A Prototype Android Food Ordering Application

GRADUATE PROJECT REPORT

Submitted to the Faculty of
The School of Engineering & Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Johan Emerensiana
Summer 2013

Committee Members

Dr. Ajay K. Katangur
Committee Chairperson

Dr. Scott A. King
Committee Member

Dr. Dulal Kar
Committee Member
ABSTRACT

This report presents a smart phone application for ordering food through an Android device. The application sends a request and receives a reply from a server. The server processes the request based on the communication tags. This tag makes one request or reply differ from another request or reply. This application is designed to help students, faculty, and staffs at Texas A&M University – Corpus Christi to look up what food is being served, where the food services are located, and how to place the order. The requirements, design, and description of the implementation are presented.
# Table of Contents

ABSTRACT .............................................................................................................................. ii

Table of Contents .................................................................................................................. iii

List of Figures ......................................................................................................................... v

1. BACKGROUND AND RATIONALE .................................................................................. 1
   1.1 What is Android? .......................................................................................................... 2
   1.2 Why choose Android? .................................................................................................. 5
   1.3 Motivation .................................................................................................................... 6

2. REQUIREMENTS AND APPLICATION DESIGN ....................................................... 7
   2.1 Requirements ............................................................................................................... 8
   2.2 Application Design ..................................................................................................... 8
      2.2.1 Consumer end-user (Android application) .............................................................. 11
      2.2.2 Database (MySQL), hypertext preprocessor (PHP), and web server (Apache) .... 18
      2.2.3 Vendor (PHP website) ......................................................................................... 25

3. USER STUDY .................................................................................................................... 28

4. TESTING AND RESULTS ............................................................................................... 29
   4.1 Consumer end-user (Android application) module ...................................................... 29
      4.1.1 Registration process .............................................................................................. 29
      4.1.2 Login and order placing process .......................................................................... 31
   4.2 Database (MySQL), hypertext preprocessor (PHP), and web server (Apache) modules ................................................................................................................................................. 33
   4.3 Vendor end-user (PHP website) module ...................................................................... 33
5. CONCLUSION AND FUTURE WORK .................................................................37

5.1 Limitations ..............................................................................................38

5.2 Future Work ............................................................................................38

BIBLIOGRAPHY AND REFERENCES .............................................................39

Appendix A: Android application end-user .....................................................40

A.1 What the user will see ..............................................................................40

A.1.1 Dashboard xml .....................................................................................40

A.1.2 Login xml ..........................................................................................41

A.1.3 Register xml .........................................................................................42

A.1.4 Hallway xml .........................................................................................43

A.1.5 Info xml ................................................................................................44

A.1.6 The University Center main xml and Subway main xml .....................45

A.1.7 Checkout xml ........................................................................................46

A.1.8 Thank you xml ......................................................................................47

Appendix B: Code Snippets ............................................................................48

B.1 Android application code snippets ..........................................................48

B.2 PHP snippets ..........................................................................................50
List of Figures

Figure 1.1 Android Architecture.................................................................2
Figure 2.1 The system design.................................................................10
Figure 2.2 Adding tag to a connection......................................................11
Figure 2.3 Creating a connection to the web server.................................12
Figure 2.4 Android end-user design flowcharts........................................14
Figure 2.5 SHA-1 Hash Algorithm implementation..................................17
Figure 2.6 Request and Reply Scheme.....................................................19
Figure 2.7 Processing and replying to a request.........................................21
Figure 2.8 PHP-DOM parser first module................................................24
Figure 2.9 PHP-DOM parser second module............................................25
Figure 2.10 Vendor’s retrieves data from database....................................27
Figure 4.1 Scenario John Doe – registration process..................................30
Figure 4.2 Scenario John Doe – login and placing order process (part 1).........32
Figure 4.3 Scenario John Doe – login and placing order process (part 2).........33
Figure 4.4 Sanddollar admin registers John Doe on the database.................34
Figure 4.5 After a successful deposit, the system gives report to the sanddollar admin ...34
Figure 4.6 Ultimate Baja sees the incoming order......................................35
Figure 4.7 After the order is prepared, Ultimate Baja presses “Ready” button....35
Figure 4.8 After the order is picked up, Ultimate Baja presses “Gone” button....36
Figure A.1 Dashboard xml....................................................................40
Figure A.2 Login xml............................................................................41
Figure A.3 Register xml........................................................................42
Figure A.4 Hallway xml. ..........................................................43
Figure A.5 Info xml. ..........................................................44
Figure A.6 The University Center main xml and Subway main xml. ..............45
Figure A.7 Checkout xml.........................................................46
Figure A.8 Thank you xml.........................................................47
Figure B.1 Singleton HTTPContext object and cookies header on every HTTPPost ......49
Figure B.2 PHP-DOM module 1. ................................................50
Figure B.3 PHP-DOM module 2. ................................................51
1. BACKGROUND AND RATIONALE

The burgeoning population of Texas A&M University – Corpus Christi with over 10,000 students representing 48 states and 67 foreign countries is unavoidable [TAMUCC 2013]. In addition, there are about 1900 faculty and staffs bringing the population of Texas A&M University – Corpus Christi to around 11,900. Consequently, with this growing population the number of food services at Texas A&M University- Corpus Christi has increased to six at various locations around the campus [TAMUCC Food 2013], including:

- The University Center
- Mary and Jeff Bell Library
- Island Hall
- Center for Instruction Learning
- Bay Hall
- Faculty Hall

The food services that are being provided at the various locations are:

- The Commons Market
- Subway
- Einstein Bros®
- Ultimate Baja
- Simply To-Go Kiosk
- Starbucks
Unfortunately, the service is inadequate considering the campus population. Resulting in long lines at both the cashier and the service lines during peak times, therefore the intent and purpose of this project is to resolve this problem with the ability to order online using a mobile application developed in android. The project has a limitation where the company also increases the number of employees to handle the incoming orders from both the food counter and the mobile application online.

