WIRELESS FINANCIAL TRANSACTION REPORTING SYSTEM FOR M-COMMERCE

Jongwook Woo, California State University, jwoo5@calstatela.edu

ABSTRACT

As mobile devices have been popular, many wireless applications for mobile devices have been developed. The paper presents the financial transaction system for online game and its architecture in J2EE. The system provides transactional report by filtering the millions of transactions on selected periods. The paper first describes the system's financial transaction formula, then, provides m-Commerce architecture, and finally illustrates how the system is expanded to wireless application on WAP by integrating J2EE and WML. The financial analyzer can gather information by the mobile device and analyze the transaction of the game with the reports displayed on the device that is portable and scalable.

Keywords: WAP, m-Commerce, e-Business, WML, J2EE

INTRODUCTION

e-Business system that includes *e-Commerce* has been popular. *IBM* defines *e-Business* as the leveraging of network capabilities and technologies in order to achieve and maintain the huge advantages for customers, suppliers, partners, and employees [9]. *e-Business* activities can be classified into three categories based on end-users of transactions: Intra-business, Business-to-consumer, and Business-to-business. Intra-business activity is to share company information and computing resources among employees on the intranet: for example, knowledge management. Business-to-business activity is to improve inter-organizational partnerships and relationships: for example, supply chain integration. Business-to-consumer, the most common activity, is to provide services to consumers who are out of organizations: for example, customer resource management (*CRM*), *e-Commerce*, and web auctions etc [6]. And, financial transaction report system that is introduced in this paper is the example of *CRM*. Accountant needs to analyze the gain and loss of a product and then to see the report for net cash and net earned revenue that are computed based on the customers' subscription data accessible through the intranet.

n-tier architecture for *e-Business* system has been presented because businesses have to improve efficiency by integrating data and applications across the enterprise. Besides, the highest levels of performance and availability must be maintained for the critical businesses. In order to enable high performance, scalability, and availability to businesses, it is to partition systems and software to more flexible blocks that have their roles [1]. Section 4 of this paper introduces *n-tier* architecture in detail.

As mobile devices and wireless telecommunications have grown these days, *m-Commerce* industry has been popular. *m-Commerce* can be defined as *e-Business* with mobile device. *m-Commerce* is the same as *e-Business* in its fundamental concept and actually it is extended from *e-Business* architecture. But, it needs wireless environment to connect mobile device to the legacy system and to develop its client logic. In this paper, the client logic is built in *WML* that is the extension of *XML* (eXtensible Markup Language). And, the *WAP* gateway is used for *WAP* communication between the legacy system in *J2EE* (*Java 2 Enterprise Edition*) for business and data access logic and *WML* client logic [11].

In this paper, section 2 is Related Work. Section 3 Financial Transaction Report describes what transactional report is and why it is needed, especially for the online game. Section 4 *n-Tier* Architecture on *e-business* and *m-Commerce* describes *n-Tier* architecture on *e-Business* and *m-Commerce*. Section 5 explains the architecture of our system. Section 6 presents experimental result. Finally, section 7 is Conclusion.

RELATED WORK

Kris *et al*. introduce the wireless protocol and languages at the moment. The Wireless Application Protocol WAP with XHTML and WML is illustrated. Besides, they describe the properties of i-mode mobile service with cHTML and J2ME (Java 2 Micro Edition) as tools for mobile application [13]. The paper just introduces the current techniques and trends in wireless applications.

Wang *et al.* describe WAP and i-Mode. Then, they provide the sample *m-Commerce* applications

simulated on Nokia Toolkit 4.0 [14]. The Wang's paper has the similar architecture to our paper's in terms of WAP. However, the *m-Commerce* architecture in our paper is implemented with WML that is integrated with the dynamic web contents in J2EE back-end system. It is even connected to DB of the application. Besides, our application is to access and display the financial transaction report.

Raymond *et al.* present their clinical data access system as a wireless healthcare application. It uses Palm Proxy server as a WAP server and PDA (Personal Digital Assistant) as a WAP browser. It uses existing back-end system built in Microsoft ASP language on Microsoft IIS web server [15].

