Unit 3 - Task 1

Intro

An information system is an integrated set of components for collecting, storing, and processing data and for delivering information, knowledge, and digital products. Business firms and other organizations rely on information systems to carry out and manage their operations, interact with their customers and suppliers, and compete in the marketplace.

Executive Support Systems

For this section, we will be using an executive support system for a supermarket's management staff.

Features

Description

The system allows you to make decisions based on your staff's behavior. It creates graphs and charts, as well as calculates real time information.

Data Type

The executive support system uses, gathers and analyses and summaries the key internal and external information used in the business.

Who is it Used by

Executive support systems are designed to help senior management make strategic decisions.

Hardware Required

- Server - This is to receive all of the data, draw the graphs, and store the data in a locatable area
- Data Collection Units
    - For example:
    - Barcode scanners
    - Handheld stock checkers
    - Punchcard machine

Software Required

- Server software to gather data and make graphs
- Client software to gather the information

Telecomunications Required

- Local network
- Internet (Optional) - This is in case you want to factor in external information in order to process the data

Functions

Input

Data is input into the system via the client devices that connect to the server and input various information such as how much stock is left, for example.

Collection

Data is collected via the client devices by using the interface on those. This interface is usually graphical, and has various buttons and text boxes in order to help data collection. The data is then collected by the server over the local network or via the internet every specified period of time.

Output

The system will usually output the information in a series of graphs and charts, as well as real time information that allow you to make informed decisions.

storage

The data is stored on the server, usually on a SQL server in database format.

Processing

The data is processed by the server in order to make the graphs and charts. Sometimes the data is specially formatted, such as capitalizing names and such.

Retreival

Many systems will allow you to export the data as an array of formats from PDF to CSV to JPG.

Control and Feedback Loops

The data could be used to help to improve the system. You can observe the data to see what are the most common mistakes, and use that information to try to improve the input of the system. For example you could decide that there are too many mistakes in the inputting of the stock levels, and you may want to have them manually checked on a monthly basis.

Open or closed

This system can be entirely closed. You may, however, want to incorporate external data in your system, for example: External prices, or profit from rival companies for comparison.

Guild Wars 2 Spidey

Description of what the system does

Guild Wars 2 Spidey is a system that uses the Guild Wars 2 API to monitor prices on the trading post from within the game. The system also does further calculations. Simply put, the best items in the game are made up of components. Most of these components can be purchased from the trading post(TP). Most of the components can be crafted/refined from other components, for example, cloth scraps can be refined into bolts of cloth, and those can be used to make cloth armor, along with other components. The problem is that all of the components are purchasable, and if you want to make a specific piece of armor, you don't know if it's cheaper to buy the raw components, or buy them pre-refined. Another solution that Guild Wars 2 Spidey (GW2Spidey) offers is the ability to list items by the largest positive difference between the total cost of an items components, and the amount that an item can be sold for on the trading post including transaction costs and listing fees. This allows you to see from crafting which item, you can make the most profit.

Data the system uses

The system pulls information from the GW2 API (Application Programming Interface). This is a service offered by GW2 that allows you to pull down data about the game in a format that is easy to import into a program (JSON). The particular data pulled(Think of TP like eBaY): - lowest selling price - used to calculate cost - It does not pull lowest buying price, I wish it did, because then you could choose instant sales over profit - components of an item - used to break down an item into it's components and figure out the cost

The people who use the system

The system is used almost exclusively by the players of GW2 in order to attempt to make money off the crafting system. The supporters of the system are not needed, as AreaNet (The author of GW2) set the APIs up to update automatically.

Hardware Required

- Server to pull information from the trading post database (Provided by AreaNet)
- Webserver to host the GW2Spidey website

Software Required

- Webserver software

- They need to program a back end
- They need to program a front end (The website is opensource, code here https://github.com/rubensayshi/gw2spidy .

Telecommunications required

- Internet - To allow people access to the site externally

Input

The data in the system in input entirely by the API. This means that ArenaNet are responsible for any inconsistencies with the data, but from my expeience, the data is very accurate as it is taken directly from the TP.

Output

The data is processed and output onto the web frontend. The web frontend lists the important data for crafting, and also individual items, displaying graphs for their price over time.

Storage

Data received from the GW2 API is stored using MySQL as you can see here : https://github.com/rubensayshi/gw2spidy/search?q=SQL&ref=cmdform

Processing

The data is processed on the server using PHP. The server uses PHP to order the data by specific values eg. profit; so that you can find the item that makes the most profit, again; for example. The server also makes graphs using logged data.

Retreival

The data is retreived by the website from the API over HTTP in JSON format. The user receives the website marked up in HTML in an easy to read format.

Control and Feedback Loops

GW2Spidy allows you to send them feedback via github. On GitHub, you can submit issues that can be solved discussed and solutions can be proposed by anyone; all you have to is accept their pull request.

Open or closed?

This system is open, as it takes it's data from the ArenaNet APIs and relys on that to run.

TheScoop

Description

TheScoop is a website I helped to produce in the summer of 2013 as a product of Young Rewired State and an entry to the Festival of Code, and came as a runner up in the category "Best Example of Coding", and had special mentions by Dallas Campbell in "Best in Show".

Bragging rights asside, the aim of TheScoop was to create an interface usable by someone who has fallen behind on news for a select period of time, and wants to catch up quickly, in a slick, fast webpage.

The website is live here: http://TheScoop.io/

Features: - Natural language form (Complete the sentence to configure the website) - System for scraping data from the BBC News website - Stores data in a MySQL database - Data uses a special algorithm that ranks news based on the length of time it's been on the front page, and it's top stories page rank - Provides APIs for anyone to access the data - Uses the open text summarizer (Popularized by summly) and readability API to condense news stories to and rough percentage of themselves (This is also available in API form)

Data the system uses

The system pulls data from the guardian APIs, and scrapes the BBC News RSS feed.

