MOBILE APPLICATION FOR VIEWING PRICE AND LOAD OF POWER DATA FROM NEW YORK INDEPENDENT SYSTEM OPERATOR

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Nimish Gupta

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

Major Program:
Software Engineering

December 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

MOBILE APPLICATION FOR VIEWING PRICE AND LOAD OF POWER

DATA FROMNEW YORK INDEPENDENT SYSTEM OPERATOR

By

Nimish Gupta

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Wei Jin (Computer Science)
Chair

Dr. GurisimranWalia (Computer Science)

Dr. Sangita Sinha (Chemistry)

Approved:

04/14/2014
Date

Dr. Kenneth Magel
Department Chair

ABSTRACT

In New York, the New York Independent System Operator (NYISO) supervises the wholesale electrical market and collects different kinds of data, including price and load data, which are available on the NYISO's website. These data are not available on mobile and tablet devices, and the NYISO website does not provide the ability to render customized reports which would allow the user to filter and manipulate the data for effective searches, analyses, and visualization. To overcome these problems, a mobile application was developed. This application not only allowed the users to effectively browse through the website's contents, but also empowered the users to gather information and to analyze it in an effective manner. The application also provides a clean interface for the user to view different attributes of the price and load data for the state, allowing user to run queries on those data.

ACKNOWLEDGEMENTS

I would like to thank my adviser, Dr. Wei Jin, for her constant guidance and support. This paper would not have been possible without her. Besides my adviser, I would like to thank my supervisory committee members, Dr. Gurisimran Walia and Dr. Sangita Sinha, for their time and consideration. I would like to express my sincere gratitude to Dr. Kendall Nygard for helping me to choose the topic for the mobile application.

Also, my sincere thanks go to my family and friends for their support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The mobile application for viewing price and load of power datafrom New York Independent System Operator (MAPLNYISO) has been built to present the data provided by NYISO in a format that is easily accessible to the user. It enables the user to interact with the system at any time and at any place (via a mobile device) and to extract the desired load and pricing data without much hassle.When running this application, users can select the files to download/upload the data into MySQL tables.The system enables the user to run queries by selecting the database tables; it will calculate the average for the data stored in the files and will display the load and pricing data. This research paper explains the mobile application for viewing price and load of power datafrom New York Independent System Operator (MAPLNYISO). “The New York Independent System Operator (NYISO) is a not-for-profit corporation responsible for operating the state’s bulk electricity grid, administering New York’s competitive wholesale electricity markets, conducting comprehensive long-term planning for the state’s electric power system, and advancing the technological infrastructure of the electric system serving the Empire State ” [1].

MAPLNYISO provides users with a graphical interface that helps them to access and to use the price and load data features of the NYISO website. The NYISO publishes pricing and load data for all the energy markets, a number of data reports for the capacity market to help market participants (MPs), and several ancillary service data reports. Due to how all these markets function, a new pool of data is created each day and needs to become available to the market participants. These data are essential for the MPs to make better buying decisions [1]. MAPLNYISOdoes this indispensable task of providing the required data to the user on his mobile device and makes accessing the NYISO pricing and load data considerably easier for the

user. Whether it be a policy maker, a stake holder, or an investor in the power system, the user has the database in the palm of his hand and is able to navigate through it smoothly. The NYISO also conducts auctions that match the suppliers offering to sell power and the retail, electric service companies looking to buy it. These auctions further strengthen the requirement and need for this system.

The purpose of MAPLNYISO would be that users are able to select or to deselect load and pricing data files to download andrun different queries. The auctioning process can be greatly facilitated by the NYISO mobile application, and access to the NYISO data will be easier for every market participant or any other user. In short, MAPLNYISO will help NYISO effectively perform its role as the impartial broker of New York's wholesale power market. The paper explains the different aspects and functionalities that could be utilized by the people offered by this mobile application. The paper explains the importance of the application and how the application could be utilized. The research paper also explains different applications that are similar to MAPLNYISOand that are available on the market. Each application is explained with brief details which include various aspects of the functionalities that are offered to users. The applications differ from each other in several respects. The technologies used by the applications vary and enable users to take advantage of the different functionalities and services supported by the technologies available. The research paper details these technologies to make the readers aware of the latest trends for mobile applications, which will also serve as a basis for comparingMAPLNYISO with the other applications that are available on the market. MAPLNYISO provides the following functionalities.

Users are able to select the load or pricing data files to view and download to the server.This feature provides the users with an option for selecting the load or pricing data files to

view,simultaneously downloading the file on the MAPLNYISO. Users are able to select a database table or tables from which to query.This feature provides the users with an option to select load or pricing data;utilizing this feature, the users are able to access the MAPLNYISO database andto select (a)particular file/files by the date range. The database's table format also provides fast access to the data using the set of queries that a user wants to apply on the database. Each table has its particular significance and perspective for the data that it stores.

Users are informed about any errors in their selection.The features that are supported by MAPLNYISO include the selecting or deselecting of load or pricing data files to be downloaded from the NYISO website. When the user accessesMAPLNYISO to perform a particular functionality, there are chances that the user can create an error or that the user can raise a request that is not handled by MAPLNYISO. With such circumstances, the mobile application notifies the users about the error that the system is encountering when the user creates the request.

LITERATURE REVIEW AND RELATED WORK

The NYISO divided the market into 11 zones (Zone A-Zone K) as shown in Figure 1 [2].The NYISO market covers around 11,000 miles of high-voltage transmission and administers bulk power markets that trade an average of $7.5 billion in electricity and related products annually [1]. The NYISO conducts auctions that match the retail electric-service companies looking to purchase power and the suppliers offering to sell it [3]. NYISO collects different types of pricing, power grid, and load data from all the New York power plants.

