* **Folder ‘Trending\_11.15.2013’**

1. **connectToDB.php**

* This script connects to the database. The following information is needed to connect to the database.
  + Hostname: “localhost” (since the scripts are being run on local machine using wamp server).
  + Username and password of phpMyAdmin.
  + Name of the database that was created while setting up phpMyAdmin.
* This file will be included in all the files that need to access the database for,
  + creating tables (script 2)
  + populating the database with data (script 3)
  + querying the database to get required information from the database (scripts 4, 5, 6, 7, 8, 9)

1. **createTables.php**

* This script builds up the database and needs to be run just once to build the tables in database from scratch. We create two tables, Event and Showing with the columns described below. For ease of distinguishing the tables from other tables of the database, these tables have a prefix ‘trending\_’. All the data related to an *event* or a *showing* is obtained from XML file:

<http://payments.cinequest.org/websales/feed.ashx?guid=70d8e056-fa45-4221-9cc7-b6dc88f62c98&showslist=true&>

* **NOTE:** For simplicity, this XML file shall be referred to as ‘URL-1’ in future documentation

**URL-1:** <http://payments.cinequest.org/websales/feed.ashx?guid=70d8e056-fa45-4221-9cc7-b6dc88f62c98&showslist=true&>

1. **trending\_Event**
   * EventID
   * Name
   * Duration
   * ThumbImage
   * EventImage
   * InfoLink
   * TabletLink – Separate queries are written to add values to this column manually.
   * BitlyLink – Separate queries are written to add values to this column manually.
   * TotalSold – Total number of Sold tickets of all *showings* falling under this *event*.
   * TrendingStatus – This column has either value 0 or 1, based on whether the ShowingStartDate HAS or HAS NOT passed the current date and time. Once the ShowingStartDate passes the current date and time, the TrendingStatus of the *showing* changes from 1 to 0.

**NOTE:** To test the scripts, the current date and time is set arbitrarily as: March 06, 2013 12:53 PM.

1. **trending\_showing**
   * ShowingID
   * EventID – This is the same EventID of table Event. Each of the showing of Showing table belongs to one of the events of table Event.
   * ShowingStartDate
   * ShowingEndDate
   * ShowingDuration
   * ShowingLegacyPurchaseLink
   * ShowingVenueID
   * ShowingVenueName
   * Available
   * Disabled
   * Sold
   * InProcess
   * RushStatus

* The following four columns contain the ticketing information obtained from the Agile ticketing API. https://prod5.agileticketing.net/API/Admin.svc/XML/EventGetSalesSummary?appkey=d09b3053-1817-4665-b41f-e1987e14433a&userkey=0fd55f05-bf17-4b48-bbff-12cd842830de&corporgid=2198&eventOrgID=2229&eventID=<ShowingID>
  + Available
  + Disabled
  + Sold
  + InProcess
* **NOTE:** For simplicity, this API shall be referred to as ‘URL-2’ in future documentation.

**URL-2:** <https://prod5.agileticketing.net/API/Admin.svc/XML/EventGetSalesSummary>?

appkey=d09b3053-1817-4665-b41f-e1987e14433a&userkey=0fd55f05-bf17-4b48-bbff-12cd842830de&corporgid=2198&eventOrgID=2229&eventID=<ShowingID>

* RushStatus – This column has value either 0 or 1 based on if the *showing* IS NOT or IS IN Rush Status respectively. If a showing has Available tickets = 0, the RushStatus of the *showing* changes from 0 to 1. In other words, if there are none tickets available for a *showing*, that *showing* would BE in rush status, otherwise it would NOT BE in rush status.

