Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute
InformingScience.org

JITEiip.org

Volume 16, 2017

BROWSER APP APPROACH: CAN IT BE AN ANSWER TO THE CHALLENGES IN CROSS-PLATFORM APP DEVELOPMENT?

Minh Huynh*	Southeastern Louisiana University, Hammond, LA USA	Minh.huynh@selu.edu
Prashant Ghimire	Southeastern Louisiana University, Hammond, LA USA	Prashant.ghimire@selu.edu

*Corresponding Author

ABSTRACT

Aim/Purpose	As smartphones proliferate, many different platforms begin to emerge. The challenge to developers as well as IS educators and students is how to learn the skills to design and develop apps to run on cross-platforms.
Background	For developers, the purpose of this paper is to describe an alternative to the complex native app development. For IS educators and students, the paper provides a feasible way to learn and develop fully functional mobile apps without technical burdens.
Methodology	The methods used in the development of browser-based apps is prototyping. Our proposed approach is browser-based, supports cross-platforms, uses open-source standards, and takes advantage of "write-once-and-run-anywhere" (WORA) concept.
Contribution	The paper illustrates the application of the browser-based approach to create a series of browser apps without high learning curve.
Findings	The results show the potentials for using browser app approach to teach as well as to create new apps.
Recommendations for Practitioners	Our proposed browser app development approach and example would be useful to mobile app developers/IS educators and non-technical students because the source code as well as documentations in this project are available for downloading.
Future Research	For further work, we discuss the use of hybrid development framework to enhance browser apps.
Keywords	browser-based apps, mobile app usage, app development, cross-platform web app, store-retrieve-display app, WORA, hybrid development framework.

Accepted by Editor Lalitha Jonnavithula | Received: March 24, 2016 | Revised: September 21, October 31, 2016; January 4, February 13, 2017 | Accepted: February 15, 2017.
Cite as: Huynh, M., & Ghimire. P. (2017). Browser app approach: Can it be an answer to the challenges in cross-platform app development? *Journal of Information Technology Education: Innovations in Practice, 16,* 47-68. Retrieved from http://www.informingscience.org/Publications/3667

INTRODUCTION

According to a recent study by Pew Research Center on Smartphone, nearly two-thirds of Americans are now smartphone owners. Among the smartphone owners, Millennials (ages 18-34) are by far the most technologically advanced user group, with a 95% cell phone ownership. Majority of the cell phones today are smartphones (Smith, 2015). Interestingly, this explosion of smartphones has caused some incredibly complex challenges for developers as well as Information Systems (IS) educators and students.

One of such challenges is how to create apps that can run on so many different platforms and devices on the market. Take the app CamScanner Pro for example. To be successful, its developers, INTSIG Information Co., must develop the same app to run on the following: iOS for iPhone; Android for devices such as HTC Desire, Samsung Galaxy, Motorola Droid Razr; Blackberry OS for Blackberry line of smartphones; Windows OS for Microsoft and Nokia Windows-based phones. Besides these major operating systems (OS), there are also other OS such as BADA from Samsung, Palm OS, Open WebOS, Maemo, MeeGo, Ubuntu, etc. Not only the developers of CamScanner face different OS but they also have to deal with different hardware from competing manufacturers.

What developers are facing are also manifested in what IS educators and students are struggling with, especially in acquiring and developing skills related to mobile web design and app development. The challenge is how to learn, understand, and apply something like mobile app development that has a steep learning curve, has no standard technology, and keeps on changing. This is simply because each mobile platform works differently due to its unique hardware and OS. As reported in previous works (Huynh & Ghimire, 2015; Joorabchi, Mesbah, & Kruchten, 2013; Native, web or hybrid mobile-app development, 2012), such a technical challenge makes teaching mobile apps programming languages and tools for native apps almost out of reach for non-technical students. For instance, the Java code written for an Android app has to be converted into the Objective-C code in order to run on an Apple iOS device. For those IS educators, it has become difficult to teach mobile web design and app development courses because the learning curve is so steep and the technology is not standardized and keeps on changing. For IS students, the technical requirements are so broad because there are so many different platforms and hardware to deal with. Consequently, they are scared away from learning and understanding about mobile web design and app development.

In this paper, we set out to explore ways to overcome this challenge. Our work is based on the previously published JITE:IIP article titled "Learning by Doing: How to Develop a Cross-Platform Web App" by Huynh and Ghimire (2015). Based on their prototype, we advance the project further by adding more features/functionalities to make the original web app better and more useful and conceptualizing the developing processes into an approach that could provide guidance for IS educators to teach and IS students to learn about app development. In this paper, we extended the previous work in three ways. The first is the additional features that we added to enhance the prototype and turned it into a fully functional app. The second is the redesign of the code so that it could be reused to create other apps. Finally, most important of all, is to leverage on the experiences learned and knowledge gained from the previous project to formulate a conceptual foundation for the browser app approach presented in this paper. Because of this extended work, we are able to propose a possible approach to the nagging problem of dealing with many different mobile platforms and devices.

Our proposed approach is based on three key premises: (1) viewing web browser as a platform, (2) relying on open-source technology standard, and (3) applying "Write once and run anywhere" concept to minimize the learning curve. Throughout the paper, we attempt to demonstrate that app development does not have to be time-consuming, highly sophisticated, and platform specific. Rather, we show how we went through the development of a series of mobile browser-based apps in a timely and efficient manner without a steep learning curve. One of our advocative premises is that there is an alternative approach to the native app development approach. We think that browser-based app may provide the answer to the cross-platform app challenge. Therefore, we organize our paper as

follows. In the first section, we highlight the previous development of the prototype called Student Services Web App and the on-going progresses. Next, we describe the major enhancements that we added to this Web App. These experiences lead us to the proposed approach for the design and development of mobile browser-based apps or browser apps in short. In the following section, we describe the proposed approach in action. This section illustrates the process of creating a new browser app based on the proposed approach. The subsequent section then provides technical details on how to practically set up an environment to run browser apps and to adopt the code to create more apps. This is then followed with a discussion section where we make the case for the browser app approach. Finally, the paper concludes with the highlights of the work, the potential contributions, limitations, and future development work.

THE CHALLENGE

The explosion of smart devices has created some incredibly complex challenges for developers as well as to IS educators and students. Since tackling all these challenges is beyond the scope of our paper, we would like to zero in one specific challenge. That is how to design and develop apps to run on cross platforms. In the academic context, the focus becomes how to teach advanced web design for a mobile environment and, specifically, browser app development given many different platforms and devices.

Let us first take a closer look at this challenge in the context of the following three factors: the complexity when dealing with cross platforms, the effect of large number of native apps, and the fast pace of technology obsolescence. The discussion of these three factors provides the backdrop for our proposed approach and our advocacy for browser apps in the later sections.

The first crucial factor is the current existence of large number of different platforms and devices on the market. Among the major platforms are IOS, Android, Blackberry, and Window Phone. These platforms run on different devices that are manufactured by different companies. While the availability of different platforms and devices offer consumers a variety of choices, they put tremendous burden on individual app developers or smaller app development shops as well as IS educators teaching mobile app development. Each of these operating systems requires completely different developer tools and development frameworks. Knowledge of one tool and one framework may not necessarily be transferable to another. Furthermore, these tools and their associated developer framework are constantly updated with new features.

When we explored and experimented with the development of native apps for Android, iOS, and Windows, we realized how difficult it was to learn the Software Development Kit (SDK) tools and master each platform specific languages. This experience led us to the choice of going with the open source software that is more resilient to changes than the proprietary options. Dealing with different operating systems is just one side of the challenge. The developers of native mobile apps must consider the variety of different hardware. Designing and developing a native app on one specific hardware is already demanding, costly, and time-consuming. Let alone making this specific app to run on so many different devices built on different proprietary hardware. What works on one device does not necessarily work on other devices. For instance, there are lower-level APIs that allow native code to access device storage, sensors, and data (Charland & Leroux, 2011), and all these APIs are different from one device to another. Therefore, to develop a native app for different platforms is technically challenging and economically demanding, especially for smaller shops of developers.