Eneider *et al.* provide the health care mobile system that is executed on web server with Palm PDA. It is to provide health care information by reducing the errors [16]. Their back-end system is CGI that has less performance than our systems in J2EE. And, Raymond and Eneider's applications are wireless healthcare systems.

FINANCIAL TRANSACTION REPORT

For business, it is important to display and analyze the gain and loss of a project. Many companies analyze their business loss or gain with the factors such as net cash and earned revenue. This section describes the basics of cash application and revenue and what are their mathematical formulae for the online game project. The online game supported by the financial system in our project has the millions of transaction with the hundreds of thousands of customers. Customer in the online game needs to subscribe it by determining its billing cycle as monthly, quarterly, semi-annually, and annually.

The financial system built on computer can provide well organized information to an accountant for the millions of transactions. If there is no such a system, it will be the nightmare for the accountant. The accountant used to use the simple application such as *Microsoft Excel* for the financial system. However, the excel file cannot handle the complicated data as the online game transaction system. Therefore, our transactional system is implemented in J2EE to compute complicated data on web for our system. The accountant needs to analyze customers' net cashes and earned revenue for a period by collecting the transactional data such as subscription amount and tax etc. The transactional system can display the transactional data and its net cashes and earned revenue for a period and it has two reports as follows.

First, there is cash application report to calculate a net cash of each customer and its summary. Since there are millions of transactions in the game, the accountant needs data in certain period. Cash application report needs each game account's subscription amount, credit card fee charged, online payment fee charged, and tax charged in order to calculate its net cash. The formula to calculate the net cash is as follows:

Net Cash = (Subscription Amount) - Tax – (Credit Card Fees) – (Online Payment Fees) (Formula 2.1)

For each customer, since the online game system stores its subscription amount and tax and fees charged, its net cash can be calculated by formula 2.1. However, the user – accountant - needs to filter the millions of customers by selecting periods as start and end dates and collect only small amounts of customers. Second, there is revenue report to calculate earned revenue for each customer by its billing cycle in a period selected by the accountant. The report needs each game account's gross earned revenue, credit card fee charged, online payment fee charged, and tax charged in order to calculate its net earned revenue. Its formula is similar to cash report formula 2.1. However, it needs to calculate daily revenue rate before calculating earned gross and revenue. The daily revenue rate is calculated as the total subscription rate divided by the number of days in the month(s) spanning the subscription as follows:

Daily Revenue Rate = (subscription amount) / (total days) (Formula 2.2)

For example, there is a customer who subscribes the game with $14 subscription amount from January 15 2007 for a monthly subscription. His account is expired on February 14 and he has 31 total days for the subscription. Thus, its daily revenue rate is $0.4516 (=14/31). Revenue will be earned on a daily basis. The rate of revenue earned will be based on the number of days in the month(s) spanning the subscription. Based on the daily revenue rate, earned revenue as of a date can be computed as follows:

Gross Revenue Earned As of Date D = (D – (subscription date)) X (Daily Revenue Rate) (Formula 2.3)

For example, for the revenue earned as of January 31 2007 of the above example, total days between January 15 and January 31 are 17 days. Thus, its gross earned revenue is $7.6772 (= 17 X 0.4516). Let's see another example, for an annually subscription that begins on Feb 15 2007 with the subscription amount $150, the user want to see the revenue earned as of April 14th. First, the daily rate will be calculated as follows:

Daily Revenue Rate = 150 / 365 = $0.4110

Total days between February 15 and April 14 are 59 days (14 days for February, 31 days for March, and 14 days for April).

Its Gross Revenue Earned as of April 14, 2007 is $24.249 (=0.4110 X 59).