1.1 What is Android?

To develop a robust Android application, it is important to understand how the operating system works. The Android architecture is indicated below.

![Android Architecture](image)

Figure 1.1 Android Architecture.
Wang et al. (2011) wrote that, “Android is a software stack for mobile devices that includes an operating system, middleware, and key applications” [Android Architecture 2011]. A brief introduction of the various levels as shown in Figure 1.1 is:

1. Applications level

The Android operating system comes by default with several core applications, such as, email client, SMS software, calendar, maps, browser, contacts, Global Positioning System (GPS) and several others. These core applications and any developers’ applications reside and run at the applications level and are written using Java.

2. Application Framework level

Android provides an open development platform giving developers the ability to build extremely rich and innovative applications. With this, developers have full access to the same framework APIs that are used by the core applications and are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much more. For instance, developers’ applications can execute core applications.

Java programming language is an Object Oriented Programming language that is supported by the powerful Java libraries. The Android architecture design allows developers to reuse and overwrite each component using the Java libraries. This is the same mechanism that allows components to be replaced by other developers.

The underlying applications are a set of services and systems, which include Views, Content Providers, Resource Manager, Notification Manager, and Activity Manager.
3. Libraries level

Android comes with a set of C/C++ libraries, System C library, Media libraries, Surface Manager, LibWebCore, SGL, 3D libraries, FreeType, and SQLite, used by the Android system and are available for developers through the Android application framework.

4. Android runtime level

Every Android application runs in its own process, with its own instance of the Dalvik Virtual Machine (VM) and has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format and optimizes for minimal memory footprint. The VM is register-based, and runs classes complied by a Java language compiler that have been transformed into the .dex format by the included “dx” tool. However, the Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management. Lin et al. (2011) concluded that Dalvik Java code is faster than native code, but in some conditions, it performs badly; therefore, software designers are encouraged to create an efficient application [Dalvik 2011].

5. Linux kernel level

Android relies on Linux kernel version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

Google Inc also provides two kits for the Android development platform [Android
1. The Android Software Development Kit (SDK) provides modular packages for developing applications that run on Android-powered devices.

2. The Android Native Development Kit (NDK) provides a toolset for programmers to implement parts of components that make use of native code in Android applications such as C/C++ language.

1.2 Why choose Android?

The Android architecture design has become an interesting field for software developers, expert programmers, and even new Android programmers. As a result, the growth of Android applications in the market is inevitable compared to other mobile operating system. Some of the main advantages of using Android are:

1. Openness and Open Source

The main advantage of developing Android applications are that a large number of Android applications available on the Internet and also the operating system is open source allowing anyone to access and use the source code.

2. Java

The advantages of using Java programming language are, it is simple, object-oriented, and familiar; robust and secure; architecture-neutral and portable; high performance; interpreted, threaded, and dynamic. The power of Java is proved on large scale applications, web servers, and support on consumer devices. Furthermore, Android applications are developed with Java, which is the most widely used programming language. As mentioned above, Java is an object oriented programming language that
allows the programmer to access powerful libraries [Android Developer 2013]. It is intended to let application developer’s code developed for one platform run on other platforms without recompiling.

3. Multitouch Screen

Android smart phones allow users to interact through gestures, such as touch, double tap, long press, drag, and pinch zoom.

4. Sandbox

Furthermore, Wei et al. (2012) mentioned about Android’s sandbox. Android’s sandbox allows users to personalize each application. It specifies how the applications are implemented on the mobile device, such as installation, update, and rejection. This method allows the user to adjust the security decisions for them [Android Privacy 2012].

1.3 Motivation

The increasing number of population must be adopted by an increasing number of food services. As a result, a long line on both cashier and counter is unavoidable. On the other hand, the growth of the Android application market increases significantly. As mentioned above, the reason behind the growing number of Android application market is Android architecture that allows programmers to access Java powerful libraries and develops a robust Android application. This project develops Android application that allowing user to place a food order through the Android mobile device. This project also implements client-server design scheme that proved as scalable network design. Therefore, implementation of client-server design scheme on Android application is expected to bring scalability in service for the user convenience.
2. REQUIREMENTS AND APPLICATION DESIGN

The purpose of this project is to increase the convenience of customers by providing scalability in service. Mobile smart phones allow many people to interact with the application as if they were interacting with their home or work computer, thanks to the operating system. Many applications with different purposes have been written, documented, published, and maintained till present; such as, mobile banking, email applications, travelers’ applications, news weather applications, and many more. All of them have the same purpose to make peoples’ lives easier as well as the purpose of the technology itself. Furthermore, the convenience costs money for the users, but, it makes money for the developers.

