Babatunde Dunmoye

Android Mobile Application Development for an Online Bookstore with PayPal Integration

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

30.04.2015

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Babatunde Dunmoye Android Mobile Application Development for an Online Bookstore with PayPal Integration 43 pages 30 April 2015
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor	Olli Hämäläinen, Senior Lecturer

Due to the increasing saturation of the mobile technology, fuelled by its inherent properties such as flexibility, ease of use, and ubiquity, mobile e-commerce has gained significant business reputation promising great productivity, high profitability and an immense level of security.

The goal of this project was to design and develop an Android online bookstore. Focusing on business-to-consumer markets, the customers of a book sales' company could purchase books conveniently and pay via the integrated PayPal service using a mobile phone. The application allows a user to register an account, login, search for particular books of interest, sort books in ascending or descending order of price and purchase book(s) in the cart with a PayPal account.

The goal of the project was achieved by observing software development procedures and principles for software designs and implementation. In achieving the goal of this project, three major parts were designed and implemented. Firstly, the design of the UI (User Interface) was implemented by following the Android design guidelines for Android devices. Secondly, a MySQL database that connects and communicates with the web server through the Internet was designed to store the book data. Thirdly, the design of the Android phone local SQLite database was realised. This allows users to use the application offline.

The result of the project was a complete Android mobile application that is targeted at delivering a solution for online shopping with the PayPal integration. The project eliminates the need to drive to a retail store outlet, find and pay for a parking place. In addition, considerable time is saved as consumers do not need to walk throughout the store in search for an item to buy.

Keywords	Android mobile application, online bookstore, PayPal integration

Contents

Abbreviations and Terms 1

1. Introduction 2

2. Project Description 3

3. Android Phone Application Platform 4

    3.1. Android Development Framework 4

    3.2. Android Software Stack 5

    3.3. Android Phone Application Architecture 6

    3.4. Android Phone Application Development 7

4. Requirement Analysis 9

    4.1. Functional Requirements 9

        4.1.1. Input Requirements 9

        4.1.2. Operation Requirements 10

        4.1.3. Output Requirements 10

    4.2. Non-Functional Requirements 10

        4.2.1. Software Requirements 11

        4.2.2. External Interface or Hardware Requirements 11

        4.2.3. Secondary Requirements 12

    4.3. Algorithm Flow Chart 12

    4.4. Use Cases 14

5. The Online Bookstore Graphical Design 21

    5.1. Design View of Start-up Window 23

    5.2. Application Control Buttons 23

6. Implementation of the Online Bookstore 29

    6.1. User Registration 29

    6.2. User Login 30

    6.3. Parsing HTTP Response to JSON Object 31

7. Database Design 33

8. PayPal Integration 38

    8.1. PayPal Sandbox Overview 38

    8.2. Accessing The PayPal Sandbox 38

    8.3. Adding Funding Sources 39

9. Submitting the Application to the Android Market Place 41

10. Conclusion 43

References

Abbreviations and Terms

ADT	Android Development Tools
API	Application Programming Interface
AVD	Android Virtual Device
CPU	Central Processing Unit
DVM	Dalvik Virtual Machine
EDGE	Enhanced Data Rates for Global Evolution
EV-DO	Evolution Data Optimized
GPS	Global Positioning System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HSPA	High Speed Packet Access
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LIBC	Standard C Library
OpenGL ES	Open Graphics Library for Embedded System
OS	Operating System
PHP	Hypertext Preprocessor
SSL	Secure Sockets Layer
SDK	Software Development Kit
SMS	Short Message Service
TCP/IP	Transmission Control Protocol/Internet Protocol
UI	User Interface
XML	Extensible Mark-up Language

1 Introduction

In recent times, the advancement in the wireless technology and the growth in market potentials have led to an increase in the number of mobile device users. The emergence of this technology has given rise to rapid development of mobile e-commerce technologies. This brings on-the-go Internet access to the general online market world without geographical and time constraints.

This project aims to demonstrate how to design and develop an Android online bookstore application that connects and communicates with a MySQL database server through the Internet using Hypertext Preprocessor (PHP) and Java Script Object Notation (JSON) format. The data are primarily stored on the server but are also loaded and stored on the mobile phone's SQLite database in the event of lack of Internet access.

To make online shopping more interactive and user-friendly, the application allows users to register an account, login, search for particular books of interest, sort books in ascending or descending order of price and buy books in the cart with a PayPal account. Purchased books are delivered to the user's delivery address via a postal agency.

Furthermore, to create a smooth and great user experience, the application does not only check if the user already exists but can add new information for a new user or update new sales on the MySQL database. The application protects the user's password with an SHA256 encryption. This project also provides a way to recommend the application to friends via Short Message Service (SMS) or email.

2 Project Description

In this project, the design and development of an Android application for an online bookstore with the integration of a PayPal payment option was carried out. The application provides a smooth shopping experience for users, while offering an interactive way of paying for products in the shopping cart.

With this application, registered users with an account can login, search for books and promotion sales, search for books in descending or ascending order of price, view descriptions of books and buy books with PayPal. Unregistered users can also have access to the store, search for books and promotion but cannot buy books unless they register.

The application communicates with a MySQL server through PHP/JSON. From the MySQL server, the application loads 10 rows of information about the books at a given time. With a further scroll on the application touch screen, another 10 rows of information are yet loaded. This continues until the end of the query is reached. The PHP scripts do the connection of the Android device to the MySQL server.

Since the Internet plays a very vital role in this application, this application is made in a way to load all data from the external server (MySQL) into the device's native database (SQLite) whenever there is a connection to the Internet. This improves users' experience because, when there is no Internet connection users can rely on the already loaded data.