We can compute Net Earned Revenue with Formula 2.3 and other fees as follows:

Net Earned Revenue = (Gross Earned Revenue of (Formula 2.3)) - Tax – (Credit Card Fees) – (Online Payment Fees) (Formula 2.4)

Several classes such as *FinancialMath*, *CashReport*, and *RevenueReport* are implemented for formula 2.1-2.4 in *financialReport* package of the financial transaction system as the part of the online game system. Besides, those classes have *JDBC* statements in order to retrieve data from Database server for account name, subscription amount, tax, credit card fees, and online payment fees etc.

N-TIER ARCHITECTURE ON E-BUSINESS AND M-COMMERCE

The traditional *Client-Server* architecture has a mainframe that includes core applications and data. The mainframe is accessed from thick clients that are big applications. We can call it *2-tier* architecture that has many loads between client and server because of their tight interoperations for its presentation logic, business logic, and data access logic. This tight interoperation has generated many issues in the current high volume business systems. It is not scalable because it should replace the entire system when its capacity is exceeded. And, it is not flexible because its presentation logic, business logic, and data access logic are tightly coupled. If the developer wants to modify its business logic, he/she should modify the entire logics. Besides, the developer must adapt or modify the business logic when it is integrated with the World-Wide-Web or other applications [1].

The *n-tier* architecture has addressed the issues of the *2-tier* architecture and become the solution of the current e-Business systems on Internet and World-Wide-Web. It partitions application functionalities into *N* independent layers, mainly three layers as in Figure 3.1. Thus, it becomes easier to integrate with the existing business systems. The layer 1 is the presentation logic that is typically hosted on Web server with web browser. The presentation logic is to send the request of client and receive its response from business logic. The response is normally dynamic or static web pages formatted to present the client. The layer 2 is hosted on mid-tier (middleware) server as business logic. It includes the business functions that are the main of the e-Business applications on *n-tier* architecture. It produces the response of the request from the client and provides it to the client. If the request is related to access data, it will pass the data access request to the back-end database server. The layer 3 is hosted on the back-end database server as database access logics. It is to handle the request of data source from the business logic. It has the functions to access data source, that is, database. Since business logic is separated from presentation logic and database access logic physically, each layer can be scalable and upgradeable independently. And, even if a layer is modified or replaced, the application of other layers does not need to be recreated. Besides, each layer can be implemented with clustered servers for its logic. The clustering enables high-performance computing, availability, and scalability [1]. Therefore, the current e-Business systems are implemented on *n-tier* architecture.

m-Commerce has the same architecture of Figure 3.1. In *m-Commerce*, presentation logic is built in WML. Client device can be mobile phone or palm pilot. Thus, middleware server is composed of *WAP* gateway server, web server, and application server. *WAP* gateway server is to convert *WAP* data to http compatible data (or the other way). When a mobile device sends a request to the *WAP* application running on the application server by selecting the system's *WAP* address as locating a *WML* file, the request is first routed through the *WAP* Gateway where it is decoded, translated to *HTTP*, and then forwarded to the appropriate *URL*. After executing the business logic in Java class referred by *WML* codes, its execution result will be generated. The execution result of the response is then re-routed back through the gateway, translated to *WAP*, encoded, and forward to the mobile client. Thus, the mobile device can display the data result responded from the *WAP* address. This proxy architecture allows application developers to build services that are network and terminal independent [11].

<<insert Figure 3.1 here >>

THE ARCHITECTURE OF THE FIANCIAL TRANSACTION REPORTING SYSTEM ON WAP

The reporting system is built in *J2EE* and *WML* on *WAP* gateway, application server, and Database server. The *WAP* Gateway acts as the bridge between the mobile network containing mobile clients and the computer network containing application servers as shown in Figure 3.1. The financial report system uses *Nokia WAP Gateway Simulator 4.0* as *WAP* gateway server. The system is mainly developed in J2EE version 1.5 on *Web Sphere version 5* application server. Its database server is *Oracle 9i* server.

WebSphere server executes business logic and data access logic in *J2EE*. The financial report logic described in section 2 is built in Java and packaged to *edu.calstatela.hipic.financialReport*. The business logic is composed of the utility classes to calculate

the net cash and net earned revenue by Formulae 2.1-2.4 based on the billing type of a customer for a period entered by the user.