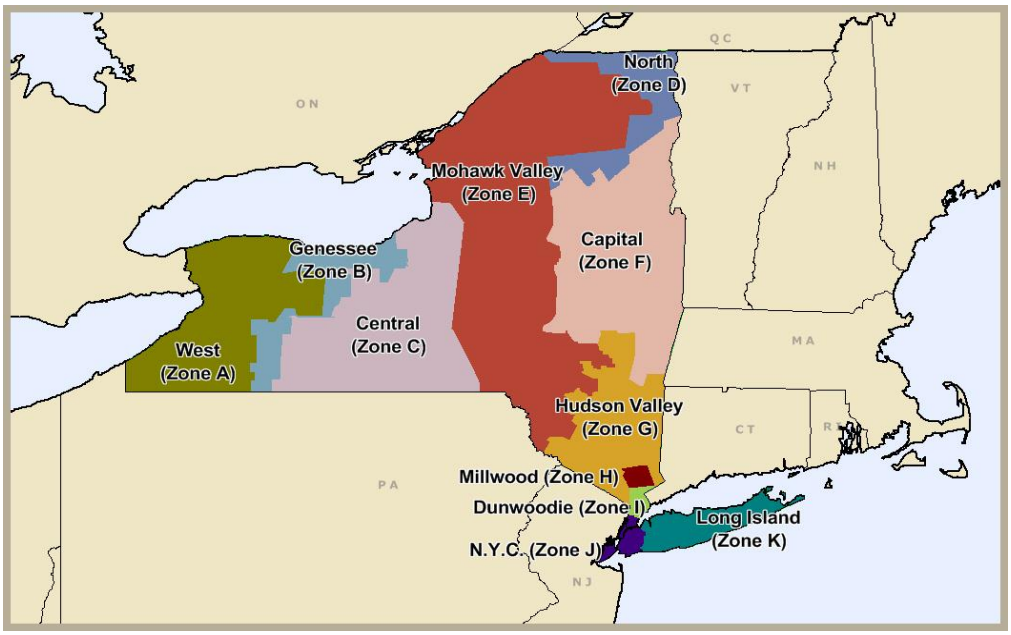


Figure 1. NYISO Zonal Map.

Pricing data include day-ahead market, real-time market, time-weighted market, hour-ahead market, and ancillary service data. The NYISO auctions the electricity price which depends on location, load, and time. (Peak-hour prices are different from non-peak hour prices.) Five-year averages for New York's annual prices are shown in Table 1 [2]. In Table 1, electricity prices directly depend on the cost and quantity of fuel which the power plant uses to fulfill the

consumers' need for electricity.After analyzing the data, NYISO determines that, with the current resources, there is enough generation, transmission, and demand response to serve the state until 2019 [3].

Table 1. 5-Year Average Price Data for Day-Ahead on On-Peak Prices.

Annual Average Day Ahead On Peak Prices ($/MWh)						
	2008	2009	2010	2011	2012	5-Year Avg
Mass Hub	$91.55	$46.24	$56.18	$52.64	$42.06	$57.75
Ny Zone G	$100.99	$49.80	$59.48	$56.41	$44.35	$62.22
NY Zone J	$112.63	$55.77	$65.76	$62.71	$46.95	$68.79
NY Zone A	$68.34	$35.54	$43.89	$41.52	$35.82	$45.03
PJM West	$83.70	$44.60	$53.68	$51.99	$40.86	$54.98

Related Work

"Stock Guru" is a mobile application (for the iPad) that has been developed to analyze stock-market data. It provides a thorough analysis of every stock, helping the user make a decision about buying or selling. For each stock, the user is provided with a guru rating, valuation rating, financial strength, momentum rating, stock screening, comprehensive financial data, risk score, risk percentile, risk return chart, Sharpe ratio, Sharpe percentile, risk-volatility comparisons, and key stock data. "Stock Guru" provides stock-analysis reports that are independent and unbiased because they are based on quantitative and statistical models. Hence, it helps the investor choose stocks using proven quantitative strategies [4]. It utilizes the Objective-Capplication programming interface(API) and the JavaScriptObject Notation(JSON) platform; it is also believed that this application may use the Yahoo! Finance API for iOS. This API allows a stock-symbol search,supports a free-text search for stocks, and helps find symbols for global stock exchanges.The Yahoo! Finance API provides a ton of information, price, volume, etc., for

each stock. “Stock Guru” usesYahoo! Query Language (YQL)which is provided by Yahoo!;YQL is a service that provides data from many sources, such as stock quotes from Yahoo! Finance [5]. Another application is the ArcGIS mobile application for ageographic information system (GIS): A GIS is a system which was developed to gather, store, analyze, manipulate, and manage as well as to present various geographical data. The ArcGIS application’s users may include GIS professionals, analysts, etc. The user can navigate maps, collect and report data, and perform GIS analysis using this handy mobile application [6]. The ArcGIS mobile application uses the Objective-C API that allowspeople to add mapping and GIS functionality to iPhone, iPod Touch, and iPad applications. The API leverages the functionality provided by ArcGIS’online and on-premise ArcGIS servers throughREpresentational State Transfer(REST) web services. The API includes a map component, a variety of layers, and tasks. The map component displays the map content from layers or webmaps which, in turn, rely on backing map services from the ArcGIS server or other geospatial technology, such as Bing, OpenStreetmap, OGC WMS, OGC WMTS, etc. The ArcGIS library depends upon and already includes the following third-party libraries:JSON framework and core-plot [7].

Both of these applications follow the set of guidelines delivered by Apple, such as taking advantage of the entire screen. Instead of using insets and visual frames, these applicationslet the content extend to the screen’s edges. Theseapplicationsconsider the visual indicators of physicality and realism. Colors, gradients, and drop shadows sometimes lead to heavier user interface (UI) elements that can overpower or compete with the content, hence we focused on the content and let the UI play a supporting role. Both applications use plenty of negative space which makes important content and functionality more noticeable and easier to understand.

Negative space can also impart a sense of calm and tranquility, and it can make applications look more focused and efficient [8].

PROPOSED SOLUTION

While developing the prototype for MAPLNYISO, I followed the Unified Software Development Process, an iterative and incremental development process; this process was described as follows: “Business value would be delivered incrementally in time-boxed cross-discipline iterations” [9]. During the application’s development phase, each iteration resulted in an increment to what was previously done. Each iteration led to improved functionality compared to the previous version [9].

I also used the defensive programming approach which is, in essence,where one has to expect the worse. Defensive programming is stated as an approach that minimizes bugs; software bugs can potentially be used by a cracker for a code injection, a denial-of-service attack, or another attack [10].

Technologies Used to Develop the Prototype

I used the Model View Controller (MVC) architecture pattern while developing the MAPLNYISO prototype. Classes and objects provided by the Android and iOS libraries were used to create buttons, views, and layouts as well as to perform actions on application. In Android, Java classes are compiled into Dalvik executables and run on Dalvik, a specialized virtual machine (VM) designed specifically for Android; Dalvik VM is a register-based architecture [11, 12]. Dalvik has some specific characteristics that differentiate it from other standard VMs [11].The VM was designed to use less space.The constant pool was modified to use only 32-bit indexes to simplify the interpreter [11].Dalvik uses its own 16-bit instruction set that works directly on local variables [11]. Thelocal variable is commonly picked by a 4-bit “virtual register” field [11]. The Android applicationsare built as a combination of distinct components that can be invoked individually. For instance, an individual activity provides a

single screen for a user interface, and a service independently performs work in the background [12]. There are four different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed. Here are the four types of application components [13].