This project is developed with:

- Eclipse Integrated Development Environment (IDE)
- Android Developer Tool (ADT)
- Android Software Development Kit (SDK)

What is Eclipse IDE?

The complete description word of “Eclipse IDE” is a multi-language software development environment comprising a base workspace and an extensible plug-in system for customizing the environment. It can be used to develop applications in Java and any other programming languages, such as, C, C++, COBOL, Fortran, Perl, PHP, Python, Ruby, and many more.

One of the features of Eclipse is its powerful multi-language development
platform. Android takes advantage of this feature by creating an ADT plug-in. This plug-in is designed to give programmers a powerful integrated environment to build Android applications. ADT extends the capabilities of Eclipse to let programmer quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug the applications using the Android SDK tools, and even export signed or unsigned .apk files in order to distribute the applications [Android Developer 2013].

2.1 Requirements

Certain requirements must be applied as follows:

- Android API 8 (Android Froyo 2.2) with internet connectivity.
- Apache version 2.2.24
- PHP version 5.5
- MySQL version 5.5.30-cll-lve
- Google Chrome browser version 27.0.1453.110 m (or later)

2.2 Application Design

By the time this project is created, it acknowledges two different data sources:

1. Data retrieved from the website: a data with a valid reference.
2. Data stored within the application: a data without a valid reference.

Therefore, this project handles these two types of data sources differently, the data retrieved from the website with a parsing processing and consequently, the data stored within the application with a different processing. The data retrieved from the website is
the University Center – classics order; this data can be seen on http://tamuccislanddining.com/dining/index.html. From the link above, it only displays the menus that are served, but it does not show any pricing. So, the Android application does not implement any pricing for the University Center – classics order. On the other hand, the data stored within the application are any other menus or locations except the University Center – classics order. The Android application makes a sample pricing system. The purpose of the sample pricing system is to demonstrate the whole design of this application.

There are two types of vendors:

1. A vendor which has a different menu every day.
   
   It is located at the University Center especially on the Classics section. The menus are available at http://tamuccislanddining.com/dining/index.html.

2. Vendors which have a common menu every day.
   
   Except the Classics section at the University Center, the rest of the vendors have common menu every day, such as, Subway, Einstein Bros, Ultimate Baja, Simply To-Go, Starbucks, and the other sections of the University Center.

As mentioned above, the project has two different data sources. Hence, the project specifies the data retrieved from the website as a dynamic data and data stored within the application as a static data. For static data source, the project handles it as hard coded menus. On the other hand, the project handles dynamic data source by using PHP-DOM as a parser; the project retrieves the menus from http://tamuccislanddining.com/dining/index.html. The PHP-DOM describes further in
section 2.2.2. The project focuses on dynamic data source. However, every software and application design must meet several requirements for it to be functionally running.

Client-server design scheme is a well-known design scheme that many developers have used. This project also implements the same design scheme.

This project contains three modules:

1. Consumer end-user (Android application)
2. Database (MySQL), hypertext preprocessor (PHP), and web server (Apache)

3. Vendor end-user (PHP website)

2.2.1 Consumer end-user (Android application)

This application allows the user to browse through available food services on the campus and the menu information for each service. The user can then select the food that he/she wants to order. The application allows the user to view their current subtotal before submitting the order. The food services are categorized by the location of the food service on the corresponding building. For users’ confidentiality, SHA-1 160-bit with output of binary string of 40 hex digits is used in developing the application. The SHA-1 hash algorithm is implemented on user passwords on both registration and login processes.

Android snippets

A client makes several connections to the server through the Android application on the device. On a bigger scenario, many clients make many connections to the server. The only way to handle this scenario is by having a unique tag name for each connection. Therefore, each connection is distinct from others. Figure 2.2 shows how to use a tag for a connection.

```java
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("vendor", ID_KEY));
params.add(new BasicNameValuePair("name", name));
params.add(new BasicNameValuePair("student_id", student_id));
```

Figure 2.2 Adding tag to a connection.
This project uses three main distinct tags:

- **vendor**: it holds the value of the designated vendors. For example, if user makes a connection to starbucks vendor, then *vendor* holds "starbucks" value. This tag is also used for making a connection to the server.

- **name**: it holds the value of the name of the user who initiates a connection. This tag becomes a primary key in retrieving the user’s balance from sanddollar database. Also, it becomes a distinct order on the vendor’s database.