For the system, the numbers of tables are created to store customer's payment information on *Oracle 9i* database server. Table 4.1 to 4.3 presents the fields of tables such as USER_INFO, TRX, and BILLING_CYCLE.

<< insert Table 4.1 to 4.3 here>>

USER_INFO table has fields for user information and a unique key (BILLING_ID+ACCOUNT_NAME). Table TRX is to represents information for each transaction with its unique key TRX_ID and it is related to USER_INFO table with BILLING_ID foreign key. Table BILLING_CYCLE is to present billing cycle information such as Monthly, Quarterly, Semiannually, and Annually. It is related to TRX table with the foreign key PLAN_ID.

The system has two main *JavaBean* classes for Data Access logic such as *CashApplicationReport* and *RevenueReport* with several supporting classes. These *JavaBean* classes have data access logics to connect DB server and to join several tables to generate the properties that are used for each report. For example, *CashApplicationReport* is composed of *JDBC* statements to access the DB and to join USER_INFO and TRX tables. Thus, the properties of *CashApplicationReport* class are *trx_id*, *accountName*, *billStartDate*, *gross*, *creditcardFee*, *onlinePaymentFee*, *tax*, *settlementStatus*, and *netCash* with *set* and *get* methods of them. Among them, *netCash* property is calculated with other properties' values by Formulae 2.1 as shown in section 3. The SQL to join the tables is shown in Figure 4.2:

<<insert Figure 4.2 here >>

The properties of *RevenueReport* class are *grossEarnedRevenue*, *creditcardFee*, *onlinePaymentFee*, *tax*, and *netEarnedRevenue* with *set* and *get* methods of them. *grossEarnedRevenue* and *netEarnedRevenue* are computed by Formulae 2.2-2.4 in Section 3. The SQL command to retrieve the properties of *RevenueReport* class is shown in Figure 4.3:

<<insert Figure 4.3 here>>

These SQLs are built in each Java class as *PreparedStatement* of *JDBC* library for high-performance computing while joining TRX, BILLING_CYCLE, and USER_INFO tables. Question marks (*?*) of the SQLs are to get input from the user during run-time as precompiled. If we just join the three tables, the time complexity will be $O(u \times x \times b)$ when u is the maximum rows of USER_INFO, x is the maximum rows of TRX, and b is the maximum rows of BILLING_CYCLE table. Among these rows, the row size x is millions of transactions for billing and must be biggest to negatively affect the performance. Therefore, the SQL commands are improved in Figure 4.2 and 4.3 with subqueries by selecting ULT virtual table. Thus, the time complexity becomes $O(x + u \times xs \times b)$ when u is the maximum rows of USER_INFO, x is the maximum rows of TRX, xs is the maximum rows of ULT, and b is the maximum rows of BILLING_CYCLE table. Since $xs << x$ as tens of transaction rows versus millions of transaction rows, the performance has been improved dramatically.

The client logic of the system is mainly built in *WML* (Wireless Markup Language) such as *cash.wml* and *revenue.wml*. As shown in Figure 4.4, *cash.wml* file takes the input parameters by the user and passes it to jsp file *cashReportWML.jsp*. The jsp file is developed to access *Java* classes of the data access and business logics. Thus, it retrieves data by joining tables with SQL command of Figure 4.2 and generates a *JavaBean* class that contains data retrieved. And, it generate *WML* file that contains data to represent Cash report as in Figure 4.5. Thus, the mobile device can display the data result responded from the system's *WAP* address.

In order to represent Revenue report, there are *revenue.wml* and *revenueReportWML.jsp* files that are similar to Figure 4.4 and 4.5. The data displayed on the mobile device by these files are billing cycle, gross revenue, online payment fee, credit card fee, tax, and net earned revenue.