1. Activities: “An activity represents a single screen with a user interface, and an activity is implemented as a subclass of Activity” [13].
2. Services: “A service is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. A service is implemented as a subclass of Service” [13].
3. Content providers: “A content provider manages a shared set of app data. Content providers are also useful for reading and writing data that is private to your app and not shared. A content provider is implemented as a subclass of ContentProvider and must implement a standard set of APIs that enable other apps to perform transactions” [13].
4. Broadcast receivers: “A broadcast receiver is a component that responds to system-wide broadcast announcements. A broadcast receiver is implemented as a subclass of BroadcastReceiver and each broadcast is delivered as an Intent object” [13].

The iOS applications are built using the target-action patterns. We specified the target as the view controller because it controls the view where these controls reside.While the application isactive and in the foreground, the system sends touch events for processing [14]. The user interface kit(UIKit) infrastructure does most of the hard work for delivering events to the custom objects [14]. For controls, the UIKit simplifies things even further by handling the touch events and calling the custom code only when something interesting happens, such as when the value of

a text field changes. View controllers also play a very important role in the application's user interface. A view controller is an instance of the UIViewController class and is responsible for managing a single set of views as well as the interactions between those views and other parts of the application [8]. Because iOS applications have a limited amount of space in which to display content, view controllers also provide the infrastructure needed to swap from one view controller to another one [8].

We used Hypertext Preprocessor(PHP), JavaScript Object Notation (JSON), Objective C for iOS, and Java for Android. PHP was used because it is a programming language that can do all sorts of things: evaluate form data sent from a browser, build custom web content to serve the browser, talk to a database, and even send and receive cookies [15]. We used JSON because it is a language-independent data format andis a lightweight, data-interchange format [15]. We chose Objective C and Java because they should be used while developing mobile applications for iOS and Android, respectively.

We could have used ASP.NET/Java for web programming and given the output in the JSON format. Because PHP supports JSON and CSV reading with its built-in functions, we preferred PHP to handle web services.

Functionality

View Functionality: We are using the classes and objects provided by the Android library and the iOS library to create buttons, views, and layouts as well as to perform actions. Based on requirement to obtain data from the NYISO website, we have designed the layout in such a way that it follows the hierarchical structure given on the NYISO website.For example, under Pricing Data, we have sub-categories such as Locational Based Marginal Pricing (LBMP), Day-Ahead Market LBMP, Real-Time Market LBMP, etc. The user has to select the groups

within these sub-categories, such as Zonal, Generator, and Reference Bus.Therefore, we have used different view controllers, sub views, buttons, list views, and a built-in browser to handle the requirement and to show the necessary data in the desired format which is compatible with a mobile phone or tablet.

File Download Functionality: When we click on view, a new browser opens in the Android or iOS application. This new browser starts with the URL from which we are loading data. The URL has a parameter called “external-url” which is being sent via the getting method(GET method). This external URL is nothing but the path of the CSV file on the NYISO server that we are supposed to read. The server checks to see if the external-url is received, and as soon as we receive the external-url, the filename from the end of the URL is fetched. Next, we check to see if the file exists in the directory; if it does exist, then we copy that file from the NYISO server; otherwise, the code will throw an error. The following code explains how files are stored ona local server using the server’s PHP programming. The comments next to each line of code determine the action it is performing.

```
if(isset($_REQUEST['external-url']))// checking whether the URL is received
{
        $url=$_REQUEST['external-url']; // The external URL value is being copied to a variable
        $filename=basename($url);    // only the file name is being extracted
        // if the external URL is http://mis.nyiso.com/public/csv/damlbmp/20131221damlbmp_zone.csv,
        // we will fetch 20131221damlbmp_zone.csv
        $filecheck=file_exist_check($filename);
        // this function checks whether a file with this name already exists in the directory and returns true or false
        if($filecheck){}
        else if (filter_var($url, FILTER_VALIDATE_URL) === FALSE)
        {
        //validating whether the URL exists or not
                die('Not a valid File');
        }
        else if(!$filecheck)
        {
        //copy the file from NYISO server
                copy_file_from_source($url);
        }
```

}

Query Functionality: After the file is copied from the NYISO server, the contents of the CSV file are read, and the file is saved in a MYSQL database. We read a CSV file and access its data using the fgetcsv() function that will read each line of a CSV file and assign each value to an array. PHP is combined with MySQL as a cross platform; interacting with MySQL makes PHP a far more powerful tool. The NYISO server is using X (cross), Apache, MySQL, PHP, and Perl (XAMPP); the NYISO server is a simple, lightweight Apache distribution that makes it extremely easy to create a local web server for testing purposes. Everything needed to set up a web server–the server application (Apache), database (MySQL), and scripting language (PHP)–is included in a simple, extractable file. We are using PHP function calls to connect with MySQL. PHP functions are a flexible and easy way to consolidate code [16]. The PHP language provides functions that make communicating with MySQL extremely simple [16]. We use PHP functions to send SQL queries to the database and leverage the PHP functions in such a way that PHP handles the details.

Save Functionality: The save_to_db.php file does the operation to save the content in the database. The following code shows how the action takes place: the $extensionsarray variable determines the structure of the table based on the data type. Based on the file's name, the type of request is determined. For example, the extension is removed from the file name; the first eight characters of the filename determine the date, and the remaining substring determines the data type.

Pricing data -> Day-ahead market LBMP -> Zonal -> Determined by damlbmp_zone in the file name.

Pricing data -> Ancillary Services -> Day Ahead -> Determined by damasp in the file name.

Load Data -> Load Forecast -> NY Load Forecast -> Determined by isolf in the file name.

By determining the value of $filearraykey ->, the corresponding table will be created if it was not created earlier. After confirming that the table is ready, we proceed by reading the CSV file and saving the data into the respective table.

The readCSV() function is used to read the CSV file and to get all the contents into an array. The function is copy_content_to_table($columnslist,$contentarray,$tablename,$firstrowheadings=TRUE).The function plays a key role in copying the content from a CSV file to the database.The data in the CSV columns are saved in the MySQL table using the INSERT Query (insert the data in an array) of MySQL through the PHP code.

Calculating Result Functionality: The Query module asks the user to give the desired input, and the given input is passed to a PHP webpage: /nyiso/query-init.php. The parameters are sent to this page; based on the parameters sent, the query will be performed on the database, and the response will be given.

Use Case

Figure 2 shows the relationships between the various identified actors and their respective use cases. The figure shows all the required functionalities that we plan to implement in our system.This application provides three options for a user who starts utilizing the interface. A user can select an option from the pricing data, load data, and query. The user can see the data, by date, for the day-ahead market, real-time market, hour-ahead market, load forecast, actual load, etc. as well asseeing queries such as average, sum, and minimum marginal cost. This application allows the user to select the date range and data type for the query.