The second factor is the existence of so many mobile apps. According to one estimate, there were more than two million mobile apps on the market just a few years ago. This number could be even higher as the number of smartphones sold was projected to reach one billion at the end of 2015 (University of Alabama at Birmingham, 2014). Just from the count at Google Play Store and Apple App store alone, users have downloaded all the apps billions of times per year (Malavolta, Ruberto, Soru, & Terragni, 2015). According to the 2014 U.S. Mobile App Report (Lella & Lipsman, 2014), the

mobile apps usage alone makes up a majority of total digital media engagement at 52%. Based on the mere number of apps, it seems like a great opportunity for anyone to jump into the market and take advantage of the large demands. However, a closer look at the number reveals an interesting shadow. That is, despite so many apps available, only a small percentage of apps are profitable. Furthermore, the native app market has been dominated mostly by large and established companies. Consequently, apps from smaller companies and/or from individual would have a hard time to compete and succeed (Brodsky, 2015). Therefore, we advocate that the focus of teaching in this context should not be on native app development but instead on mobile web design and web-browser based apps because of its inherent advantages. Such advantages include the following: browser apps' searchable nature, less complex platform to deal with, non-proprietary technology, adaptable to "write once run anywhere" technology. We will elaborate more on these advantages in the subsequent sections.

The last factor is the quick pace of technology obsolescence. The obsolescence affects not only software but also hardware. Any new upgrade in either software or hardware on those mobile devices would have direct impact on the apps that run on those devices. In most cases, the apps would also need to be updated and upgraded. In some case, the changes in OS or hardware render the apps inoperable or incompatible. To update the code in these apps would require time and knowledge and, hence, create tremendous burden on the developers. Therefore, it becomes not only expensive but also difficult to maintain native apps in a cross-platform environment. However, since mobile web design and web apps are based on browser, the update and grade in the code of these apps could be done much faster than that in the native apps. Furthermore, the code and the modification could be done just once and then the apps could run from anywhere. This is one clear advantage of browser apps.

These three factors pose significant challenges to the developers, IS educators, and students when learning and working with native app development. However, these challenges open an opportunity for a fresh look into the advantages and potentials of mobile web design and browser based apps. In the subsequent sections, we present our proposed teaching approach that focuses on browser apps rather than native apps. We begin first with the review of previous work and our on-going progress, and then describe in detail our proposed approach. It is followed with a demonstration of our approach in action and the description of what we have accomplished.

THE PREVIOUS WORK AND ON-GOING PROGRESS

In the article "Learning by Doing: How to Develop a Cross-Platform Web App", Huynh and Ghimire (2015) described their work in the development of the Student Services Web App. This app is a simple cross platform Web App that provides the information of services available to students at a university. The article first presented a detailed step-by-step development process. Then, it provided the illustration of the Student Services Web App: how it works and what features it offers. It also discussed the code written and the development techniques used in the project. The article ended with thoughts for further development of cross platform mobile apps. Based on the successful delivery of a functional prototype as described in Huynh and Ghimire (2015), we have taken steps to advance the project further. The first advancement that we contribute is to add more features/ functionalities to make the original web app better and more useful. The second and more important advancement is our conceptualization of the experiences learned and the knowledge gained into an approach that could help IS educators to teach and IS students to learn the nuts and bolts of browser based app development without a steep learning curve.

Since the introduction of the first version of Student Services Web App, we have made these following enhancements. We added:

- Feature to allow users specify their own content in the database,
- Feature to support flexible number of data fields and different user-defined data types,
- Feature to display map for location and direction,

- Functionalities like supporting video and audio and allowing multiple searchable fields,
- A simplified process for data input such as allowing users to input data directly onto the server,
- Support for international characters/fonts.

These enhancements provide the groundwork for turning the initial Student Services Web App into the proposed approach for teaching and developing a series of the so-called Storing, Retrieving, and Displaying (SRD) browser apps. The reason for naming the category of these apps as SRD is that they are designed specifically to interact with a web database and to perform the three basic functions of storing information, retrieving, and displaying the requested information. Although these apps are similar to other typical database web applications, they are distinctive in the way we design and configure them. Our SRD browser apps allow users to define their own content and feed in their own data without any need for programming. This unique characteristic reflects our contribution in giving the users the ability to create their own specific apps. In other words, by allowing users to come up with their own content and use the content to feed the app, this means that the users can now create and define an app to fit their own specific context or customize their app to meet their own need. By not fixing the number of data fields and not predefining the data types, we enable users to customize their own information according to their need without being constrained by our written code.

Other features such as user-defined search fields, location map display, and support of multimedia and international characters/fonts make the groundwork much more useful and adaptable to use in different contexts including non-English environment. For instance, users can define the data types that match with their specific content including multimedia and context sensitive content. The data types that are currently supported include the following: text, audio, video, image, email, phone, URL, and map. Another unique feature is the ability for users to define their own search criteria. This feature allows users to specify more than one key field and, hence, offer them more robust search/query capability on their content. The support for international characters/fonts allows users to build content for not just English but any foreign language or multiple languages.

Furthermore, as web browsers on mobile devices mature, the environment becomes much more stable and uniform. As a result, browser apps can deliver many performances and functionalities similar to those in native apps. For the case of accessing the database, retrieving, and displaying information from the database, browser apps are better suited for the task than native apps. As mobile web browser continues to proliferate, its apps will be more appealing because they can perform more complex and sophisticated tasks just like native apps but without a steep learning curve.

When we develop our browser-based apps, we choose to stay with the established open standards in the mobile era. We use HTML5, CSS3, JavaScript, and PHP. Given much enhancements in HMTL5 and recent CSS3, we are able to write code that can run on any browser regardless of the hardware or OS and can be responsive to any device screen size. We take advantage of the "responsive web" capabilities enabled by the recent CSS.

To maximize the impact of our coding effort, we apply the concept of "Write once, run anywhere" (WORA). "Write once, run anywhere" is a slogan created by Sun Microsystems to promote the benefits of the Java Platform. WORA was a good idea, but its practical implementation had been limited due to the complexity involved in an abstraction layer between the 'compiled code' and the operating system and processor (Daugherty, 2010). Though the original mechanism for Java Platform is not applicable in a web environment, the idea of WORA is still very powerful. In the context of mobile browser apps, the implementation of WORA is quite feasible. The emergence of responsive web capabilities, mobile-specific features in HTML5, CSS3, and availability of devices with larger, higher resolution screens all have made the mobile Web a reality. That is the wealth of useful information from the Web is now accessible at any time and in any place through mobile devices. Our simple SDR apps are suitable to serve as gateway to such a wealth of information on the Web. Almost all the mobile devices nowadays offer web browsers; therefore, browser apps in general, and our SRD

apps in particular, provide an ideal implementation of WORA. With WORA, developers as well as IS educators and students need to just design, write, and develop code once and then run it many times and adapt it in any context. The process of creating a new web app is as simple as modifying a configuration file. There is no need to rewrite the code as in the case of native apps.

Our proposed SRD apps are the specific type of apps designed to perform these three main functions: Store, Retrieve, and Display (SRD) information from the web-based database. For the scope of this paper, we focus mainly on this category of SRD browser apps. Such apps have many advantages and are quite useful despite their simplicity. One, using the web to access information in a database is an easy and timely way to disseminate information to wider audience. Two, web browser software is much easier to use than any proprietary query tools. Three, the web interface requires few or no changes to the database. Four, being able to retrieve and display information on a mobile device adds value to the information because users have a convenient way to search and find what are relevant and timely to them in any place and at any time. As described earlier, our SRD browser apps are unique in that we design our code to allow users to define their own content, set their own search criteria, and include any available data type to fit their own context. This is how we enable users to create their own browser apps without the need to deal with the underlying code as demonstrated in the next section.