- **student_id**: it holds the value of the student A number of the user who initiates a connection. Similar to *name* tag above, this tag becomes a secondary key in retrieving user’s balance on sanddollar database. Also, it becomes a distinct order on the vendor’s database.

```java
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpPost httpPost = new HttpPost(url);
httpPost.setEntity(new UrlEncodedFormEntity(params));

HttpResponse httpResponse = httpClient.execute(httpPost);
HttpEntity httpEntity = httpResponse.getEntity();
InputStream is = httpEntity.getContent();

BufferedReader reader = new BufferedReader(new InputStreamReader(is, "iso-8859-1"), 8);
StringBuilder sb = new StringBuilder();
String line = null;
while ((line = reader.readLine()) != null) {
    sb.append(line + "n");
}
is.close();
```

Figure 2.3 Creating a connection to the web server.
In Figure 2.3, the *url* holds the address of the server and the *params* is a list of
tags, as shown in Figure 2.2, that are sent to the server on a connection request. Each
connection is encoded by the JSON Format. Figure 2.3 is a default request to the server;
however, this project implements PHP SESSION on every request to the server and
maintains the *same* login session key. The snippet of embedded PHP SESSION request is
listed in the Appendix B. The design of application is shown in Figure 2.4.
The screenshots of each screen for all the activities are listed in the Appendix A. On application start, the first screen that is loaded is the dashboard activity screen. It contains a link which leads to the login activity screen. The login activity screen has a login button and a link to access the register activity screen. A new user can perform a
registration on the register activity screen. A message will be displayed on successful or unsuccessful registration. The register activity screen has a register button and a link to access the login activity screen. After successful registration, the user can press the link to access the login activity screen. On unsuccessful login, a message will be displayed to the user. After successful login, the application displays the hallway activity screen. The hallway activity screen shows options for six different buildings where the food services are available. Each of the places contains two activities:

1. Info activities which show information of the food service available in the corresponding building.
2. Main activities which show the menu for the food and drink which can be selected by the user. The food pricing is also listed along with the items.

After the menu is selected, the user can proceed to the checkout process. The checkout process contains two activities:

1. Checkout activity which displays the order and the subtotal before final submission. It provides a detailed list of the menu with the pricing information.
2. Thank you activity which shows a “thank you” for the purchase and from which building the order can be picked up. It also displays the order subtotal.

Dashboard activity

One unique behavior of the dashboard activity is it notices the user login status. If the user login status is active, then the dashboard activity displays the user name. On the opposite, if the user login status is non-active, then dashboard activity does not display
the user name. The dashboard activity screen has a link that enforces the login status to be non-active before the user activates the login status. The screenshot is provided in Appendix A.

For security reasons, the dashboard activity is executed first when the application is started. It enforces the login status to be non-active before the user activates the login status. For instance, one scenario, student A forgets to log out his/her account on this application, and then he/she lends the phone to student B for placing an order. Since student A forgets to log out, student B could access the application without the login process and place an order using student A’s sanddollar account. It means student A loses his/her allowance.

The key of the unique behavior of dashboard activity is the internal database on the Android device. The internal database on the Android device is SQLite. The login activity creates a login table in the internal database that maintains the user name and password internally. When the application is executed, the dashboard activity sends a query to retrieve the entry on the login table. If the login table has an entry, the dashboard activity displays the retrieved user name. Conversely, dashboard activity does not display any user name. In addition, when the logout button is pressed, it deletes the entry on the login table. Therefore, the unique behavior of the dashboard activity ensures the login table has at most one entry.

**Encryption implementation**

For this project, SHA-1 160-bit with an output of binary string of 40 hex digits is used in developing the application. When a new user registers an account, the password is
hashed on the android application module. Then, it sends the registration information to the server and stores it into a database. On the login process, the Android application module retrieves information from the server database and compares the two hashed passwords, hashed password from the android application and hashed password from the server database.

On a downside, this project cannot deliver a secure HTTPS protocol. In order to implement HTTPS protocol, an SSL certificate is required. There are two types of SSL certificates: self-signed and third-party-signed. A self-signed certificate is free, but third-party-signed is not free, such as, VeriSign Certificate Authority. For the purpose of practice, self-signed certificate is sufficient. In contrary, it is not suggested to use a self-signed certificate for system design implementation. Therefore, a third-party-signed certificate seems too much for a prototype of the TAMUCC food ordering service project implementation.

![SHA-1 Hash Algorithm implementation](image)

**Figure 2.5 SHA-1 Hash Algorithm implementation.**
Clear history order

The project not only allows users to submit their order but also allows users to delete their past order history. This feature is added as a button in each vendor’s main activity. When the button is pressed, it sends a delete order history request to the server. Then, the server checks where the request is coming from, by using `vendor` tag, `name` tag, and `student_id` tag. It then performs the deletion of the user’s past order history. When there is no history available, the vendor main activity recreates itself and displays an unsuccessful message as a toast window.

Payment method

Since the project cannot deliver a secure HTTPS protocol, it does not implement any banking transaction. However, in order to get the picture of how the payment method works, the project implemented a prototype of sanddollar as the only payment method. Likewise other payment method, it checks whether the user has sufficient balance before submitting the order. If the user has insufficient balance, then the application notifies the user to contact sanddollar office.

2.2.2 Database (MySQL), hypertext preprocessor (PHP), and web server (Apache)

Android application sends a request to the server through the network. The server receives and processes the request, and then it replies back to the Android application through the network, as shown in Figure 2.6.
Why PHP?

The reasons for using PHP in this project are [Web Design 2012]:

1. Free

   Unlike ASP.NET and JSP, PHP is free of cost. Since this project is a prototype food ordering application, so there is no need to use any commercial development platform.

2. Good portability

   PHP can be implemented on any operating system (Linux, Windows, and so on) and is also compatible with any standard Web server (such as, Apache Web server)

3. Simple grammar

   PHP has many similarities with the C programming languages, which makes easier for C programmers to use.