<<insert Figure 4.4 – 4.5 here >>

EXPERIMENTAL RESULTS

The financial report system is implemented with servers like *IBM WebSphere 5* application server, *Nokia WAP Gateway Simulator 4.0*, and *Oracle 9i Database* server. The mobile application is developed with *Nokia Mobile Browser4.0 SDK*. Figure 5.1 and 5.2 shows one of the experimental results of the Cash and Revenue reports respectively in the *Developer Platform SDK*. Depending on the input dates and billing cycle, the report should have different results.

The experiment is not to evaluate the efficiency of the system because the system depends on the existing back-end application in *J2EE*, *WebSphere* Application Server, and Nokia Active *WAP* server. The experimental result is to show that transactional

and financial system can be expanded to mobile environment and implemented and actually used in wireless environment for cellular phone users.

The user enters the start date and end date to retrieve data for the cash report as in Figure 5.1 (a). Then, after submitting the dates, its cash report of customers between those dates is displayed on the LCD panel of the mobile deveice, which is composed of each customer's Bill Start Date, Transaction ID, Account Name, Gross Amount, Credit Card Fee, Online Fee, Tax, and Net Cash calculated as in Figure 5.1 (b).

<<insert Figure 5.1 here >>

For the revenue report as in Figure 5.2 (a), the user enters the start date and end date and then selects its billing cycle to retrieve data. Then, after submitting the dates and its cycle, its revenue report of customers between those dates by the cycle is displayed on the LCD panel of the device, which is composed of each customer's Bill Cycle, Account Name, Gross Earned Amount calculated, Credit Card Fee, Online Fee, Tax, and Net Earned Revenue calculated as in Figure 5.2 (b).

<<insert Figure 5.2 here >>

CONCLUSIONS

e-Business has been the latest approach for business. Besides, as wireless communication has grown up *m-Commerce* becomes popular. Accountant needs financial transaction reports to analyze the loss and gain of a project. For our online game, its financial report system has been implemented on *n-tier* architecture for its *m-Commerce* system with *WAP*, which is extended from the legacy financial report system. The paper also presents its architecture and how it is implemented.

REFERENCES

1. "Building a Better e-Business Infrastructure: N-tier Architecture Improves Scalability, Availability and Ease of Integration", Intel® e-Business Center White Paper, http://www.intel.com/eBusiness/pdf/busstrat/industry/wp012302.pdf, 2001
2. "Technology Overview for .NET Framework 1.1"**,** Microsoft Corporation, http://msdn.microsoft.com/netframework/productinfo/overview/default.asp
3. "Got Dot NET (.NET Framework Website)", http://www.gotdotnet.com/team/lang/, Microsoft Corporation, 2003
4. "C#/JScript/CLI Implementations Shared Source Licensing Program", http://www.microsoft.com/resources/sharedsource/Licensing/CSharp_JScript_CLI.mspx, Microsoft Corporation, June 2003
5. "The Petstore Revisited: J2EE vs .NET Application Server Performance Benchmark", http://www.middleware-company.com/j2eedotnetbench, Middleware Company, Oct. 2002.
6. "Design Considerations: From Client/Server Applications to e-business Applications", Indran Naick, Luca Amato, Jason K O'Brien, Jim Nicolson, Tsutomu Oya, http://www.redbooks.ibm.com/redbooks/SG245503.html, Dec 1999
7. Nokia Mobile Browser Development SDK 4.0, http://www.forum.nokia.com, Nokia 2002
8. Nokia WAP Gateway simulator 4.0, http://www.forum.nokia.com, Nokia 2002
9. "IBM e-business Technology, Solution, and Design Overview", Brian R. Smith, Charles Ackeifi, Thomas G. Bradford, Prabhakar Gopalan, Jennifer Maynard, Abdulamir Mryhij, http://www.redbooks.ibm.com/redbooks/SG246248.html, August 2003
10. **"**Java 2 Platform, Enterprise Edition (J2EE)", http://java.sun.com/j2ee/, Sun Microsystems, Inc, 2007
11. WAP Forum, http://www.wapforum.org/, Open Mobile Alliance, 2007
12. IBM WebSphere, http://www-306.ibm.com/software/websphere/, IBM, 2007
13. "Developing mobile wireless applications**",** Internet Computing, IEEE, K Read, F Maurer, Jan/Feb 2003, Volume: 7, Issue: 1, pp81- 86
14. "Design and Evaluation of M-Commerce Applications", Wang, J.J. Song, Z. Lei, P. Sheriff, R.E**,** Communications, 2005 Asia-Pacific Conference**,** 03-05 Oct. 2005, pp 745-749
15. "Secure remote access to a clinical data repository using a wireless personal digital assistant (PDA)", RG Duncan, MM Shabot, Proceedings in AMIA (American Medical Informatics Association) Symposium, 2000
16. "Approach to mobile information and communication for health care", EA Mendonca, ES Chen, PD Stetson, LK McKnight, International Journal of Medical Information, Aug 2004; 3(7-8): pp631-63