The combination of the above elements enables us to achieve our desire in making the code easy to use in many different contexts and by many different users. We strive to make the code easy for users to adopt and to use our code to create new apps. When we modified the design and rebuilt the code, we put much effort in making them simple so that potential adopters of our code do not have to be programmers to understand and use it. This is the reason why we implemented configuration files as shown in the appendix where custom setting could be made. For instance, to create a new app from an existing app, all we needed to do was to create a database with new content, provide the database connection parameters, and edit the heading, footer and images, and modify the configuration file. The new app that stores, retrieves, and displays new content is ready to run without any major modification of the code.

OUR PROPOSED APPROACH IN ACTION

This section illustrates how we used the proposed approach to create a SRD browser app from the existing app. Specifically, our focus is on the step-by-step demonstration of how we took the code from the Student Service App to create a new SRD browser app. The result shows that by adopting our existing app, one can learn about the mobile app and even create a series of browser-based apps without a steep learning curve. After that, we provide the basic instruction for potential users who want to adopt and deploy our code. The instruction covers the setup of an environment and the process to download code and configure it. The final part in this section offers conceptual details in the design of our code. This would be useful to those who are interested in learning more about the technical aspects in the design of our SRD browser app. We also included more technical explanations on the code for reference in the appendix of the paper.

USING AN EXISTING BROWSER APP TO CREATE ANOTHER NEW APP

The purpose of the illustration is two-folded. One is to review how the Student Services Web App works and the other is to explain what we did to create another new app. The entire process does not require a steep learning curve as in the case of native app.

Let us first recap how the Student Services App works. The operation of the Student Service Web App can be summarized in three steps. The first step is to search by the key field Service/Office. The user also has an option to select other search fields such as Building or Email. The second step is to type in characters or words in the search box. A list of matching records is displayed accordingly. The last step is to select the desired record. The user can choose the desired record by clicking on it. The

app then displays the detailed information requested by the user. Figure 1 shows the interface and functions of Student Services Web App.

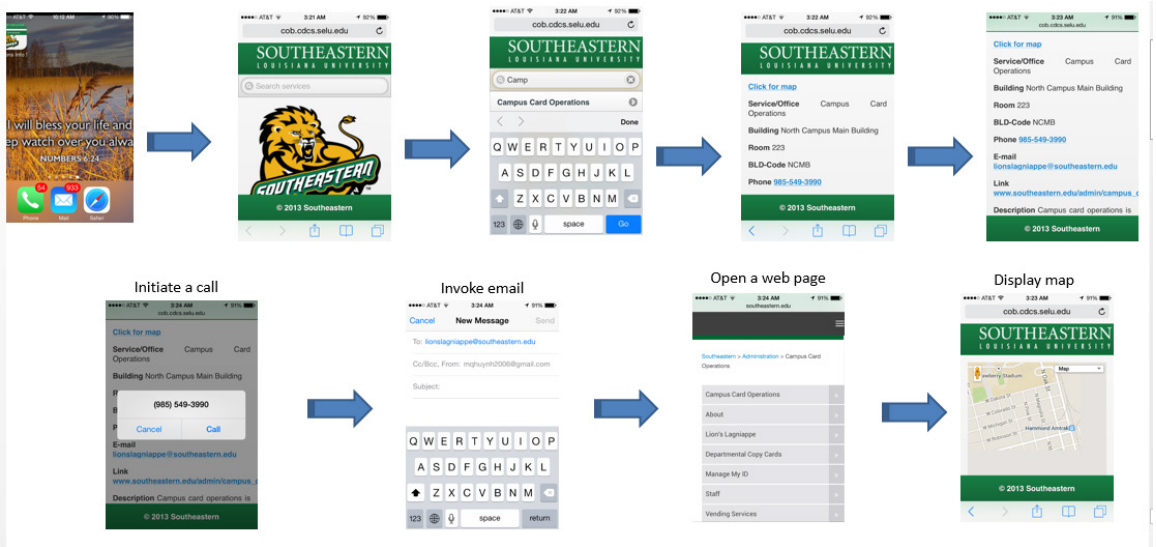


Figure 1. Interface and functions of student services web app

Next, let us describe in detail the process involved in the creation of a new browser app based on the code and the logic design in the Student Services Web App. The new browser app that we presented in this illustration is called the Academic Major Web App. The purpose of this particular app is to provide basic and relevant information to students interested in a certain academic major offered at any typical university. Basically, when users run the app, they can search by the degrees offered and by the Colleges where the degrees are offered.

In responding to the users' search, the app would retrieve and deliver the basic contact information on where and what academic majors are offered at a university. For instance, if one were interested in accounting, the app would provide the college, the location, the phone number, the email, and the web site for details relevant to accounting.

Below is the step-by-step description of how we created the Academic Major Web App using our proposed approach.

The first step is to capture the needed data. In this case, it was academic major data. We gathered all the data related to the degrees offered at our university. We stored the data in a Microsoft Excel file as shown in Figure 2. In this Excel file, we did include the heading row along with rows of relevant data.

	A	B	C	D	E	F	G
1	Colleges	Degrees	Building	Room	Phone	Contact E-mail	Link
2	College of Business	**Accounting (BS)**	Garrett Hall	61	985-549-2052	accounting@selu.edu	http://www.southeastern.edu/acad
3		**Finance (BS)**	Garrett Hall	61	985-549-2052	accounting@southeastern.edu	http://www.southeastern.edu/acad
4		**Management (BA)**	Garrett Hall	56	985-549-2051	management@selu.edu	http://www.southeastern.edu/acad
5		**Business Administration (BBA)**	Garrett Hall	56	985-549-2086	businessadmin@selu.edu	http://www.southeastern.edu/acad
6		**Marketing (BA)**	Garrett Hall	74	985-549-2277	debra.denoux@southeastern.e	http://www.southeastern.edu/acad
7		**Supply Chain Management (BS)**	Garrett Hall	74	985-549-2277	debra.denoux@southeastern.e	http://www.southeastern.edu/acad
8		**Business Administration (MBA)**	Garrett Hall	75	985-549-2146	mba@southeastern.edu	http://www.southeastern.edu/acad

Figure 2. Defining rows and columns for the data to be imported

After we finished capturing the data and stored them in an Excel file, the next step was to transfer the data onto the server. The details of the software and hardware needed for the server are de-

scribed in the subsequent section "Instruction for Adopting and Deploying our Code in Other Contexts". At this point, we assumed that the server environment was already set up and ready for use.

Here, we describe the actual process for the transfer of information from the Excel spreadsheet to an online MySQL database. From our experience, the best format for importing Excel data onto MySQL is to save the data in Excel as OpenDocument. This is done by using "save as type" in Excel to save the data in OpenDocument spreadsheet format. The reason for using OpenDocument is that MySQL can read and import OpenDocument spreadsheet more efficiently and accurately, especially when the data contains foreign fonts or special characters.

On the server side, we logged on our Ubuntu server and accessed MySQL via the program called PhpMyAdmin. We ran PhpMyAdmin from a web browser using this url: http://localhost/phpmyadmin. In PhpMyAdmin, we used the function "Import" to read the data from the OpenDocument spreadsheet and save them into a table in MySQL database on an Ubuntu server as shown in Figure 3.