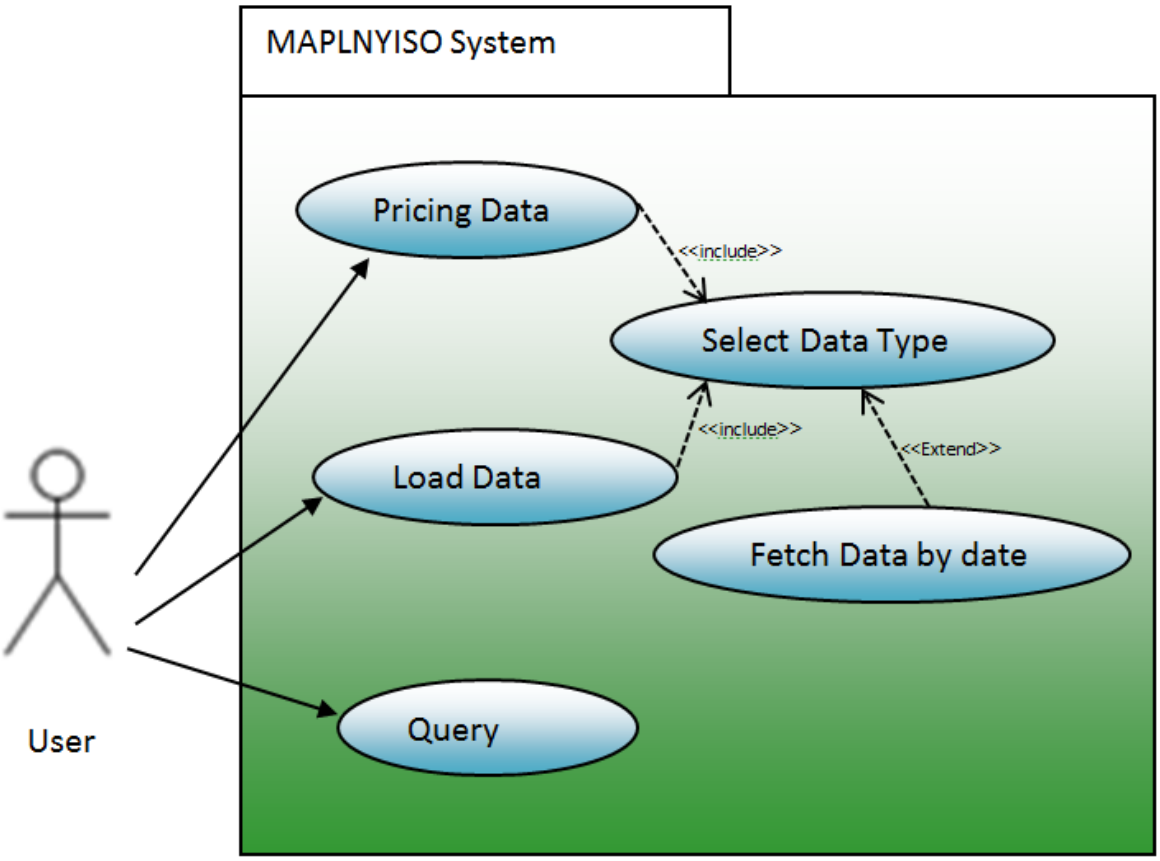


Figure 2. Use Case Diagram.

EXPERIMENT AND TESTING

Functionality of the Application

When the application starts, the web service for the initialization of variables is called. Based on the structure of the array given in these variables, we are categorizing the data into two parts.

1. Pricing Data
2. Load Data

On the home screen, there are three options:

1. Pricing Data
2. Load Data
3. Query

The layout is shown in Figure 3.Once the user clicks either of these two options, i.e., pricing data or load data, a new view opens where the user has to select a sub-category (e.g., Ancillary Services or Day-Ahead Market LBMP).The layout is given in Figure 4.When the user touches the checkbox next to the category, all the internal sub-categories will be checked. When the user touches the fetch data buttons, a webservice is called.The list of all items the user selected is sent through the webservice call. The response data hold the list of webpage URLs where the content can be read. The webpage links are categorized by the user's choice of categories before sending the web service call.

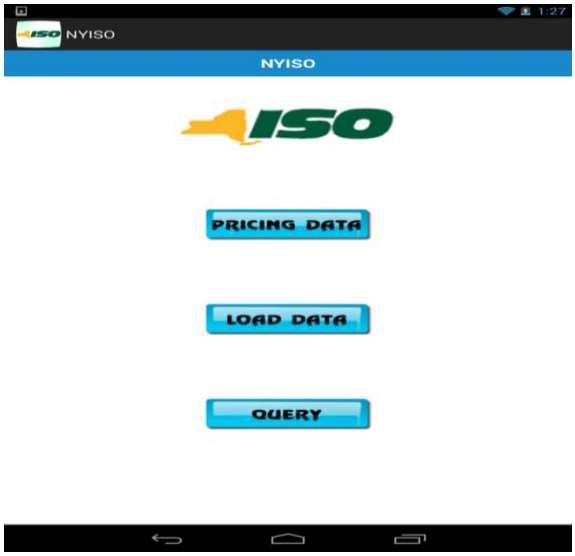


Figure 3. Home Screen.

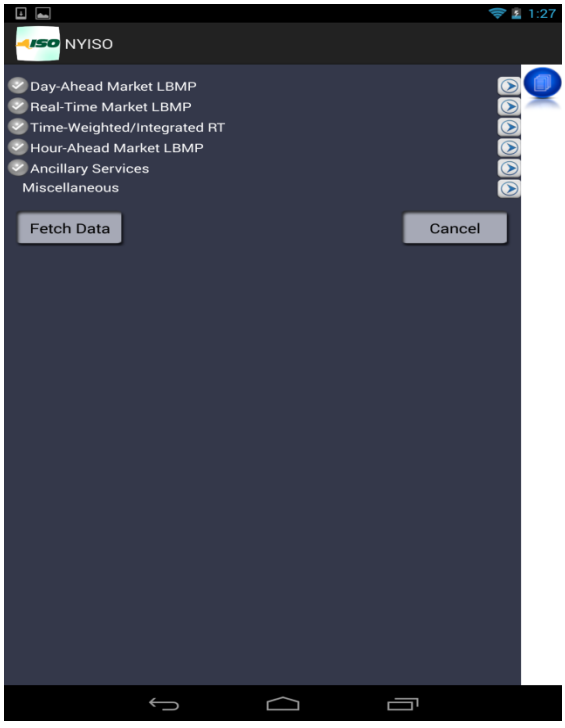


Figure 4. Option Under Price Data.

When the user touches the checkbox next to the category, all the internal sub-categories will be checked. When the user touches the fetch data buttons, a webservice is called.The list of

all items the user selected is sent through the webservice call.The response data hold the list of webpage URLs where the content can be read. The webpage links are categorized by the user's choice of categories before sending the webservice call.