Who uses the system

The system was designed to be used by people who wanted to quickly catch up on news.

Hardware Required

- Webserver

Software Required

- Webserver - Apache2 2.2.22
- MySQL database server - MySQLd 5.6.15
- PHP - PHP5 mod for apache2
- Web files - eg. HTML pages and PHP scripts
- PHPMyAdmin - For testing, browsing, and editing the database

Telecommunications required

- Internet - To allow people access to the site externally

Input

Data is input from the guardian APIs, and scrapes the BBC News RSS feed

and stored in the database after being processed

Output

The data is processed and output onto the web frontend. The web frontend lists the important news from the past specified period of time. The website uses only all minified JavaScript and only a single CSS file. There are almost no images on the page (other than the loading circle). This optimises the page to run exetremely quickly and smoothly. The output is also shrinkable to mobile widths, without changing the layout of the page.

Storage

Data is stored inside a MySQL database, and is administrated by PHPMyAdmin installed on the server. The database in deleted after the output extends 2 months to save server space, as it litterally saves a copy of every link on the BBC News RSS feed from every 5 minutes.

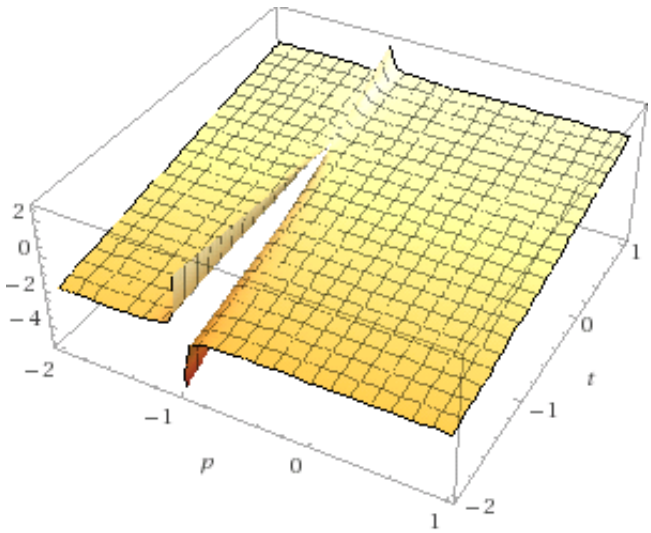
Processing

The data is processed in PHP as it is scraped from the BBC News RSS feed. PHP calculates a score of a story's popularity by using the total time it has appeared on the top stories feed incorparating it's page rank, also.

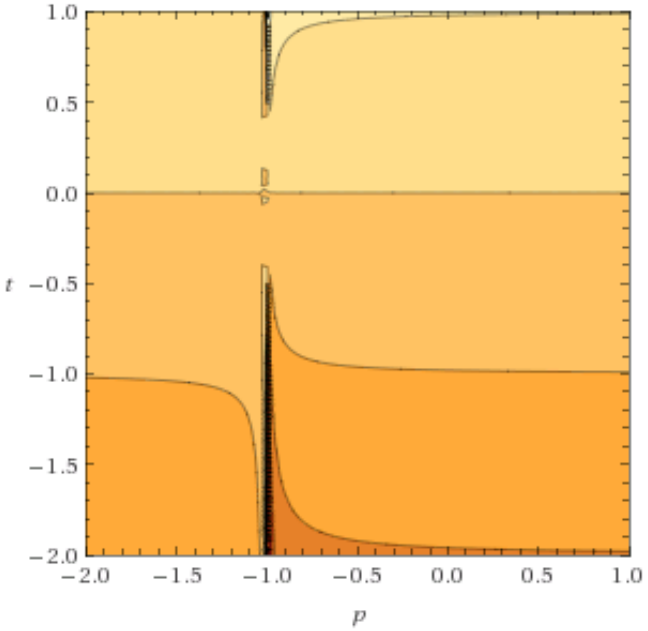
The actual algorithm used is: [time on front page(mins)] / 5 * (1 + 0.02 / ([position on page] + 1))

Which can be simplified to $\frac{1}{5}\left(1+\frac{0.02}{1+p}\right)t$

Here's a pretty graph that displays the build up of points while increasing values of t and p independantly:



Here's a colour plot because WolframAlpha:



Retreival

The data is retreived by the website from the inbuilt API (That was the part I coded) over HTTP in JSON format using WebSockets in JavaScript. The user receives the website marked up in HTML in an easy to read format.

Control and feedback loops

While we were at YRS, there were a lot of people to help up to debug. We'd go around giving out our source code and an URL to our development server, and challenge other programmers to break the website. Only one person managed to find a bug, and it was a shell injection exploit that allowed him the execute *NIX commands on our server from his web browser. This proved low risk, as the UID that was running the code had no permissions to write anything on the server, but while fixing the bug, it turned out to be a forgotten shell escape in the OTS API for the 'Ratio' field.

This was, in my opinion, a very good way to get technichal feedback, as it targets and identifies the most important issues in a friendly environment where we can trust users not to exploit the system before it goes live.

For more trivial feedback, we asked UI testers to submit issues to GitHub.

Open or closed

This application is open (As required by the rules of the festival of code) as it pulls data from the BBC News RSS feed, and also from the guadian web APIs. It also offers some APIs to the public. These are demonstrated here: thescoop.io/test.html. This code is also open sourced, code available here: github.com/edlongman/thescoop