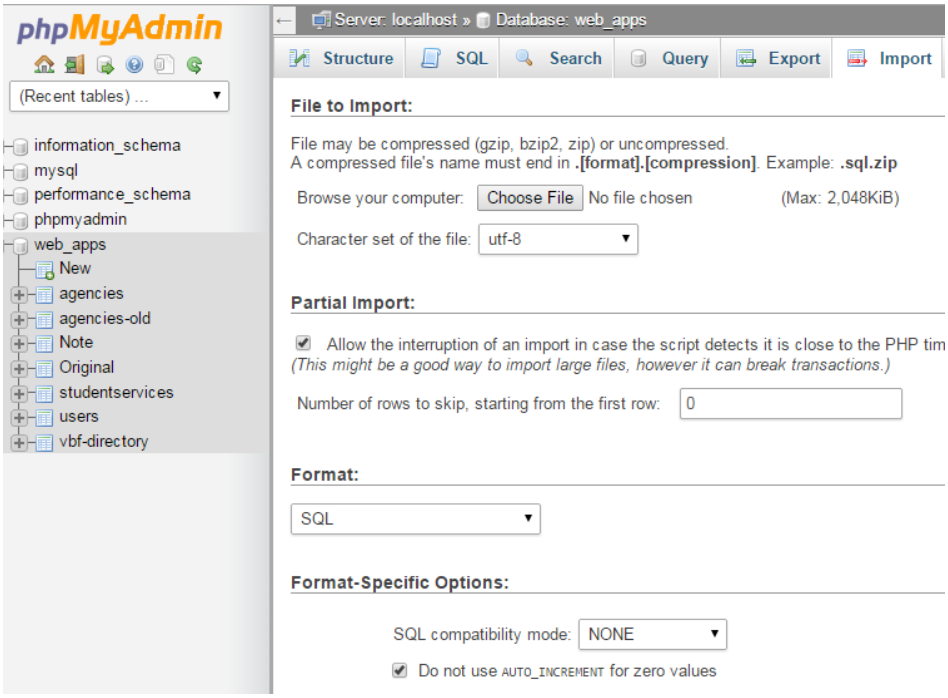
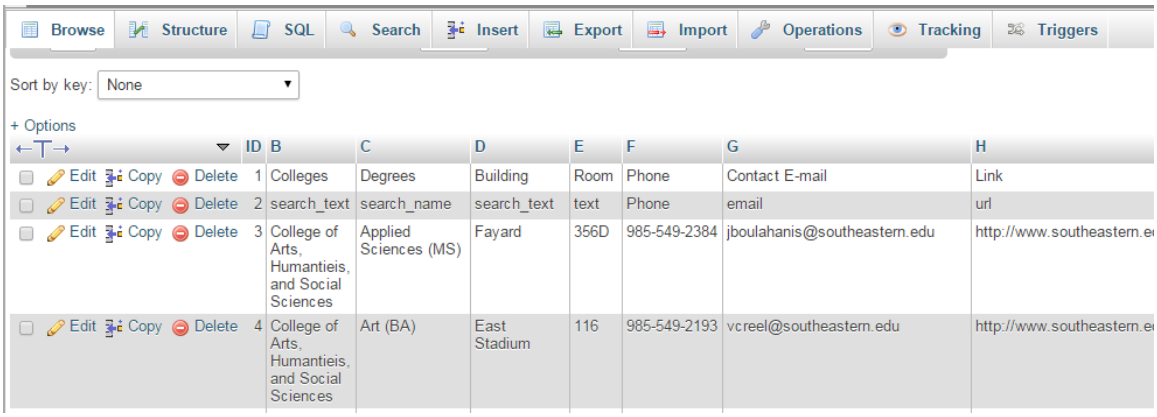


Figure 3. Using PHPMyAdmin to import data from the OpenDocument spreadsheet to create a table in MySQL database

In our case, after we imported the data and saved them in the table called "Academic Majors", we could view the content by clicking "Structure" on the menu. Figure 4 shows an example of what rows and columns in our "Academic Majors" table look like. The first and second rows are unique in our design because they are set up to define the content and the search capability in the app.

Figure 4 shows the first row contained the field names such as Colleges, Degrees, Building, etc. These are the headings in our Excel file. The second row contains the data types such as search_text, search_name, text, phone, email, url. We prepended the word "search" in front of the data type to designate the field to be a searchable field. For instance, "Colleges" with the data type ***"search_text"*** has text as data type and could be searched. "Degree" and "Building" are two other searchable fields. When running the Student Services Web App, all the fields designated with "search_" are displayed as searchable fields as shown in Figure 5. The rest of the rows are just instance of actual records/data.



	ID	B	C	D	E	F	G	H
Edit Copy Delete	1	Colleges	Degrees	Building	Room	Phone	Contact E-mail	Link
Edit Copy Delete	2	search_text	search_name	search_text	text	Phone	email	url
Edit Copy Delete	3	College of Arts, Humantieis, and Social Sciences	Applied Sciences (MS)	Fayard	356D	985-549-2384	jboulahanis@southeastern.edu	http://www.southeastern.e
Edit Copy Delete	4	College of Arts, Humantieis, and Social Sciences	Art (BA)	East Stadium	116	985-549-2193	vcreel@southeastern.edu	http://www.southeastern.e

Figure 4. The content of data in Academic_Majors table after the import step

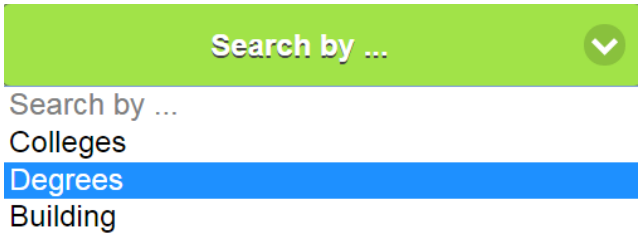


Figure 5. An example of "Search by" searchable fields

The next step was to take all the code from Student Services Web App and modify them to work with the new browser app. As we stated earlier, our design's goal is to make the process easy even to the non-programmer. Although we copied the entire codes of Student Services Web App, to make a new app, all we need to do is to modify the ***config.php*** file as described next.

```php
<?php
//server config
$server = "localhost";
$username = "john";
$password = "john1234";
$database = "web_apps";
$table = "AcademicMajors";

//topbar config
//$topbar_image= "img/cdcs-logo.png";
$topbar_text = "Southeastern LA University";
$home_content_image ="img/cdcs-home.png";
//$home_content_text = "<h1>Welcome to Southeastern!</h1>";


//footer text config
$footer_text = "&copy CDCS 2016";
/* $topbar_text, $topbar_image, $footer_text, $footer_image are
variables to be used for customized header and footer.  For instance,
$topbar_text = "Southeastern LA University";
$home_content_image ="img/cdcs-home.png";
$footer_text = "&copy CDCS 2016";
*/

?>
```

Figure 6. Portion of the actual code in the *Config.php* where customized parameters are set

After copying all the existing code, we edited the ***config.php*** in core folder. We provided the server, username, password, database, and table name that were associated with Academic Major App. These

are the necessary parameters used by other php files to establish connection to the designated database. In the ***config.php*** file, there are also other variables such as ***$topbar_text***, ***$topbar_image***, ***$footer_text***, ***$footer_image.*** These variables provide users options to add customized header and footer. Figure 6 shows part of the actual codes in the ***config.php***. All these parameters are what users need to specify so that all the codes point to the right database, access the specified information, and display user-customized logo, images, header and footer.

After editing the ***config.php*** and including the appropriate parameters, we finished the process. We had successfully create a new SRD browser app and named it Academic_Majors. When running the code, this new app would appear as shown in Figure 7.