4. Wide extension

   Many databases and Web server extensions are provided by PHP to interact with other applications. Additionally, PHP allows developers to access the open source library for developing and providing more functionality.
5. Rapid development

PHP continues to develop rapidly because of its open source code.

Why MySQL?

The reasons for using MySQL in this project are [Web Design 2012]:

1. Account security

Every account of MySQL is composed of the users' name, password, and the position, which includes security check and reasonable rights. There are three different security checks in the MySQL: authorization, register confirmation, and access control. The reasonable rights demand means that there are different rights for different accounts and there are different rights for different data sheets or database as identical user.

2. Stored data encryption

MySQL implements encryption keys in 128-bit (MD5, AES, DES), and 160-bit(SHA-1); however, MySQL allows programmer to modify the source code and increase the key size. Consequently, the longer the chosen key, the longer encryption/decryption time takes place. In some cases of massive information application program, encryption of a small part of data, such as user password, would be as effective and efficient as in the case of a small information application program. However, these passwords should be stored in the data base as the encrypted form but not as the plain text form. In general, the sensitive data (including MySQL) is encrypted by Hash algorithm in most DBMS.
**PHP snippets**

On the web server, PHP handles replies and processes each request. It checks whether the request is set the `vendor` tag value and `vendor` tag value is not empty. Then, it assigns the incoming `vendor` tag from the user to PHP local variable for further processing. Next, the other processing code can be performed. When the processing is done, it encodes the result to an array and sends the array back in JSON Format, as shown in Figure 2.7.

```php
<?php
if (isset($_POST['vendor']) && $_POST['vendor'] != '') {
    //get vendor’s tag
    $vendor = $_POST['vendor'];

    $response = array("vendor" => $vendor, "success" => 0, "error" => 0, "subtotal" => 0);

    if ($result = mysql_query($myquery)) {
        if (mysql_num_rows($result) == NULL) {
            $response["error"] = 1;
            $response["error msg"] = mysql_error();
            echo json_encode($response);
        } else {
            // it performs some computation
            //

            $response["subtotal"] = $subtotal;
            $response["success"] = 1;
            echo json_encode($response);
        }
    }
}
?>
```

Figure 2.7 Processing and replying to a request.
Server Information

The web server is hosted on the free hosting web server, at http://www.podserver.info/. When we create a new account, the account comes with database (MySQL), hypertext preprocessor (PHP), and web server (Apache) installed. So, we do not need to be worry about the installation. This part holds the main source code. It accommodates the communication line from consumers to vendors. MySQL holds the database containing important users’ personal information, pricing items, and sales order history. Then, on PHP part, it provides information about orders that includes order in, order out, and order history. Finally, the Apache web server provides a powerful and secure connection for client-server communication. It acts as a secure intermediate communication channel between android application and MySQL database with the syntax communication exchange from PHP, such as $_POST and $_REQUEST variables.

The server has one database with several tables. The tables are:

1. users table: it contains users’ sensitive information.
2. users_sanddollar table: it contains users’ allowance balance.
3. order tables: there are six order tables. They contain order placed by the users for each of the food services located in various buildings.
4. price tables: there are six price tables for each of the food services located in various buildings.

The same server also consists of PHP code accordingly for several food services. They are:
1. android index: contains login and registration process from Android application.

2. vendor index: there are seven vendors index; one of them is sanddollar office and the rest of them are the food services located in various buildings across the campus.

3. checkout process: there are six separate code that handle the six vendors independently. Each of them inserts the entry to the order table database accordingly.

4. order history retrieved: contains retrieving order history from order tables on the MySQL database based on each vendor’s tag.

5. pricelist retrieved: contains retrieving several items’ price list from price tables on the MySQL database based on each vendor’s tag.

6. uc classics main parser: parses html elements on the main dining website.

The web server (Apache) is already installed at podserver and can be accessed at http://www.podserver.info.
**PHP-DOM as a parser**

Parser code snippets are provided in Appendix B. The parser has two modules:

1. The first module: parses http://tamuccislanddining.com/dining/index.html. It retrieves the current date and compares whether today’s date is in the range of dates in a week. Then, it grabs the corresponding link address. This link address is the input of the second module.

![Figure 2.8 PHP-DOM parser first module.](image-url)
2. The second module: grabs the input from the first module and parses the link address. Then, it retrieves current day and searches the corresponding menu.

Next, it returns all the menus to the Android application.

![Image of the second module]

Figure 2.9 PHP-DOM parser second module.

**PHP SESSION**

PHP SESSION is a build in feature of PHP that preserves certain data across subsequent accesses. The project also implements SESSION on each request, such as, login request, history and pricing retrieval, and submitting order. In a result, the login session is maintained for certain amount of time on every last access time. Once the time is reached, the user is enforced to re-login.

2.2.3 **Vendor (PHP website)**

This part shows all of the approved user orders. The approved order means the
user order has been successfully paid through the sand dollar system. Furthermore, each of the food services has their own user interface. After the vendor sees an approved order pop up on the screen, the vendor starts making the order. The food is expected to be ready in a certain amount of time and before the consumer comes to pick it up.