1. **updateDB.php**

* If one is running the script for the very first time after creating of the tables in the database, this script populates the Event and Showing tables with following data.
  + The *event* and *showing* details are populated to respective tables from ‘URL-1’.
  + Along with *events* with *showings*, this XML file also contains *events* that do not have any *showings*. Such *events* aren’t populated to the Event table. Only the *events* that have at least one showing are entered in the Event table.
  + The ticketing information is populated to the Showing table from the Agile ticketing API ‘URL-2’.
  + TotalSold for any *event* is computed while scanning each of the *showing* under that *event*.
* If one is running the script when the tables are already populated with data, the script updates the tables with new information from the XML1 file and the Agile ticketing API.
* This script takes approximately 20 - 25 minutes to complete its entire process. The script takes such a long time because it makes an Agile API call for number of times equal to the number of *showings*, which is a very large number.
* **Specifics common to scripts 4 through 7:**
* Each of the scripts 4, 5, 6, 7 generates an XML containing various details related to *events*, *showing* and ticketing, as required.
* The scripts have name with prefix ‘generateXML\_’ followed by suffix that shows what kind of information the generated XML will contain.
* Once the script is run, the XML generated XML will be stored in the same folder as the PHP file and will have the name same as the suffix of the PHP file. For e.g.: PHP file named ‘generateXML\_ Trending\_Top5Showings.php’ will generate XML named ‘Trending\_Top5Showings.xml’.
* This XML file is also printed on the screen in the form of PHP array, after the PHP script is run.
* **Common procedure followed in each of the scripts 4 through 8:**
* Query the database to give all the *events* and/or *showings* from the Event / Showing table and save the required attributes in PHP array.
* This PHP array is passed as an argument to a function ‘phpResponseToXML( )’, which converts the PHP array to XML data and saves it as an XML file.
* This is followed by the definition of the ‘phpResponseToXML( )’ function.

1. **generateXML\_EventsWithShowings.php**

* This script generates an XML with all the *events* that have at least one *showing*.
* It saves the attributes:
* EventID, Name, Duration, ThumbImage, EventImage, TabletLink for *event*.
* ShowingID, StartDate, EndDate, Duration, SalesMessage, LegacyPurchaseLink, VenueName for *showing*.
* The script queries the database to give all the *events* from the Event table and save the required attributes in PHP array.
* For each of the *event*, it queries the database to give all the *showings* from the Showing table for that *event*. The required attributes of these *showings* are concatenated to the same PHP array which stores the *events* information.
* Call the function phpResponseToXML( ) which saves the XML in EventsWithShowings.xml.

1. **generateXML\_EventsWithShowingsSummary.php**

* This script generates an XML same as above (script 4), but also includes the *showing* summary i.e. ticketing details of each *showing*.
* It saves the attributes:
* EventID, Name, Duration, ThumbImage, EventImage, TabletLink for *event*.
* ShowingID, StartDate, EndDate, Duration, SalesMessage, LegacyPurchaseLink, VenueName for *showing*.
* Available, Sold, Disabled, InProcess as ticketing details.
* The script queries the database to give all the *events* from the Event table and save the required attributes in PHP array.
* For each of the *event*, it queries the database to give all the *showings* from the Showing table for that *event*. The required attributes of these *showings* are concatenated to the same PHP array which stores the *events* information.
* Call the function phpResponseToXML() which saves the XML in EventsWithShowingsSummary.xml.

1. **generateXML\_Trending\_ Top5Events.php**

* This script generates an XML file that contains,
  + Name and TabletLink of the top 5 trending *events*
  + StartDate and VenueName of the most recent upcoming *showing* of that *event*
* The script queries the database to give all the *events*,
  + that have atleast one showing yet to be screened i.e. with trendingStatus = 1
  + ordering them in decreasing order of the TotalSold tickets.
* The required details are saved in the PHP array.
* Call the function phpResponseToXML( ) passing the PHP array as argument. The function generates the XML file Trending\_Top5Showings.xml.

1. **generateXML\_RushStatus\_Top5Showings.php**

* DUE TO LACK OF NEW DATA, THIS SCRIPT WOULD GENERATE AN EMPTY XML FILE RIGHT NOW.
* This script generates an XML file that contains,
  + StartDate and VenueName of the top 5 rush status *showings*
  + Name and TabletLink of the *event* under which the *showing* falls
* The script queries the database to give all the *showings*,
  + that are in rush status i.e. RushStatus = 1
  + that are yet to be screened i.e. trendingStatus = 1
  + ordering them in decreasing order of the ShowingStartDate tickets.
* The required details are saved in the PHP array.
* Call the function phpResponseToXML( ) passing the PHP array as argument. The function generates the XML file Trending\_Top5Showings.xml.