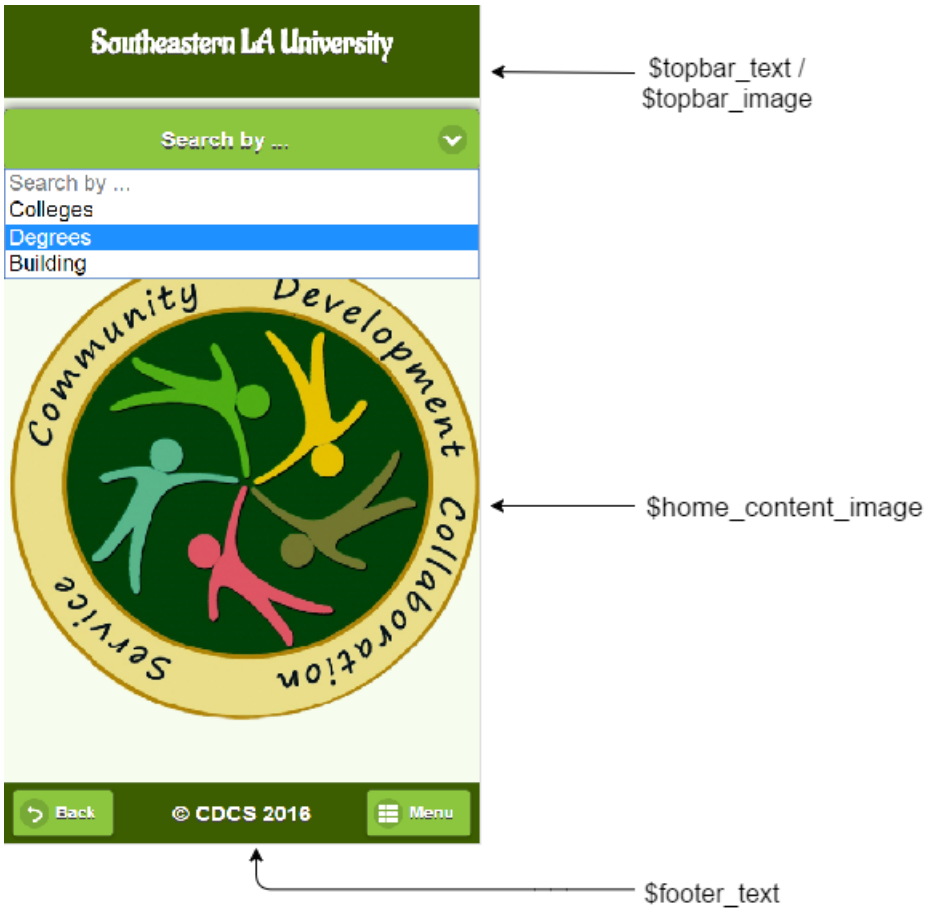


Figure 7. Our Web app interface after customization

As demonstrated in the example above, the code for Student Services Web App was untouched in the process of creating a new app. The only things that we needed to change were the parameters in the ***config.php*** file. The way that we designed and wrote our code is to make it as easy to adopt as possible. Even a non-programmer can pick up the code, adopt it, and use it to create different apps. All this is possible because of our design in using a configuration file which is separated from the actual code.

Another important point that we want to emphasize is the cross-platform nature in our browser app. We used HTML, PHP, JavaScript so that the only thing required to run is a web browser regardless of various operating systems and different hardware devices. Whether the device is an Android smartphone or an iPhone, the browser app will run from the browser. There is no need to change the underlying code. It is much simpler than in the case of native apps where code that was written for Android devices would have to be rewritten for iPhone environment.

When we enhanced the Student Services Web App by integrating different elements in our proposed approach, we came up with code that is quite robust and adaptable. As illustrated above, it does not take much work to create other database-driven browser apps given the existing code. In fact, we have created two additional apps for the communities outside of the university. Other apps currently under planned development include student organizations app and local business directory app.

Instruction for Adopting and Deploying our Code in Other Contexts

In order to share our code and our browser app, we provide a brief instruction here for those who are interested in our browser app and want to use it on their own server environment. Here are the basic software and hardware recommended.

Basic software and hardware include:

- A server computer. We used Dell PowerEdge server.
- Ubuntu server operating system. We used the server OS version 14.04 (or the later) 64 bit.
- MySQL. We used version 5.5.46.
- PHP. We used version 5.5.9.
- Apache. We used version 2.4.7 (Ubuntu).
- PHPMyAdmin utility. We used version 4.0.10.

The above requirement is based on the available resources at our own site. It is possible to have different setup. For instance, instead of using an actual server computer, a virtual server running Ubutu on a cloud hosting service site is a possible alternative. We have tried this setup on DigitalOcean (https://www.digitalocean.com/). DigitalOcean offers a cloud computing environment designed for developers and enterprises. This environment is relative easy to set up and less expensive to pursue.

Since the description of how to set up the server environment is beyond the scope of this paper, we will not discuss the details of the installation here. Instead, we offer the URL below. This URL gives an excellent resource where specific and detailed instruction to install Ubuntu server version 14.04 is available. This URL is http://www.tecmint.com/ubuntu-14-04-server-installation-guide-and-lamp-setup/

Once the environment is set up, then the code for our existing browser App could be downloaded and installed in a folder on the Web server. For instance, on our server, we created a folder WEB-APP/Academic_Majors/ and copied all the codes including the folders here. After that, the users could follow the steps illustrated above to create a new browser app. Here is the summary of those steps necessary in the process of creating a SRD browser App.

- First, set up your server environment.
- Next, download the SRD browser app code available at GitHub.
- Then, set up your database with the data in your context
    - First, identify the need for a database driven app
    - Next, collect the relevant data and store the data in a spreadsheet
    - Then, import the data into the online database such as MySQL
    - Configure the first two rows as the field names and as data types associated with each field name.
- Finally, configure the codes
    - Copy the codes into the appropriate folder on your server
    - Modify the configuration file with appropriate parameters.
    - Run the app with the URL.

Conceptual View of the Design of our SRD Browser App Code

The conceptual view of all the code based on our proposed approach could be visualized in Figure 8. Basically, there are three categories of code written for our SRD browser app. They are Back-end files, View files, and Helper files. Figure 8 shows the relation, dependency, and flow sequence among these files.

