden legalese is displayed and a checkbox indicating the operator accepts the unacceptable terms and conditions is checked, the button would do nothing. After validations, a prompt forcing the operator to type "yes" to be able to proceed, is sickening. If the operator succeeds to issue the alarm under annoying warnings and sarcastic wording, the missile attack **warning will prove to be false** seconds later and a cancellation button appears. Seemingly that one-button task which just creates confusion, is not designed to do anything. North Korean hackers will say the final word.

If the operator fails to or decides not to issue alarm:

Annoying imagery and countdown will continue. Once the allocated time is over, the screen reads "no input device", imitating a consequence of **missile impact**.

Certainly, the app can be closed anytime by simply closing browser. All jokes live within the boundaries of the app and no prank will be encountered after the user leaves.