A view is displayed where all the sub-categories selected by the user are as shown in Figure 5. When the user clicks on any category, data related to the corresponding category are shown and ordered by date. The date range is from the current date to 10 days earlier. Corresponding to every date, buttons related to the data are displayed. By touching these buttons, a webpage will be shown in a webview.

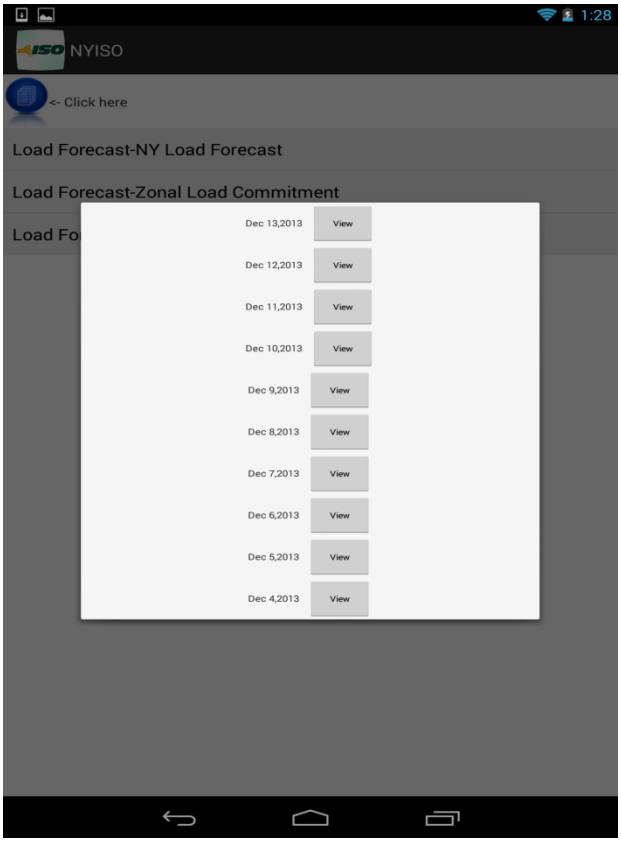


Figure 5. View Files.

The functioning of the webview is as follows:

1. It will copy the related file from the NYISO server to a local server.

2. The CSV file will be read, and the corresponding data will be saved in the MySQL database using PHP programming.

On the home screen, there is one more option: query. When the user clicks on query, he can select from the previously mentioned primary and secondary categories. After he selects the desired function, a webservice call is sent using the required parameters, and the webpage gives the query result as shown in Figure 5.

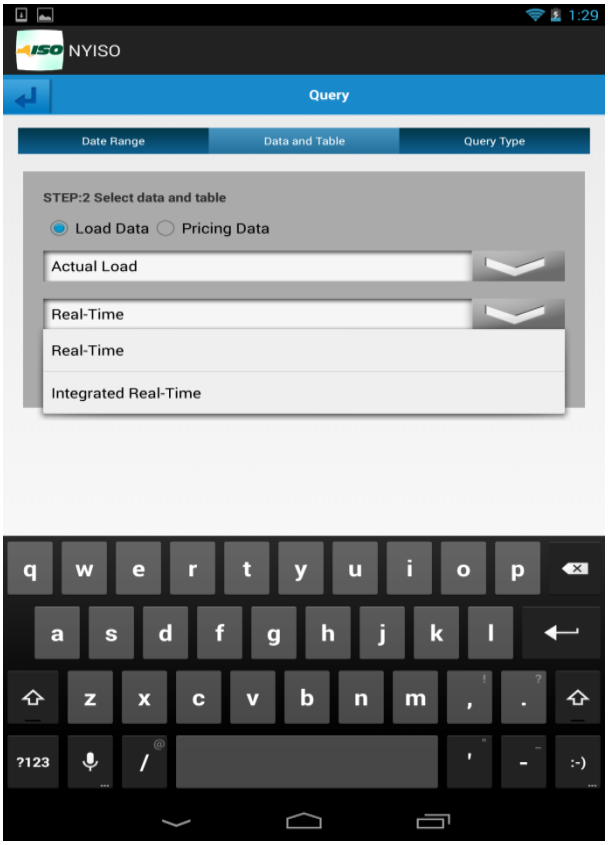


Figure 6. Query.

Suppose the user needs to calculate the average and minimum marginal-cost losses that result with Day-Ahead Market LBMP- Zonal for December 20, 2013, and December 21, 2013. The user has to follow certain steps to obtain the desired result. This query result is shown in an alert view as shown in Figure 7.