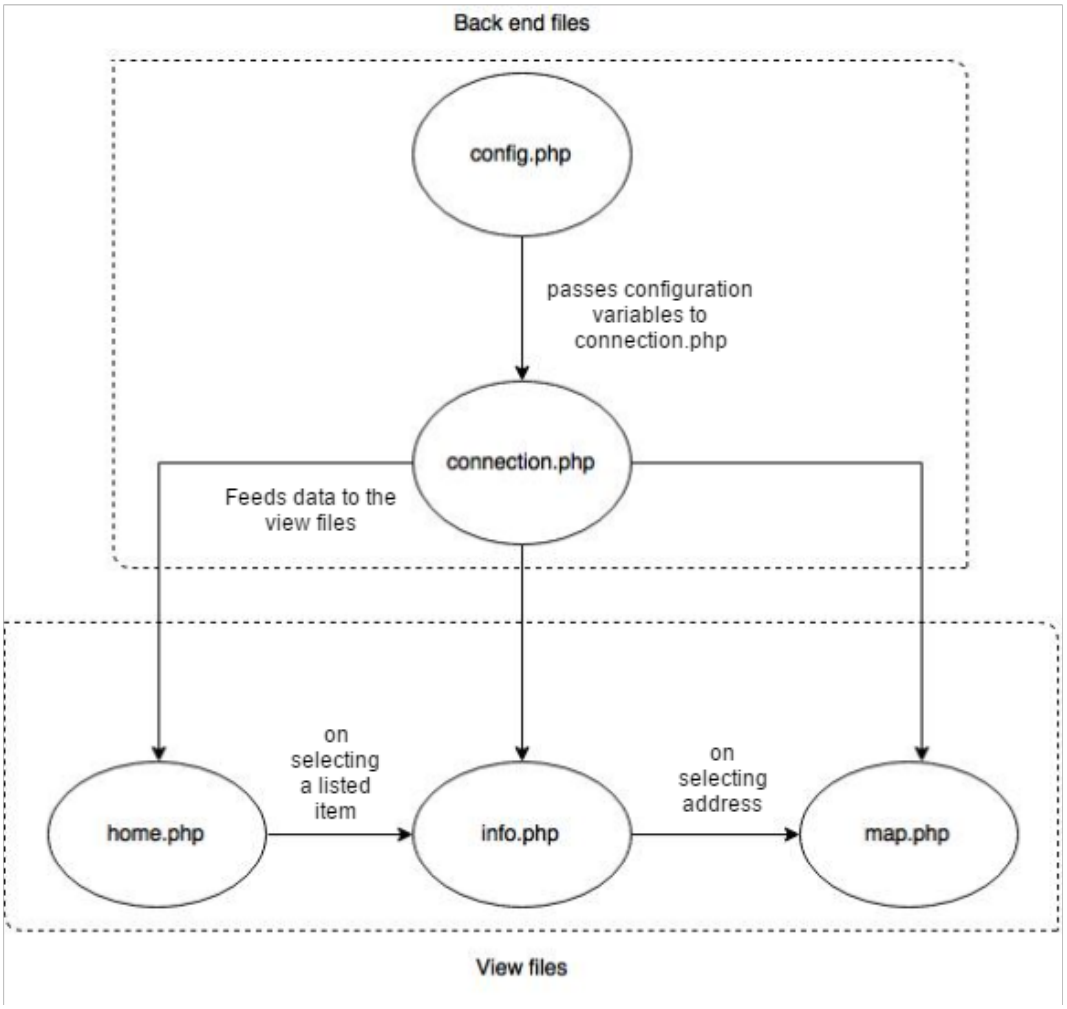


Figure 8. Design logic and flow of the SRD Web App codes

Config.php and ***Connection.php*** are designated as the Back-end files. Their main function is for use in the configuration of a custom browser-based app. All the configuration parameters, such as app name, header text/image, footer text as well as the database access credentials, are specified in ***config.php*** file. The actual code for connecting and accessing the database are written in ***connection.php*** file. In addition, the ***connection.php*** file also extracts raw data from the database and formats them accordingly by using different files such as ***home.php***, ***info.php*** and ***map.php***. Figure 8 shows Back-end files inside the dotted box at the top.

As shown in Figure 8, the View files are inside dotted box below the Back-end files. They include ***home.php***, ***info.php***, and ***map.php*** files. Their main function is to display information retrieved from the database when users run the app and make a search. In a typical sequence, users would land on ***home.php*** when they initially open the app. When a particular item is selected from the search result list, the code in ***info.php*** runs and displays the results associated with the search for the particular item. If the item has geo-location information, the information would be displayed using the

code in ***map.php*** file. As shown in Figure 8, each one of the View files use ***connection.php*** file to access the information from the database.

Not shown in Figure 8 are the Helper files. These files include ***home-content.php***, ***header.php***, ***footer.php*** and ***topbar.php***. Their main function is to allow customized format for the content in the home page and top bar as well as the appearance of the header and footer. For instance, specific colors and logo as well as specific name would be defined in these files. In addition, there are ***utils.php and Helpers.php*** files. ***Utils.php*** contains some helper functions for matching and parsing strings such as URLurl and ***Helpers.php*** defines function that could be reused in different parts of the application in order to cleanse and validate raw data.

Figure 9 provides the basic file and folder structure in the code of a typical SRD browser app.

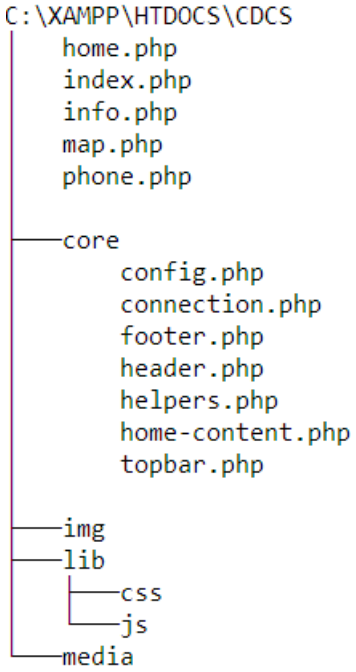


Figure 9. Basic file and folder structure in the code of a SRD browser app

For those who are interested in adopting the codes and creating custom SRD browser apps or using the code to teach an IS or web-app programming course, we provide additional detailed explanations in the appendix. There, we include illustrations of code along with documentation to help users understand much deeper the design of our code.

DISCUSSION

As stated in the Introduction, the rapid growth of smart device usage has led to some complex challenges to developers as well as IS educators and students. One of such challenges is to how to learn and teach the design and development of apps to run across different platforms. Each platform works differently due to its unique combination of hardware and OS. As a result, mobile apps programming languages and tools for developing native mobile apps are platform-specific. Practically, code written for one mobile platform cannot be used on another platform. This means, for instance, the Java code written for an Android app must be rewritten in Objective-C code in order to run on an Apple iOS device. This is one of the major reasons why learning and teaching the development and maintenance of native apps for multiple platforms has been one of the major technical challenges facing not just the mobile development community but also the IS educators and students (Huynh

& Ghimire, 2015; Joorabchi, Mesbah, & Kruchten, 2013; Native, web or hybrid mobile-app development, 2012).

In this paper, we advocate that instead of viewing different OS and devices as platforms for apps, we propose to view and use web browser as a base platform. As described earlier, there are many good reasons for viewing and using a web browser as a platform. With a web browser as a platform, we can push for the development of browser apps rather than native apps. Out of millions of native apps on Google Play and Apple Store, only a small percentage of apps are profitable. The opportunities for small app developers are squeezed to the vanishing point while the cost and the entry barrier are getting higher. On the other hand, browser apps offer many significant advantages (Brodsky, 2015). They give small companies a chance to be visible. There are now "responsive" Web techniques that automatically adjust to the device's display capabilities. Browser apps are searchable and can be accessed just by clicking on a link. Browser apps are easier to develop and update. Separate versions are not required for different devices. With new capabilities from HTML5 and CSS3, the performance gap between browser-based apps and standalone native apps is getting narrower. Successful mobile web apps can do more than merely replicate a normal Website. Responsive Web techniques along with devices having higher resolution screens will make browser-apps more useful and mobile friendly (Brodsky, 2015).

More importantly, the widely availability of web browsers on any mobile device make web browsers an ideal environment for WORA. Together with using web browser as a platform and applying open-source standard technology, it is now practical and feasible to write code once and run it anywhere on any different devices without the technical barriers as in the case of native apps. WORA technology/methodology is the best way to fend off the problem of platform proliferation and quick technology obsolescence (Banerjee, 2011; Daugherty, 2010). It will also reduce the cost incurred in the design and development of apps because code can be reused very efficiently. It is a very efficient reliable approach as we demonstrated in our process of creating a new browser app from an existing app. From our experience, we discover that the category of apps best suited to our proposed approach is the SRD type of browser apps. A simple set of code in the existing browser app can be reused with many different contents. All of these are the reasons and motivations for us to advocate for the teaching and development of an approach that focuses on browser apps as an alternative to the complex and highly technical requirements in native apps.

CONCLUSION

Smartphones are widely used for navigating numerous important life activities ranging from searching for health information, using educational resources, finding jobs, reading news, sharing contents with others, following the driving direction, performing tasks, to staying connected to the world, and many other usages. For many users, these devices serve as a key entry point to the online world (Smith, 2015). Yet, the explosive growth in the use of mobile devices has led to the proliferation of platforms, the creation of millions of apps, and the rapid pace of technology obsolescence. Amid an environment of constant changes, IS educators and students, like developers, are facing some of the incredibly complex barriers not only in keeping up with the technology but also in learning and developing skills to contribute to the tremendous potential in mobile computing.

Throughout the paper, we offer a possible answer to this challenge by sharing our own experience and success in designing, developing, and creating a series of SRD browser apps. We describe in detail our proposed approach to guide the development of those browser apps. In sum, we first followed Huynh and Ghimire's (2015) work on how to develop a cross platform Web App in general. Next, we focused specifically on the available code in the Student Services Web App and enhanced it by adding more features. More importantly, we redesigned the code so that it can be easily used in other contexts. The result leads us to the creation of a series of fully functional Store, Retrieve, and Display (SRD) browser apps in a short time and without a significant learning curve. In the Discussion section, we have made the case for pushing forward with browser apps because there are many

inherent advantages in the approach that we proposed. At the core of our approach are these three principles:

1. Coding once and using the codes in many different contexts—the concept of "Write Once, and Run anywhere" (WORA),
2. Making the code easy to adopt and use by others—the spirit of sharing and contributing to user community,
3. Building browser-based apps from HTML5, CSS3, JavaScript, and PHP—the embrace of common platform and open-source standard and viewing and using web browser as a platform for the app development rather than various OS or specific device.

