P3: News Aggregator

Due May 26 by 11:59pm **Points** 100 **Submitting** a website url

A News app for our client: While we were building our SurveySez app, the client asked for another app that will also extend the capabilities of the Client Protosite (**sp16**).

This new app will be built by your team!

News Syndication, Categorization and Caching: For our third group project we'll be building a PHP application that provides categorized news pages from syndicated RSS feeds. These pages must come from feed data stored in a database. The feed data must be cached on a session level so news pages generated during a current session are stored so the data does not need to be retrieved from the remote survey beyond the first page hit.

Integrate RSS data into Feed Pages: You'll be building pages that integrate customized RSS feeds into the client protosite. The RSS feed must carry the look and feel of the protosite, plus include an RSS feed from Google, etc. The pages should carry both the news stories and images if possible that accompany the news items.

A News Page with a Link to 3 News categories, with custom 9 custom feeds: The entry point to the application should be a page linked to the protosite as a page with a name such as "**News**" and then link to pages or an application that accommodates at least 3 news categories, with at least 3 custom feeds under each. An example would be a category of **Music**, under which were feeds named **Reggae, Blues** & **Jazz** *(That would be **one** of the 3 required categories & custom feeds)*

2 Database Tables Required: You must store your **feed** and **category** information in database tables. Therefore you'll need a minimum of 2 tables. These tables could **greatly** resemble the books & categories tables from the Joins lesson on the class website!

List/View: The Category/Feeds pages will make great use of techniques we've learned such as List/View. Be sure to study the example List/View pages in the **demo** folder of **sp16**!

Session Caching: Every time we hit an RSS page provided by a third party we introduce delay and consume network resources. Therefore we'll require you to cache the data once retrieved for a number of minutes (example, 10) that will enable your app to store data for that feed via a session and **not** retrieve the feed data again until the cache period is up. Each news feed should have a separate cache mechanism.

Cache Time/Date & Clearing: If RSS data is cached, the retrieval time/date should be displayed on each feed, along with the programmatic ability to clear the cache for **that** feed.

You must also be able to programmatically clear the cache of **all** feeds. (Delete all existing session data)

Project Requirements

Project Document: Besides a working application, you must also show your program design pieces visually on a project document shared by your team and also with my email: **williamnewman@gmail.com (mailto:williamnewman@gmail.com)**

Visual Elements Required: Your program design document should identify an IPO chart and/or a flowchart of how the application will operate and process data.

Work Log: Your project document must include notes from each team member at the bottom of the document identifying what was done, how long it took each and every time work is done on the project. Here's an example notation:

2/12/2016: (Bill) Updated P3 Aggregator assignment to include updated requirements. (.5 hours)

Each member should place the most recent work at the top, so that each work log entry descends in chronological order.

Source Code via Repo Required: Please show your source code by placing your code in a repo and providing a link to your repo. If you wish to use a private repo, you can use my GitHub username and share it with me: **newmanix**

NEW - work on separate branches in repo: There's enough work now to divide and conquer the code as well. Be sure to branch your code and only merge back to master with code that works.

NEW - use issues in repo: Utilize the **issues** **(https://guides.github.com/features/issues/)** capability in GitHub. At least use issues to track items that need attention so others know you need help!

Standards - The app should be built as nearly as possible to **PSR-1** **(http://www.php-fig.org/psr/psr-1/)** and **PSR-2** **(http://www.php-fig.org/psr/psr-2/)** standards

phpDocumentor Documentation Standards Followed: The additional element we'll add to this project is the requirement to document your code internally per phpDocumentor standards. View the lesson named **Documentation** on the class website for details!

Roles & Responsibilities

Since this is a larger project, consider splitting up the responsibilities so everyone is productive.

Roles: Be sure the work to be done is split up as evenly as possible, with each member taking on multiple **roles** **(http://theprogrammersparadox.blogspot.com/2011/06/roles-and-responsibilities.html)** as required. These roles could could fulfill duties such as:

- **DB Schema Design:** Build the tables, joins and load data
- **Visual Design:** build the HTML page mockups and partial repeating HTML fragments
- **Research:** Research feeds, resources & techniques necessary. In this case, having separate individuals research the RSS and the use of Sessions to store data
- **Documentation:** Making sure the code is documented properly and completely
- **Code Review:** Check for standards compliance and consistent documentation
- **Repo Management:** Now that we're branching code, someone should manage merges and conflicts
- **Testing:** Testing the code to be sure it succeeds in many conditions, including upon input of bad data
- **Project Management:** Making sure all items are completed on time, and no important pieces are missing
- **Application Design:** Building the fundamental programmatic pages
- **Page/Data Architecture:** Determine the pages & data required for the app

Early/Middle/Late Roles: The roles identified above can be distributed so work is done effectively at every stage of the project. We'll define early/middle/late as the following:

- **Early:** Before code has been written. Research, visual and db design, experimentation, goal setting and algorithms can be the early focus
- **Middle:** Everyone should be coding - hopefully in different areas of the app
- **Late:** Developers can code review each other's work, and perform testing and documentation

Example Division of Work: Below we'll take a 3 person team and identify some roles designed to occupy each team member during early, late & middle stages of a project:

1. Member #1: Early - Visual Design, Project Management Late - Testing, Documentation
2. Member #2: Early - DB Design, App Design Late - Project Management, Code Review
3. Member #3: Early - Research, Repo Management Late - Code Review, Repo Management

All members code in middle stage: The middle stage is not mentioned above as all team members would be coding at that stage.

Extra Credit: You can gain up to **30 points** extra credit in building the capability to **add** feed data through an administrative interface. The best starting point for this piece is the **demo_add.php** page in the **demo** folder.

Extra Extra Credit: You can gain up to **30 points** extra credit in building the capability to **edit** feed data through an administrative interface. The best starting point for this piece is the **demo_edit.php** page in the **demo** folder.

Submission: Your submission for this project is the link to your project space, which should have links to all relevant artifacts and a link to your working application!