Web Caffeine

Überkraft – Final ditital**crafts** Project
19 Aug 2016

Überkraft

Overview

"Überkraft” is a (fictional) German Engineering company that provides electrical equipment throughout the world, with offices in several German cities in addition to Atlanta, GA.

This web application is designed to serve as a *Service Portal* for employees of Überkraft. Primary objectives of the application are to:

- Function as a *multi-language website.* Pages can be displayed either English or German.
- Collect and store *repair requests* in a database.
- Provide advance warning of a possible trend in product failures by automatically sending a warning email to the Service Manager if the number of repair requests received in one day exceeds a pre-defined threshold.
- Provide graphs which shows recurring product failures.

Technical and Implementation Details

The web front-end contains almost no static content. Instead, content is stored in a postgreSQL database.

The user can choose the preferred language. The current page can be re-displayed in the required language at any time.

The application in designed in such a way that that changes to existing text content only requires updating the applicable rows in the database – no HTML updates are needed to change existing text. The website need not be taken offline to do this.

Backend processing is coded in the Python programming language. Routes are used in the application to direct page flow.

The application requires users to be logged in before some pages can be accessed. When appropriate, the user is directed to login before accessing restricted pages. User ID’s and *encrypted* passwords are stored in the database.

What I Learned

I expected that the most difficult part of writing this application would be managing multiple languages. Python “dictionaries” were constructed to contain the required English or German information. All that was necessary to switch from English to German, or vice-versa, was to switch the dictionaries. I did have to implement some character encoding in order to support letters with umlauts, such as the U in *gemütlichkeit.*

Error handling was a substantial part of the coding. Care had to be taken to ensure that required fields were populated, etc.

The most challenging part was being able to accommodate user authentication and switching languages *simultaneously*; the routing required to do this was much more complicated than I originally anticipated.

For a demonstration, see my portfolio at http://www.web-caffeine.com.

Web Caffeine
www.web-caffeine.com
678-464-4937

Tim Sanders
tim.sanders@web-caffeine.com
http://www.linkedin.com/tsanders30004
http://www.github.com/tsanders30004