We have demonstrated how our coding and app development were guided by these three principles above. We illustrated in detail how to use the code in one app and turn it into a different app for use in a different context without much complexity. We explained the code so that other users can understand the inner work of the code and further develop it to build more apps. The process of adopting and using the code is quite straightforward. The case in point is that a simple modification of parameters in the configuration file is all that it takes to create a new browser app. We advocated the use of open source technologies such as HTML5, CSS3, PHP, and JavaScript rather than relying on the native app approach. HTM5L, CSS3, PHP, and JavaScript are commonly used languages and they have a strong community of developers and users.

This work offers a refreshing look into the merit of mobile web design and browser-based app development. It highlights the potentials in teaching and using our proposed approach though simple but powerful. The evidence rests in the successful creation of a series of browser-based apps in a short time and without a steep learning curve. Given that the code was already written and available, users can easily create a series of other "Store, Retrieve, and Display" types of browser-based apps. Each of the browser apps can be configured specifically to work with the user's defined content stored in an online database. Therefore, we hope that our proposed browser app development approach and example will be useful to those who are interested in the app development and to those who teach mobile web design or mobile app development courses. We share the source code as well as documentations in this project so developers/IS educators and students can download, experiment, adopt, and adapt. Any typical users including non-programmers can also take the code, install, configure, and run it with minimal modifications. Moreover, we invite more participation from a community that involves in searching, exploring, and experimenting alternative approaches to that of native apps to push for mobile web computing. Finally, we hope that the merit of our proposed browser-based approach may help to inspire other similar efforts to overcome the complexity encountered in the development, deployment, and management of cross platform mobile friendly apps and to contribute to shape the direction in the future mobile app development.

Limitations and Future Work

As discussed in the earlier section, using the code and applying the browser app approach enable us develop a series of SRD apps quickly and efficiently. We showed that our approach does not require a steep learning curve and extensive technical background. This is beneficial in the context of teaching and learning about mobile app development. However, we acknowledged that we did not address the issue of usability at all in this paper. This is perhaps one of the major limitations in the paper. The usability testing should be integrated into the cycle of app development. We recognize our biases when viewing and using our own apps. Although we think that they are useful and the interfaces are easy to use, such perceptions can only be verified through usability testing. Therefore, it is important to take into account the usability factor. We should have observed how users use the apps in a real-world context and obtained feedback from them for enhancements. By including the evaluation from a user perspective, future study will help to show whether our proposed approach is practically useful and our apps are fully functional and adaptable in different contexts. The evaluation of these apps and the users' feedback on our proposed approach would offer valuable insights for making the code

as well as the approach more comprehensible for adoption by IS educators and more refined for use by students.

With the proposed approach, we are able to produce a series of browser apps for any content and for cross platforms. From the technical perspective, though these browser apps could work on any mobile devices, essentially they are still not yet true mobile apps. They are not downloadable. The characteristics of browser apps is that they are accessed from the Web and run mainly from the browser. Hence, they cannot function without a connection to the internet. Recently, we have explored cross platform SDKs such as Xamarin to turn our browser-based apps into true mobile apps. Xamarin allows developers to build cross platform mobile apps for Android, iOS and Windows using the same C# codebase. Xamarin code could be compiled into native platform codes. After some experience with it, we realized that Xamarin is still not yet a stable platform for our application needs. Many times, just like the native SDKs, Xamarin codes were not backward compatible when there were new SDK version updates. Recognizing this trend, we plan to pursue the future work in developing the SRD web apps into hybrid mobile apps.

Hybrid mobile apps allow developers to use standardized web technologies such as HTML5, CSS3, and JavaScript. This is the same standard used in our proposed SRD framework. However, in hybrid mobile apps, the developers also include the capability to deliver all service requests to the Platform API. This part requires the use of a hybrid development framework (e.g., Apache Cordova, ionic framework, etc.) to provide a native wrapper for containing the web-based code, and a generic JavaScript API to bridge all the service requests from the web-based code to the corresponding platform API. It is the native wrapper that enables hybrid mobile apps to be packaged, deployed, and distributed across platform (Irvine & Maddocks., 2013; Malavolta et al., 2015; Wargo, 2012). Among the benefits with the hybrid mobile apps are cross-platform portability, the reuse of existing knowledge of web developers, simpler and less expensive and less time consuming development processes. In sum, they allow developers to create a single mobile app using web standards, and to consistently distribute it across multiple mobile platforms with (minimal to) no changes (Malavolta et al., 2015). Therefore, hybrid mobile app approach might offer us a mechanism to turn our SRD browser-based apps into mobile apps that could be compiled into binary executable files, could be downloaded from App stores onto users' devices, and could be run both online and offline. This is what we plan to do in the next phase of our development.

REFERENCES

Banerjee, U. (2011, December 6). *Write Once Run Anywhere (WORA) or cross platform mobile development tools – A comparison.* Retrieved September 18, 2016 from https://setandbma.wordpress.com/2011/12/06/wora-platform-comparison

Brodsky, I. (2015, December 21). Deathmatch: The mobile Web vs. mobile apps. *Computerworld.* Retrieved September 18, 2016 from http://www.computerworld.com/article/3016736/mobile-wireless/the-mobile-web-vs-mobile-app-death-match.html

Charland, A., & Leroux, B. (2011). Mobile application development: Web vs. native. *Communications of the ACM*, *54*(5), 49-53.

Daugherty, E. (2010, February 17). *Does Write Once Run Anywhere work?* Retrieved September 18, 2016 from https://dzone.com/articles/does-write-once-run-anwhere

Huynh, M. & Ghimire, P. (2015). Learning by doing: How to develop a cross-platform web app. *Journal of Information Technology Education: Innovations in Practice, 14,* 145-169. Retrieved February 19, 2015 from http://www.jite.org/documents/Vol14/JITEv14IIPp145-169Huynh1842.pdf

Irvine, M., & Maddocks, J. (2013). *Enabling mobile apps with IBM worklight application center.* IBM Redbooks.

Joorabchi, M., Mesbah, A., & Kruchten, P. (2013). Real challenges in mobile app development. In *Empirical Software Engineering and Measurement*, *2013 ACM/IEEE International Symposium on* (pp. 15–24).

Lella, A., & Lipsman, A. (2014). The U.S. mobile app report, 2014. *comsCore* [white paper]. Retrieved November 18, 2015 from https://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report

Malavolta, I., Ruberto, S., Soru, T., & Terragni, V. (2015). End users' perception of hybrid mobile apps in the Google play store. *Proceedings of the 4th International Conference on Mobile Services : IEEE*. Retrieved November 18, 2015 from http://www.ivanomalavolta.com/files/papers/MS_2015.pdf

Native, web or hybrid mobile-app development. (2012, April). IBM Corporation. [White paper] Document Number: WSW14182USEN.

Smith, A. (2015, April 1). U.S. smartphone use in 2015. *PewResearch Center Report on Internet, Science & Tech*. Retrieved October 1, 2015 from http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/

University of Alabama at Birmingham. (2014). *The future of mobile application*. Online Masters in Management Information Systems. Retrieved November 18, 2015 from http://businessdegrees.uab.edu/resources/infographic/the-future-of-mobile-application

Wargo, J. (2012) *PhoneGap Essentials: Building cross-platform mobile apps*. Addison-Wesley.