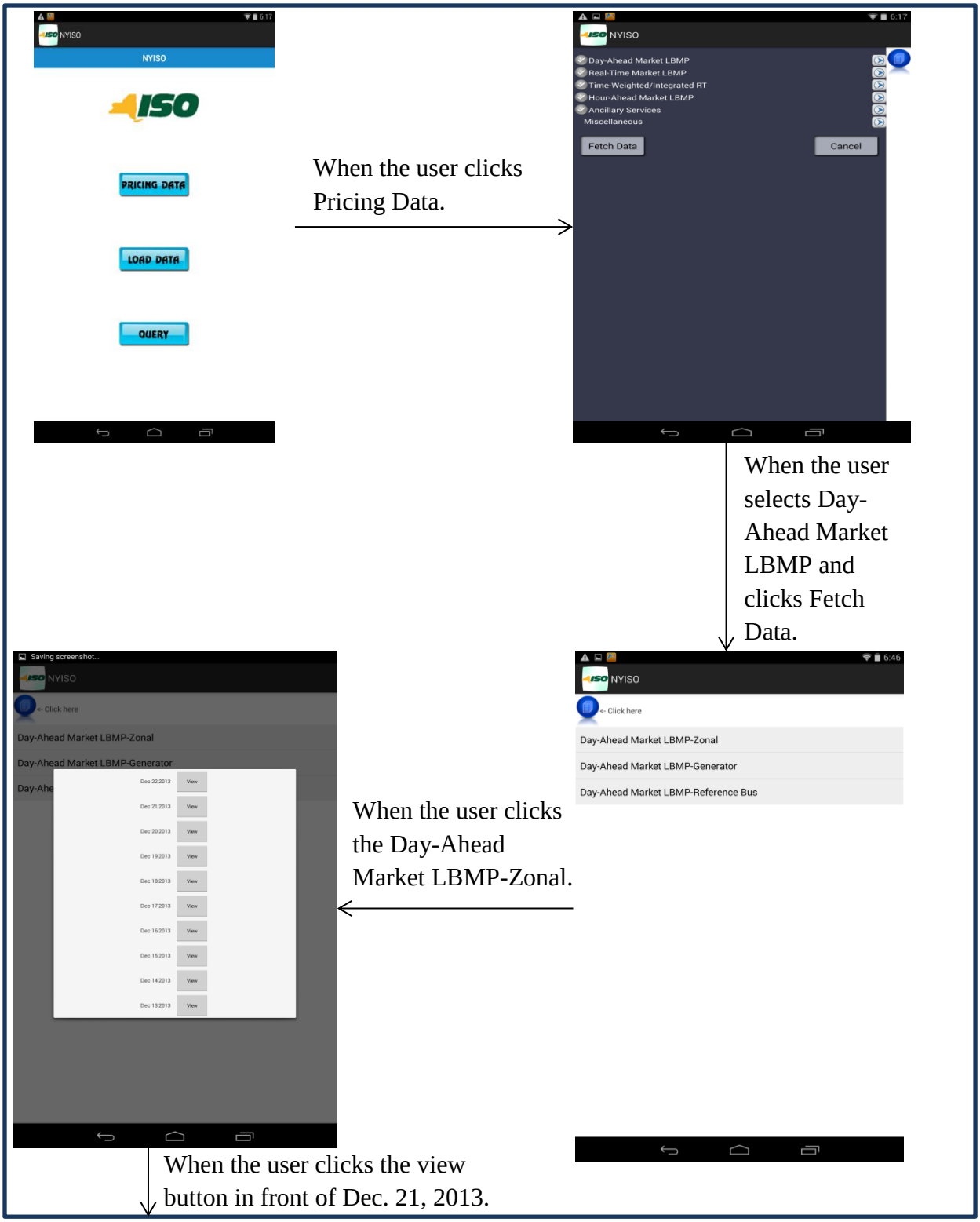


Figure 7. Results.

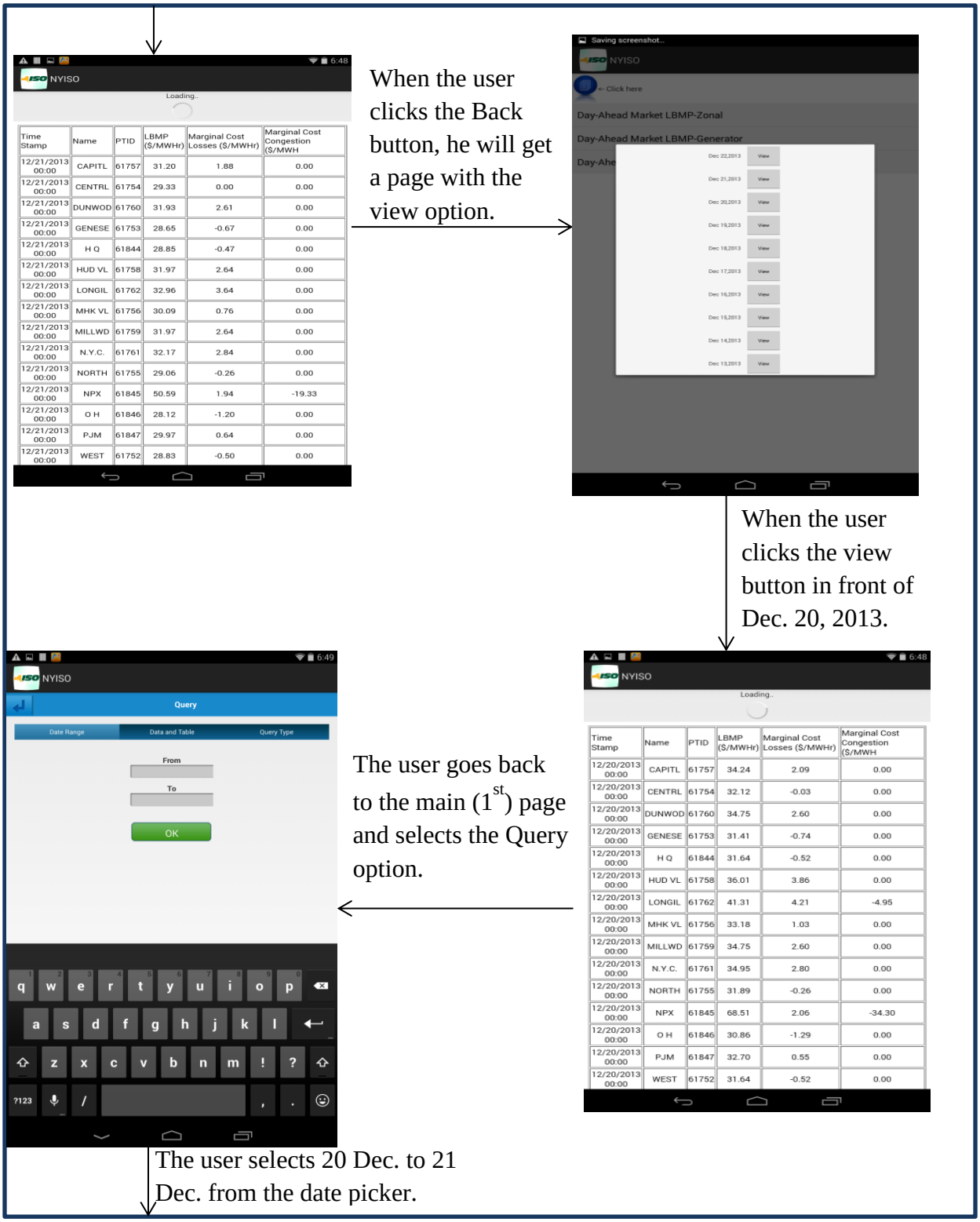


Figure 7. Results. (continued)