TABLES AND FIGURES

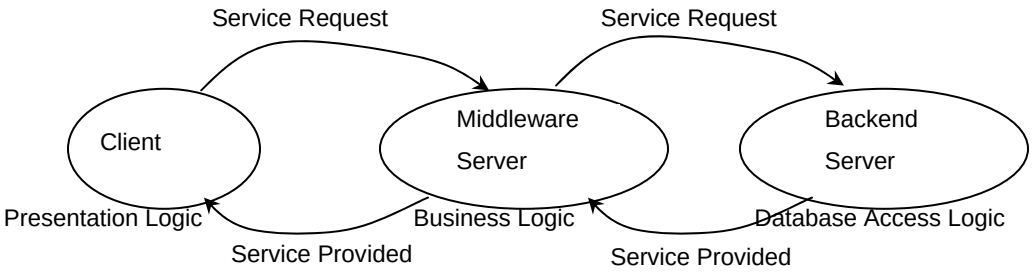


Figure 3.1. *N-tier* Architecture

Table 4.1. USER_INFO

BILLING_ID	ACCOUNT_NAME

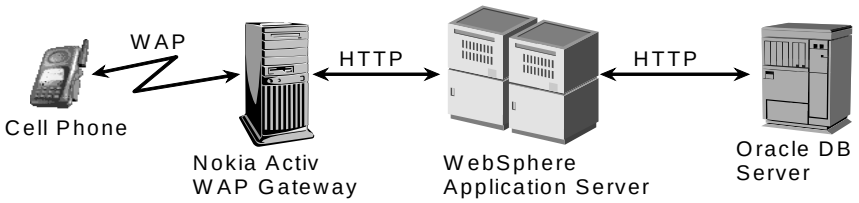
Table 4.2. TRX

TRX_ID	BILLING_ID	PLAN_ID	SETTELMENT_STATUS	SETTLED_DATE
GROSS	BILL_START_DATE	TAX	ONLINE_PAYMENT_FEE	CREDIT_CARD_FEE
BILL_END_DATE				

Table 4.3. BILLING_CYCLE

PLAN_ID	CYCLE

Figure 4.1. Report System WAP application Architecture




```
select unique TRX_ID, BILL_START_DATE, ACCOUNT_NAME, GROSS, CREDIT_CARD_FEE,
      ONLINE_PAYMENT _FEE, TAX, SETTLEMENT_STATUS
      from USER_INFO,
            (select unique TRX_ID, BILL_START_DATE, GROSS, CREDIT_CARD_FEE,
                  ONLINE_PAYMENT _FEE, TAX, SETTLEMENT_STATUS, PLAN_ID, BILLING_ID
              from TRX
              where SETTLED_DATE >= ? and (SETTLED _DATE - 1)< ?  and
                        SETTLEMENT_STATUS='ST' ) ULT,
             BILLING_CYCLE
      where TRX.PLAN_ID = BILLING_CYCLE.PLAN_ID and
                  USER_INFO.BILLING _ID = ULT.BILLING _ID
```