APPENDIX

TECHNICAL EXPLANATIONS ON CODES WRITTEN AND DESIGN LOGIC IN THE SRD WEB APP FRAMEWORK

For those who are interested in adopting the codes and creating custom SRD web app or using the codes to teach an IS or web-app programming course, this section will be helpful and relevant. Since we use GitHub repository hosting service to store the SRD Web App codes, users can easily obtain the source codes as well as the documentations at this URL: https://github.com/mqhuynh/Mobile-CMS

In this section, we focus on explaining the design logic as well as selected parts of the codes that may not be obvious to users. As we described, we included the actual codes so it will be easier for users to follow.

When the SRD Web App first run, the file index.php is invoked. The main purpose of this file is to determine the type of device that users are using. It detects whether users are running the App from either a mobile device or a computer. If the device is a mobile device, then home.php is loaded. Otherwise, phone.php is loaded. This is achieved by the codes shown in Figure A1, in particular the function isMobile.any(). Phone.php is basically our way of providing users on a computer an emulated mobile view of home.php to have the look and feel of the app on a mobile device even though the App itself is run on a computer browser.

```
// Opens normal view if the client device is mobile.
if(isMobile.any()){
    window.location.href = "home.php";
}

//If not, open emulated mobile view.
else{
    window.location.href = "phone.php";
}
```

Figure A1. isMobile.any() function - code snippet from index.php

Home.php/phone.php loads connection.php, header.php, topbar.php, home-content.php and footer.php as shown in Figure A2. These files are used to establish connection to the database. In addition, they also load the external CSS and JS libraries, the top heading bar of the app and display the

content in the app home page as well as the footer of the App respectively. Home.php/phone.php uses all these resources to display the homepage of the application.

```php
<?php
require_once("core/connection.php");
require_once("core/header.php");

...

foreach ($list_of_items as $item) {
  echo $item;
}

require_once("core/home-content.php");
require_once("core/footer.php");

?>
```

Figure A2. Code snippet from home.php/phone.php

On searching and selecting different services, info.php file is opened to show the information corresponding to that particular service. Info.php file also uses connection.php, header.php, topbar.php and footer.php as shown in Figure A3 for the same purpose as of home.php/phone.php. This file loops through each fields of selected item from the searchbox in home.php. The data is then displayed in an understandable form according to their different data types. For instance, a field with datatype as image will display image instead of image link using HTML img tag as shown in Figure A3.

```php
<?php
    foreach ($fields_of_row as $data) {
        if($datatype == "email"){
            echo '<a href="mailto:'.$data.'">'.$data.'</a>';
        }

        ...

        if($datatype == "image")){
            echo '<img src="'.$data.'"/>';
        }
    }

?>
```

Figure A3. Code snippet from info.php

Map.php displays the location of an "address" datatype using Google Maps API. When an address is clicked in info.php, the corresponding latitude and longitude of the address is ascertained using the API and the map is displayed. The codes that accomplish the tasks of getting the location and displaying the map is shown in Figure A4.

Config.php allows users to configure database connection parameters as well as different custom content in the App such as homepage appearance, footer text and top bar image (Figure A5).

```
function initialize() {
  // set center to the address location, appropriate zoom lable and map type.
  var mapProp = {
    center:latLng,
    zoom:18,
    mapTypeId:google.maps.MapTypeId.ROADMAP
  };
  var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
  var marker = new google.maps.Marker({
    position: latLng,
    map:map,
  });
  marker.setMap(map);

  //Shows name and address on clicking the map marker
  google.maps.event.addListener(marker, 'click', function() {
    map.setCenter(marker.getPosition());
    var info = new google.maps.InfoWindow();
    info.setContent(name+", "+address);
    info.open(map,marker);
  });
}
google.maps.event.addDomListener(window, 'load', initialize);
```

Figure A4. Code snippet from map.php

```
<?php
//server config
$server = "server name"; // link to the server
$username = "database_username";
$password = "yourpassword";
$database = "database_name";
$table = "table_name";



//topbar config

// Either, link of the image that appears at the topbar
//$topbar_image= "img/cdcs-logo.png";

 // or text the goes on the topbar of the app.
$topbar_text = "Southeastern LA University";

// image that goes on the homepage of the app
$home_content_image ="img/cdcs-home.png";

 // Text/Message that goes on the homepage of the app
//$home_content_text = "<h1>Welcome to Southeastern!</h1>";

//footer text config
$footer_text = "&copy CDCS 2016";
?>
```

Figure A5. Code snippet from config.php

Connection.php establishes connection to the MySQL server using the configuration set in config.php file. The data extracted from the database are stored in different arrays to use in different part of the application.

```php
<?php
// to use the connection setting from the config.php file
require_once('core/config.php');

// file containing functions to check data validity and other util functions
require_once('core/helpers.php');

$connection = new mysqli($server, $username, $password, $database);
$query = 'SELECT * FROM '.$table;
$result = $connection->query($query);

while($row = $result->fetch_array(MYSQLI_NUM)){
    if($is_first_row){
        storeAsFieldName();
    }
    else if($is_second_row){
        storeAsDataType();
    }
    else {
        storeAsActualData();
    }
}

?>
```

Figure A6. Code snippet from connection.php

Home-content.php contains content that goes in the homepage of the application. It is embedded inside home.php.

```php
<?php
// display the homepage content as configured in config.php file.
 if(isset($home_content_image)){
    echo '<img width="90%" src="'.$home_content_image.'" />';
 } else if(isset($home_content_text)){
    echo $home_content_text;
 }
?>
```

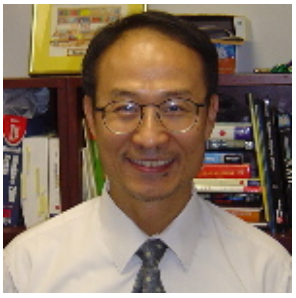
Figure A7. Code snippet from home-content.php

Helpers.php defines function that can be reused in different parts of the application in order to cleanse and validate raw data.

```php
<?php

// this function generates a valid link from different types of values.
function linkify($link){

    // local file links are going to start with "&" to remove confusion,
    // strip out "&" from the string to create a valid link.
    if(strpos($link, "&") === 0){
        return substr($link, 1);
    }
    // if the url is empty or null, return empty string
    if(empty($link) || is_null($link)){
        return "";
    }

    // if the url does not start with "http", append "http://" infront of it.
    if(!(strpos($link,"http") !== false)){
        $link = "http://".$link;
    }
    return $link;
}

// validates link if they are not empty or not blank.
function is_good_link($url){
    return !empty($url) || !($url == "");
}
?>
```

Figure A8. Code snippet from helpers.php

BIOGRAPHIES



Minh Q. Huynh is an Associate Professor at Southeastern Louisiana University. He received his Ph.D. from State University of New York at Binghamton, and his B.S. in Computer Sciences from University of Maryland University College. His teaching interests are in the areas of Digital Marketing, Web design, Introduction to MIS. Research interests include software development using open source, web database implementation, web apps, technology supported distance education, IT for small businesses. His publications appear in journals such as the Journal of Information Technology Education: Innovations in Practice (JITE:IIP), the Communications of ACM, Journal of AIS, Communications of AIS, European Journal of IS, Journal of Electronic Commerce in Organizations, Journal of Information Systems Education, Business Studies Journal.

Prashant Ghimire completed his Bachelor degree in Computer Science from Southeastern Louisiana University. He is passionate about modern JavaScript technologies, mobile application development and UI/UX design. He has worked with many popular web development technologies such as AngularJS, Ionic, Loopback, jQuery, Bootstrap, ASP.NET, Laravel, CakePHP, MySQL, MongoDB, etc. Prashant has also been involved in research project on web application development and cloud computing. His work on "Linking QR code to service learning" was featured at the IARSLCE-Research poster session. His research on QR Code/Barcode processor was published on Business Studies Journal. He also presented this research at the 6th Annual General Business Conference at Sam Houston State University. He is currently working as a Software Engineer at Leviton Security & Home Automations.