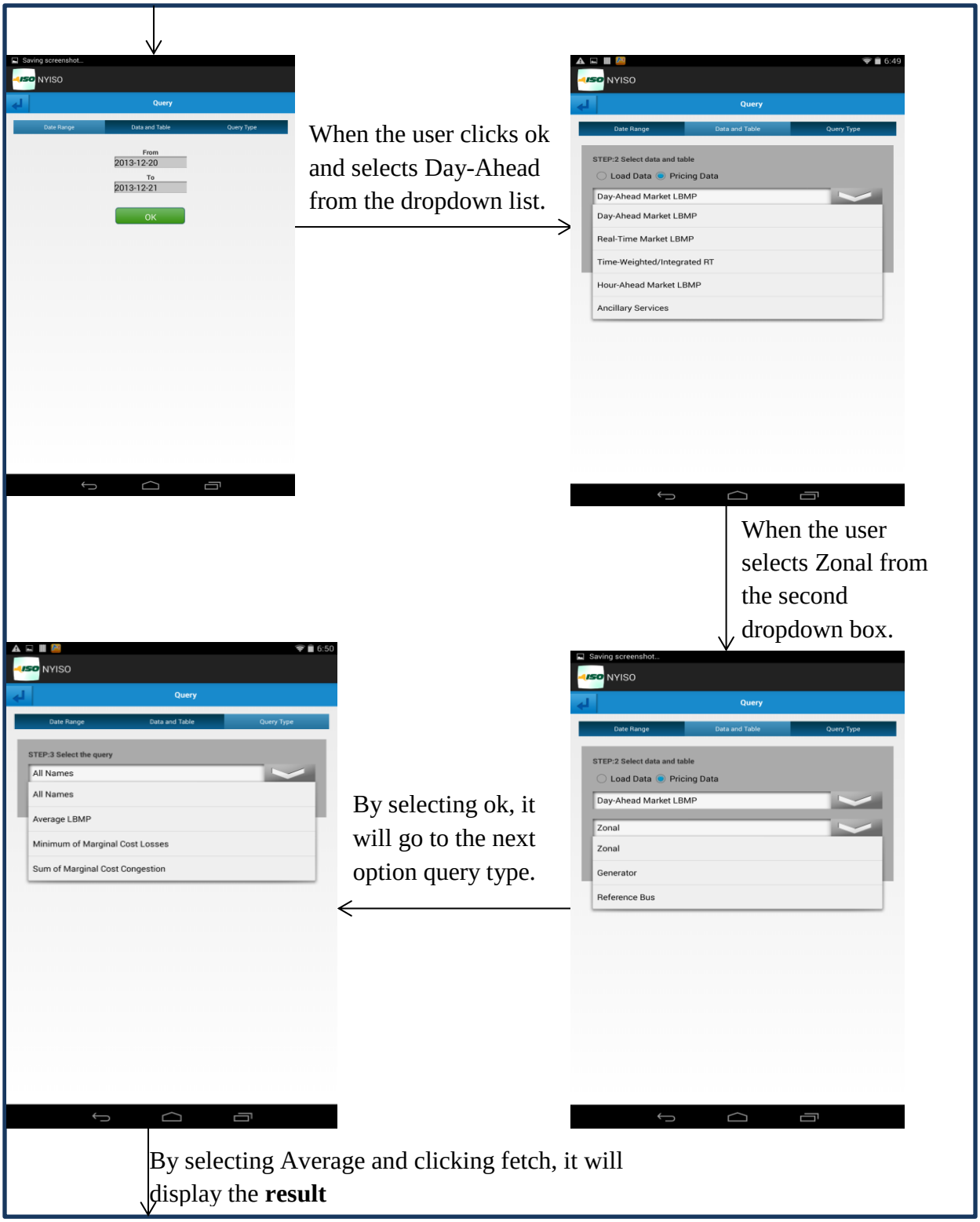
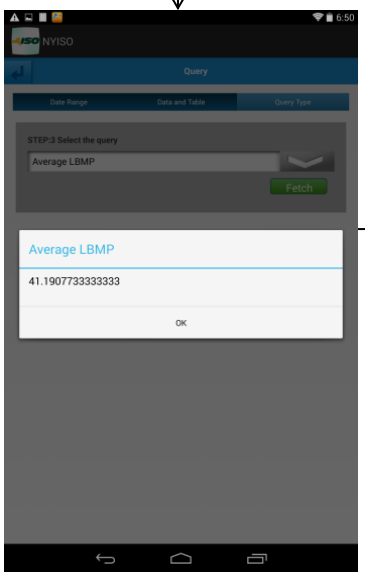
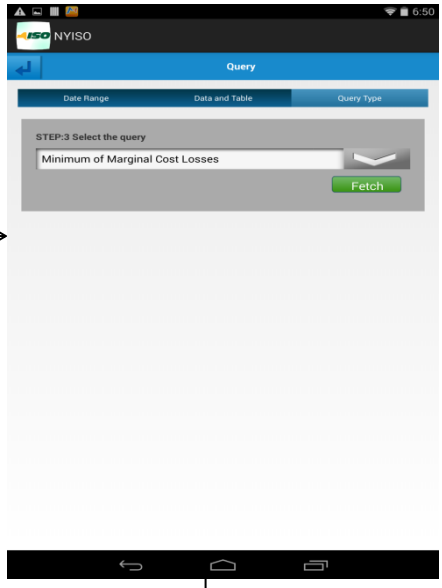


Figure 7. Results. (continued)

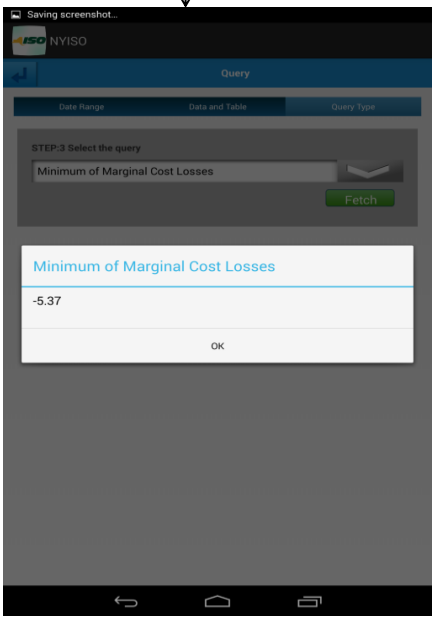


This screenshot shows the average result for the Dec. 20 and Dec. 21 data files from the Day-Ahead Market LBMP of Zonal category.

Click Ok; select the Minimum of Marginal Cost losses from the dropdown box.



Click on fetch; it displays the **result.**



This screenshot shows the minimum for the marginal cost losses that result with the Dec. 20 and Dec. 21 data files from the Day-Ahead-Zonal category.

Figure 7. Results. (continued)

Testing

This step aimedto test the developed MAPLNYISO prototype with all the possible inputs to determine if the software works properly when subjected to different inputs. The main goal of testing for this paper was to ensure that the developed system was bug-free and dynamic enough to handle all kinds of user input. The approaches were Test Driven Development (TDD) for the system components and unit testing for one of the results using asimple test for PHP.

Because we are using PHP and because of its compatibility with JavaScript, we have used a third-party widget, JavaScript, for error handling. JavaScript ensures that proper values are sent between the scripts and the database. This component consists of two main scripts:

1. save_to_db.php is used to save the content to the database.
2. save_to_db_Process.phpis the backend logic that takes the parameters from save_to_db.php and checks the database to see if data for that date exist or not.

Table 2. Save Test.

	Test Case ID	Test Process	Purpose
1	Save_to_db_TID_1	Check to see if the correct parameters (date) were being passed.	The test was done to see if the correct parameters were being passed between the pages. The test passed when the correct parameters were sent.
2	Save_to_db _TID_2	Check to see if the script was connected to the right component.	The test failed when the script was not in the right folder.
3	Save_to_db _TID_3	Check to see if the test script existed.	The test passed if the path to the test script was not found.