Figure 4.2 Cash report SQL

```
select unique TRX_ID, BILLING_CYCLE, account_Name, GROSS, ONLINE _PAYMENT_FEE,
      CREDIT_CARD_FEE, DISCOUNT, TAX,
    from USER_INFO,
        (select unique TRX_ID, GROSS, ONLINE_PAYMENT _FEE,
                CREDIT_CARD_FEE, TAX , BILLING_ID, PLAN_ID
                from TRX where SETTLE_DATE >= ? and (SETTLE_DATE-1) < ?  and ");
                        ((BILLING_END_DATE >= ? and (BILLING_END_DATE-1) < ?) or
                        ((BILLING_START_DATE-1) < ? and BILLING_END_DATE >= ?))  and
                        SETTLEMENT_STATUS='ST') ULT,
              BILLING_CYCLE
      where TRX.PLAN_ID = BILLING_CYCLE.PLAN_ID and
                USER_INFO.BILLING_ID = ULT.BILLING _ID
```

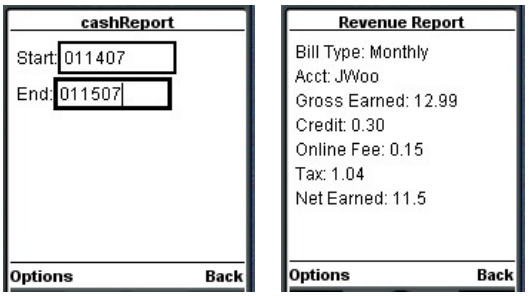
Figure 4.3 Revenue Report SQL

```
<%@ page language="java" contentType="text/vnd.wap.wml" %>
 <?xml version="1.0"?>
 <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"  "http://www.wapforum.org/DTD/wml_1.1.xml">

 <wml>
        <card id="cashReport">
                <do type="accept">
                        <go href="cashReportWML.jsp" method="post">
                                <postfield name="sDate" value="$(sDate)"/>
                                <postfield name="eDate" value="$(eDate)"/>

                        </go>
                </do>
       <fieldset title="Report Dates">
            <p>Start:
                <input type="text" name="sDate" maxlength="6" format="*N"/> <br />
              End:
                <input type="text" name="eDate" maxlength="6" format="*N"/>
          </p>
        <fieldset />
        </card>
</wml>
```

Figure 4.4 cash.wml

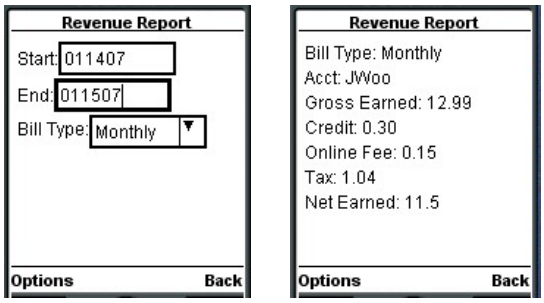
```
<%@ page language="java" contentType="text/vnd.wap.wml" %>
// jsp codes to get the parameters for start date sDate and end date eDate
// And to retrieve data from the DB by the parameters and create ResultSet rs for cash report

 <?xml version="1.0"?>
 <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"  "http://www.wapforum.org/DTD/wml_1.1.xml">

 <wml>
        <card id="cashReport">
                <% while(rs!=null && rs.next()) { %>
                    <p>Date: <%= rs.getBillStartDate() %></p>
                    <p>TRX: <%= rs.getTrxID() %></p>
                    <p>Acct: <%= rs.getAccountName() %></p>
                    <p>Gross: <%= rs.getGross() %></p>
                    <p>Credit: <%= rs.getCreditCardFee() %></p>
                    <p>Online Fee: <%= rs.getOnlinePaymentFee %></p>
                    <p>Tax: <%= rs.getTax() %></p>
                    <p>Net Cash: <%= rs.NetCash() %></p>
                <% } %>
        </card>
 </wml>
```

Figure 4.5. cashReportWML.jsp



(a) Cash Report Input (b) Cash Report

Figure 5.1 Cash Report



(a) Revenue Report Input (b) Revenue Report

Figure 5.2 Revenue Report