This project have seven total vendors’ website. They are:

1. sanddollar index: simulates depositing money to a sanddollar student account.
2. vendor index: there are six vendors index and each of them shows the placed order by the user.

Several screenshot regarding depositing money is provided in Chapter 3.

**PHP snippet**

The vendor’s end-user system is a web based user interface. Each vendor views and interacts with a corresponding PHP website. Since MySQL, PHP, and Apache reside and run on the same machine, the communication, computation, and data exchange occur internally on the server. It happens between PHP and MySQL. The snippets of vendor Starbucks in retrieving the incoming order from database, as shown in Figure 2.5.
<?php
    $query = "SELECT * FROM `starbucks_order_db` ORDER BY `uid` DESC";

    if ($result = mysql_query($query))
    {
        if (mysql_num_rows($result)==NULL)
        {
            echo 'No results found.';
        }
        else
        {
            echo '<table border="3">';
            while ($row = mysql_fetch_assoc($result))
            {
                /*
                // it echos the fetched array
                // in a formatted table
                */
            }
        }
    }
?>

Figure 2.10 Vendor’s retrieves data from database.
3. USER STUDY

As a user centered testing, user study helps to improve the application, such as the system design and user interface of the application. There are two groups of respondent. The first group is computer science students. The second group is non-computer science students. This user study aims at how users feel about the application and what further improvements in the application.

The user’s feedback focuses mainly at “single login single order” application design, it expects to have “single login multiple orders” application design. Therefore, a new button is added on the thank you Activity. This button leads the user back to the hallway Activity and allows the user to make another order. The rest of the feedback are “Confirm” button on checkout process, save username and password, and attractive in design and color.

The “Confirm” button is implemented. Unfortunately, the project cannot deliver several feedback. The project does not implement the “Save username and password” function as it interacts with the sanddollar payment system where user’s sensitive information is stored. The project leaves “attractive in design and color” feedback as future work.
4. TESTING AND RESULTS

This section uses a scenario to test the overall project. The test scenario flows accordingly:

1. Consumer end-user (Android application) module
2. Database (MySQL), hypertext preprocessor (PHP), and web server (Apache) module
3. Vendor end-user (PHP website) module

4.1 Consumer end-user (Android application) module

4.1.1 Registration process

A student, named John Doe, downloads the TAMUCC Food Ordering Application on his Android API 8 (Android Froyo 2.2) device. He runs the application and sees the first screen as shown in Figure 4.1(a). Then, he clicks “Go to the login screen” link and sees the login screen as shown in Figure 4.1(b). Since he does not have the account yet, he clicks “I do not have account. Register Me!”. He sees the registration screen as shown in Figure 4.1(c). He enters his information, student ID, full name, email, and password. Then, he presses “Register” button. When the registration successful, it displays a screen as shown in Figure 4.1(d). Contrary, the registration can be failed if the student ID is already stored in the database.
Figure 4.1 Scenario John Doe – registration process.

(a) John Doe runs the application and sees the dashboard activity.
(b) The application leads John Doe to the login activity.
(c) John Doe navigates to register activity.
(d) John Doe’s registration is successful.
4.1.2 Login and order placing process

After John Doe successfully registered, he clicks “Go to the login screen” that leads him to the login screen. He enters his student ID and password, then clicks the login button, as shown in Figure 4.2(a). The hallway activity, where he can select various food services location, comes to the screen, as shown in Figure 4.2(b). John Doe decides to order some food from “Center for Instruction Learning”, and then the Center for Instruction Learning Info screen shows up, as shown in Figure 4.2(c). He clicks “Click here to place an order …” and the Center for Instruction Learning Main screen appears. On top of the screen of the Center for Instruction Learning Main, it shows John Doe’s order history. As a new user, John Doe does not have any order history, as shown in Figure 4.2(d). After he is done with selecting the menus, he clicks “Click here to place an order . . .”. It leads to the checkout process, where John Doe can review his order before submitting it, as shown in Figure 4.3(a). He agrees with his review order, and then he clicks the “Checkout” button. On the successful order submission, the “Thank you” screen comes to the front that displays John Doe’s name, his total order, and the place where he can pick up his order, as shown in Figure 4.3(b). In addition, if John Doe places his order in the University Center, it also shows a notification as “The classics order will be charged separately”.
Figure 4.2 Scenario John Doe – login and placing order process (part 1).

(a) John Doe enters his A number and password.
(b) On successful login, the application proceeds to hallway activity.
(c) John Doe selects “The University Center” and sees the info activity.
(d) The application leads John Doe to the main activity where he can select any menu from “The University Center”.
Figure 4.3 Scenario John Doe – login and placing order process (part 2).

(a) The application leads John Doe to checkout activity and displays the order.
(b) John Doe submits the order and sees the thank you activity.

4.2 Database (MySQL), hypertext preprocessor (PHP), and web server (Apache) modules

Recall in the section 2.2.2, this module performs computation internally on the web server.

4.3 Vendor end-user (PHP website) module

Before John Doe places his order, he must have appropriate balance on his sanddollar account. This scenario describes John Doe does not have a sanddollar account. John Doe goes to sanddollar office and deposits $200.00, as shown in Figure 4.4. After a successful deposit, the system gives feedback, as shown in Figure 4.5.
Figure 4.4 Sanddollar admin registers John Doe on the database.