1. **display\_Trending\_Top5Events.php**

* This script takes in the ‘Trending\_To5Events.xml’ and prints the following data:
* Event Name, with a hyperlink to its TabletLink
* StartDate in format: MM/DD HH:MM (12-hour)
* Venue Name

1. **display\_RushStatus\_Top5Showings.php**

* This script takes in the ‘RushStatus\_Top5Showings.xml’ and prints the following data:
* Event Name, with a hyperlink to its TabletLink
* StartDate in format: MM/DD HH:MM (12-hour)
* Venue Name

1. **displayTablet.php**

* This script takes in the ‘RushStatus\_Top5Showings.xml’ & ‘Trending\_To5Events.xml’ and prints the following data:
* Event Name, with a hyperlink to its TabletLink
* StartDate in format: MM/DD HH:MM (12-hour)
* Venue Name
* Since this script is for displaying the trending and rush films on the tablet app, the films are displayed in a 165x369 box with appropriate styling (css/displayTabletStyle.css).

1. **generateLogFile.php**

* This script generates a text log file which contains 24 sets of following information, one for each hour of the day.
* Top 5 trending events
* Top 5 rush screenings
* At the end of each day, we would have a log file with name logFileYYYYMMDD.txt.

1. **sendEmail.php**

* This script sends email to required people (tentatively, Halfdan, Kathleen, Lou, Matt) every day during the festival with an attachment of the log file (generated by generateLogFile.php).
* The log file contains top 5 trending events and top 5 rush showings of that day for every hour.
* Subject of the e-mail: "Trending Films Log File for MM/DD.
* Message of the e-mail: "Please find the Trending films Log File for MM/DD. If you have any questions, please speak with Matt or Lou.

* **FOLDER – css:** This folder contains the styling .css file for the ‘displayTalblet.php’ script.
* **FOLDER – XML Files:** This folder contains the XML files generated by the scripts in the main folder ‘Trending\_11.15.2013’.

**EventsWithShowings.xml**

* This is the XML file generated by generateXML\_EventsWithShowings.php script.

**EventsWithShowingsSummary.xml**

* This is the XML file generated by generateXML\_EventsWithShowingsSummary.php script.

**Trending\_ Top5Events.xml**

* This is the XML file generated by generateXML\_Trending\_Top5Events.php script.

**RushStatus\_Top5Showings.xml**

* DUE TO LACK OF NEW DATA, THIS IS AN EMPTY XML FILE RIGHT NOW.
* This is the XML file generated by generateXML\_ RushStatus\_Top5Showings.php script.
* **FOLDER – Log Files:** This folder contains the log files generated by the script generateLogFile.php. Currently, there are two log files in this folder. However, there will be more log files in future to be emailed when needed.

**logFile20131009.txt**

* This is the log file generated for the day October 9, 2013.

**logFile20131011.txt**

* This is the log file generated for the day October 11, 2013.
* **(Suggested) Cron Job Schedule for various scripts:** Some scripts need to be run every hour of the day or which some need to run once every day. Based on the requirement, the following cron jobs should be scheduled.
* The database should be updated every hour at HH:01 AM/PM
* The log file of the current day should be updated every hour at HH:30 AM/PM. The new top 5 trending and rush list shall be appended to the log file of that day.
* At the end of the day, an e-mail with attachment logFileYYYYMMDD.txt will be sent.
* An example schedule for a particular day would look like follows:

1. 12:01 AM – run updateDB.php

12:45 AM – run generateLogFile.php

1. 1:01 AM – run updateDB.php

1:45 AM – run generateLogFile.php

1. .... .... .... .... ....

.... .... .... .... ....

1. .... .... .... .... ....

.... .... .... .... ....

.... .... .... .... ....

.... .... .... .... ....

1. …. .... .... .... ....

…. .... .... .... ....

1. 11:01 PM – run updateDB.php

11:45 PM – run generateLogFile.php

11:50 PM – run sendEmail.php