Now, we will show the tests for the system components. System-component testing consisted of scripts that are connected to other scripts. We did more black-box testing on the system. We gave components with inputs and evaluated the output to deduce whether the test passed or failed.

Table 3. Test.

Test Case Number	Test Case ID	Purpose
1	View_TID_1	To see if all database files are displayed with the application view page.
2	View_TID_2	To see if all files for a particular category are displayed on the view.php page once a user clicks a particular category.
3	Query_TID_1	To see if the correct dates are given by the user when he selects a date option.
4	Query_TID_2	To see if the data and table list are displayed when the user searches for a query.
5	Query_TID_3	To see if null is displayed when the user enters a query for which a database is not present.
6	Query_Result_TID_1	To see if the correct parameters are given and the desired result is displayed.

Response Time

We calculated the response time based on the time it takes to switch from one page to another one. The response time was captured by sending the request and then checking the server's response time. We used the connection DidFinishLoading method which gives the complete response time while going from one request to another. On average, it took around 3-5 seconds to the load the application when the internet connection was ideal. When going from one page to another, it took less than 1 second of time, however, when we clicked fetch data, the total time was around 3-15 seconds, depending on how many options were selected for the price and load data. The response time for the fetch data varied a lot because we obtained data from the

NYISO website, so the response time depended on the load on that website at that time. We also copied the same data to a local server, and the time it took to copy the data varied from 2-20 seconds.

Table 4. Response Time.

Number	Functionality	Time in Seconds
1	Load Application	3-5
2	Load Option/Switch to Different Option	.5-2
3	Fetch Data	3-15
4	View/Download	2-10

What if We Use Different Technology

The logic behind the application is to copy files from the NYISO server, read them, and save the details to a MYSQL server. Copying files from the NYISO server can be achieved through web technology that supports CSV file reading, but to make the application portable and to read the data, we use the PHP and JSON web services.

Challenges Faced

Many challenges were faced while developing the application. The first challenge was to select the best technology and method available; we spent around a semester understanding the various technologies available. The second challenge was frequent upgrades for the Android and iOS operating systems. We started developing the application when we had an iPhone 4S, but because technology is evolving very fast, we had to switch to the iPhone 5S. Therefore, we devoted extra time to make the application compatible with the iPhone 5S because the default layout features of iOS 6, such as the colors for buttons, text, and the background, were not compatible with iOS 7. As a result, we had to define those features programmatically and to run features on iOS 7. Also, the recent upgrade of the Android 4.4 KitKat version was a challenge

because the application kept crashing with the date picker. We had to change the Target SDK in the manifesto file to resolve the issue. In PHP coding, when we tried to copy multiple files at once, we used to get a Request Timed Out error. To eliminate the issue, we decided to only copy one file from the readfile.php webpage.

CONCLUSION AND FUTURE WORK

Conclusion

When a huge number of data files are involved and calculation is to be performed by running queries on those data, the task of developing the mobile application becomes very challenging. We needed to present all the data shown on the website, so we had to create a design that allowed us to show all the available data on the small screen in a clear and concise manner so that the user does not feel overwhelmed. We were able to achieve all these objectives and goals through providing a mobile-application prototype MAPLNYISO to the market participants, including all entities that produce, transmit, sell, and/or purchase capacity, energy, and ancillary services in the NY wholesale market, for resale [17]. This application will help the MPs make better decisions while selling and buying electricity on the wholesale market. MAPLNYISO will help NYISO's mission to serve the public interest and provide benefits to consumers [17].

Future Work

This application prototype can be implemented on the NYISO server so that it becomes live and accessible for all stakeholders. The current version has supported queries such as computing the mean, median, and average for all tables referenced. Future directions include the support of more query types and integration of more functionality related to market data, such as power-grid data. We are also looking at including reports, maps, and graphs through the sencha framework and possibly including XAML data visualization module in the prototype in the future work.

REFERENCES

1. “NYISO: New York Independent System Operator." NYISO. [Online]. Available: http://www.nyiso.com/public/about_nyiso/nyisoataglance/index.jsp>. [Accessed December 2013].
2. Federal Energy Regulatory Commission, Market Oversight. [Online]. Available: http://www.ferc.gov/market-oversight/mkt-electric/new-york/elec-ny-yr-pr.pdf.[Accessed December 2013].
3. Power Trends 2013 Alternating Currents, New York Independent System Operator, May 2013. [Online]. Available:http://nyssmartgrid.com/wp-content/uploads/2013/05/Power_Trends_2013_May_2013_FINAL.pdf . [Accessed May 2013].
4. Value Prime Research and Analytics. [Online]. Available: http://valueprime.net/iphone-apps.php. [Accessed November 2013].
5. Yahoo Finance API. [Online]. Available: http://blog.sallarp.com/yahoo-finance-api-for-ios/. [Accessed August 2013].
6. ArcGIS App, Apps for Smartphones and Tablets. [Online]. Available: http://www.esri.com/software/arcgis/smartphones/arcgis-app. [Accessed August 2013].
7. ArcGI. [Online]. Available: https://developers.arcgis.com/en/ios/api-reference/index.htm. [Accessed May 2013].
8. Apple Guidelines. [Online]. Available: https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/index.html#//apple_ref/doc/uid/TP40006556-CH66-SW1. [Accessed June 2013].
9. Implementation Practice. [Online]. Available: http://www.aras.com/blogs/ACE-2010-international/09-ACE-2010-Aras Implementation-Best-Practices.pdf. [Accessed June 2013].
10. Defensive Programming. [Online]. Available: http://en.wikipedia.org/wiki/Defensive_programming. [Accessed March 2013].
11. Java and Android API. [Online]. Available: http://en.wikipedia.org/wiki/Comparison_of_Java_and_Android_API. [Accessed March 2013].
12. Android Guide. [Online]. Available: http://developer.android.com/guide/index.html. [Accessed June 2013].
13. Android Fundamental and Guide. [Online]. Available: http://developer.android.com/guide/components/fundamentals.html. [Accessed July 2013].
14. Patterns for iPhone. [Online]. Available:

http://www.dummies.com/how-to/content/knowing-the-targetaction-patterns-for-iphone-app-0.navId-407021.html. [Accessed February 2013].
15. JSON. [Online]. Available: http://www.json.org/. [Accessed November 2013].
16. CSV in PHP. [Online]. Available: http://www.codedevelopr.com/articles/reading-csv-files-into-php-array. [Accessed July 2013].
17. NYISO Manual and Guide. [Online]. Available: http://www.nyiso.com/public/webdocs/markets_operations/documents/Manuals_and_Guides/Guides/User_Guides/mpug.pdf. [Accessed December 2013].