Figure 4.5 After a successful deposit, the system gives report to the sanddollar admin

After John Doe places his order on the Center for Instruction Learning order, the Center for Instruction Learning vendor sees an incoming order, as shown in Figure 4.6. The Center for Instruction Learning counter prepares and labels the order as ready by pressing the “Ready” button. When the “Ready” button is pressed, the button disappears,
as shown in Figure 4.7. John Doe picks up his order, the University Center – classics counter clicks “Gone” button, which indicates that the order has been picked up. When the “Gone” button is pressed, the button disappears, as shown in Figure 4.8.

![Figure 4.6 Ultimate Baja sees the incoming order.](image1.png)

![Figure 4.7 After the order is prepared, Ultimate Baja presses “Ready” button.](image2.png)
Figure 4.8 After the order is picked up, Ultimate Baja presses “Gone” button.

John Doe comes and picks up his order at Ultimate Baja that is located at Center for Instruction Learning building.
5. CONCLUSION AND FUTURE WORK

This project encounters two growing number of Texas A&M University – Corpus Christi population and of Android application market. TAMUCC website emphasized that about 11,900 students, faculty and staffs in total. Similarly, the Android application market reaches over 8 million and still growing. The main reason of Android application market growth is openness and open source; this is a welcome invitation for developers to come and build on learning and sharing experiences environment. Furthermore, developers are given a capability to utilize the resources of the mobile devices. Therefore, providing food services at TAMUCC is straightforward with the help of Android mobile devices.

We have provided a tool that simulates the food ordering through Android mobile device to the food services on TAMUCC. The system design is carried out as client-server design scheme with three different modules, such as, Android end-user application, web server, and vendor end-user website. The Android end-user application is fairly simple to use, from the registration, login, selection of food service location, selection of the available food menu, reviewing the order, and finally, submitting the order. Meanwhile, the web server maintains PHP scripts and MySQL database, such as, handling login and registration requests, handling order submission requests, handling order retrieval requests, maintaining the user information, maintaining the order history of each vendor, and hosting the vendor’s website. Then, the vendor end-user website allows the vendor to view the incoming order and to change the status of the order into “Ready” and “Gone”.

37
5.1 Limitations

To sum up, this prototype can be implemented only if:

- The company that provides the food services is willing to cooperate with TAMUCC in order to deliver an ease service.
- The company that provides the food services is willing to increase the number of employees for handling the incoming orders from both the food counter and the mobile application online.

5.2 Future Work

This project provides a complete functionality for client and server scheme in placing food orders; however, the project is far from perfect and is open to improvement. Some future work for this project is:

- Implementation of HTTPS protocol with SSL certificate.
- Improvement of graphical user interface design (“look-and-feel”) on both Android application end-user and website vendor end-user. For example, implement images for every menu, implement map view for every vendor’s location, and implement students’ picture for every placed order; the picture can be retrieved from sanddollar database.
- Converting sample data information into real data information by making sample data information available online.
- Implementation of pricing on real data information which affects on each vendor’s website.


Appendix A: Android application end-user

A.1 What the user will see

This appendix A shows screen shots of the running application. We show the login.xml, register.xml, hallway.xml, einstein_info.xml, subway_main.xml, checkout.xml, and thankyou.xml.

A.1.1 Dashboard xml

This screen comes in design for the user friendliness. On the first execution, it checks whether the user is still logged in. If the user is still in the login status, then this screen shows the user’s name and forces the user to logout. Figure A.1 shows the login status; the non-active login status is shown in the left figure and the active login status is shown in the right figure.

![Dashboard xml screenshot](image)

Figure A.1 Dashboard xml.
A.1.2 Login xml

This screen is the login screen and the starting point of the application. This view has two EditText fields where users can enter their student ID and password. It also has a link for a new user to register an account, as shown in Figure A.2.

![Login xml](image)

Figure A.2 Login xml.
A.1.3 Register xml

This page has four EditText fields where user can enter theirs student ID, full name, email, and password. It also has the link to the login page, as shown in Figure A.3.

Figure A.3 Register xml.
A.1.4 Hallway xml

This page is the main page where all of the food services are located. Each of the links lead to a brief description of the food service, as shown in Figure A.4.

Figure A.4 Hallway xml.
A.1.5 Info xml

This page shows a brief description of the Einstein Bros®. It also has a link to place an order, as shown in Figure A.5.

![Figure A.5 Info xml.](image-url)
A.1.6 The University Center main xml and Subway main xml

This page shows placing an order on Subway food service. At the top of the screen, it retrieves the user’s past order history. There are many options, such as, 6 or 12 inch sandwich, selection of bread, selection of flavor, selection of vegetables, selection of dressing, and selection of meal or just a sandwich. The options on this page are similar to when we are placing an order at Subway in person, as shown in Figure A.6.

![Figure A.6 The University Center main xml and Subway main xml.](image)
A.1.7 Checkout xml

This screen shows a user’s order review before the user confirms the order and makes a payment. It also shows the order subtotal, as shown in Figure A.7.

Figure A.7 Checkout xml.
A.1.8 Thank you xml

This screen shows a successful payment, subtotal, and location where food service is located, as shown in Figure A.8.

Figure A.8 Thank you xml.
Appendix B: Code Snippets

B.1 Android application code snippets

The code snippet demonstrates how Android makes a connection to a web server and retrieves a PHP SESSION on login request. After the login process, HttpPost maintains login session key on the header of every request, as shown in Figure B.1.

```java
public class JSONParser {

    // singleton httpcontext instance on every httpclient
    static private HttpContext httpcontext = null;
    static InputStream is = null;
    static JSONObject jObj = null;
    static String json = "";

    public static HttpContext getHttpcontext() {
        if(httpcontext == null){
            JSONParser.setHttpcontext(new BasicHttpContext());
        }
        return httpcontext;
    }

    public static void setHttpcontext(HttpContext hcontext) {
        JSONParser.httpcontext = hcontext;
    }

    public JSONObject getJSONFromUrl(String url, List<NameValuePair> params) {
        // Making HTTP request
        try {
            // Create a local instance of cookie store
            CookieStore cookieStore = new BasicCookieStore();

            // HttpContext
            HttpContext hcontext = JSONParser.getHttpcontext();
            hcontext.setAttribute(ClientContext.COOKIE_STORE, cookieStore);

            // defaultHttpClient
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new HttpPost(url);
            httpPost.setEntity(new UrlEncodedFormEntity(params));
            //httpPost.setHeader("Cookie", "PHPSESSID=" + PHPSESSID +"; path=/");
        }
    }
}
```
Figure B.1 Singleton HTTPContext object and cookies header on every HTTPPost
B.2 PHP snippets

The PHP-DOM parser has two modules; module 1 parses the web page http://tamuccislanddining.com/dining/index.html and outputs a specific web page. The code snippet is PHP-DOM module 1, as shown in Figure B.2.

```php
/*module 1 - main page parser*/
function getTags($dom, $uppermaintagName, $uppermainattrName, $uppermainattrValue, $midmaintagName, $midmainattrName, $midmainattrValue, $lowmaintagName, $newdate ){
  $html = '';   
  $domxpath = new DOMXPath($dom);  
  $newDom = new DOMDocument;  
  $newDom->formatOutput = true;  
  $filtered = $domxpath->query("//$uppermaintagName" . '[@' . $uppermainattrName . "='" . $uppermainattrValue . "]" . "//" . $midmaintagName . "=" . $midmainattrValue . "'/" . $lowmaintagName);  
  $menunode = $newDom->createElement('menu');  
  $menunode->normalize();  
  $i = 0;  
  while ($myItem = $filtered->item($i++)){  
    $node = $newDom->importNode($myItem, true); // import node  
    $node->normalize();  
    $itemnode = $newDom->createElement('item'); // create a new node  
    $itemnode->normalize();  
    $itemnode->appendChild($node); // append imported node to new node  
    $menunode->appendChild($itemnode); // append new node to menu node  
    $menunode->normalize();  
    $newDom->appendChild($menunode);  // append menu node to DOMXPath  
  }  
  $newDom->normalize();  
  $params = $newDom->getElementsByTagName('a');  
  foreach ($params as $param) {  
    $newparam = $param->nodeValue;  
    $tiny = explode(" ", $newparam);  
    $tiny[5] = str_replace("", "", $tiny[5]);  
    $newa = (int)$tiny[3];  
    $newb = (int)$tiny[5];  
    if ($newa <= $newdate && $newdate <= $newb) {  
      $par = $param->getAttribute('href');  
      $newpar = 'http://tamuccislanddining.com/dining/' . $par;  
    }  
  }  
  return $newpar;  
}
```

Figure B.2 PHP-DOM module 1.
The code snippet is PHP-DOM module 2. It receives module 1 output, parses the specific web page, and retrieves the menu, as shown in Figure B.3.

```php
/*module 2 - specific page parser*/
function getSpecTags($spechtml, $uptagName, $upattrName, $upattrValue, $midtagName, $midattrName, $midattrValue, $lowtagName)
{
    $specdom = new DOMDocument;
    $specdom->preserveWhiteSpace = false;
    @$specdom->loadHTMLFile($spechtml);

    $html = '';
    $domxpath = new DOMXPath($specdom);
    $newSpecDom = new DOMDocument;
    $newSpecDom->formatOutput = true;

    $filtered = $domxpath->query("//$uptagName" . '@' . $upattrName . '=' . "$upattrValue" . "//$midtagName" . '@' . $midattrName . '=' . "$midattrValue" . "//$lowtagName");

    $menunode = $newSpecDom->createElement('menu');
    $menunode->normalize();

    $i = 0;
    while( $myItem = $filtered->item($i++) ){
        $node = $newSpecDom->importNode( $myItem, true ); // import node
        $node->normalize();
        $itemnode = $newSpecDom->createElement('item'); // create new node
        $itemnode->normalize();
        $itemnode->appendChild($node); // append imported node to new node
        $menunode->appendChild($itemnode); // append new node to menu node
        $menunode->normalize();
        $newSpecDom->appendChild($menunode); // append menu node to DOMXPath
    }
    $newSpecDom->normalize();

    $html = $newSpecDom->saveXML();
    return $html;
}
```

Figure B.3 PHP-DOM module 2.