The administrator can add new information to the MySQL database. For example, new users and new sales can be added. The application also checks if the user already exists to avoid duplicate users.

3 Android Phone Application Platform

The important features about the use of Android as a development environment are centred around the Application Programming Interface (API) it provides. Android is an application-neutral platform, which provides the opportunity to create applications that have access to the device hardware through series of APIs libraries. The following are common Android features:

- Free development, distribution and licensing
- Wi-Fi hardware access
- API libraries for implementing accelerometers, compass, camera, Bluetooth and location-based services like GPS (Global Positioning System).
- Framework for localization
- Support for 2D and 3D graphics using OpenGL ES 2.0 (Open Graphics Library for Embedded System)
- SQLite Database for storage and retrieval
- Media API that supports playing and recording audio and video format
- Open-source HTML5 (HyperText Markup Language) Webkit-based browser
- Shared data possibility through content providers, intents and notifications
- Support for background services for processes and applications
- Provision for application components' reuse and the replacement of generic applications.
- Memory and process management. [1,6.]

3.1 Android Development Framework

Each Android application is written in the Java Programming language and run on a separate instance of a virtual machine called Dalvik Virtual Machine (DVM) contrary to the Java Virtual Machine (JVM). The Android runtime is responsible for the memory and process management, which kills the process when the need arises to release memory to the Android runtime. The Android runtime and the Dalvik Virtual Machine sit on a Linux Kernel that interacts with the low-level hardware. A set of APIs is then necessary to expose the underlying hardware features and services. [1,12.]

3.2 Android Software Stack

Android applications run on top of a Linux Kernel and a collection of C/C++ libraries that provide applications with hardware functions of the device: hardware drivers, memory management and power management. The Linux Kernel is also responsible for process management, which starts a separate process for each application and multiple threads can execute the application in the same process. Figure 1 below shows the Android Software Stack.

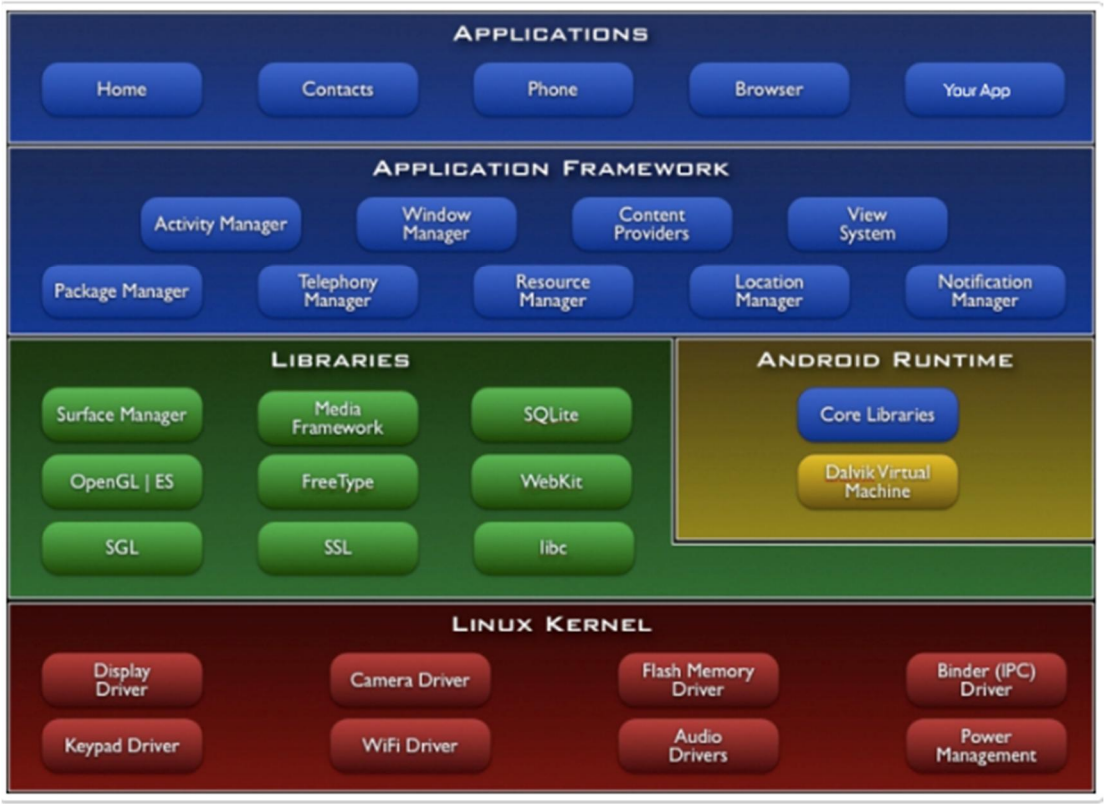

Figure 1. Android Software Stack. Reprinted from Android Architecture (2012) [3]

On top of the Linux Kernel are the native libraries that handle graphics, media, SQLite database, OpenGL, network, Secure Sockets Layer (SSL) & Webkit, Standard C Library (LIBC) and surface management. The Android run time, which houses the core Android libraries and the Dalvik Virtual Machine also sit on the Linux Kernel. The Android run time powers the applications. The core libraries provide most of the classes and functionality in core Java libraries as well as the Android generic libraries. Dalvik is optimized for mobile devices and makes it possible for Android applications to run on a separate instance or process effectively. It relies on the Kernel for allocating Central

Processing Unit (CPU) execution time for processes and their threads through scheduling. [2,1.]
On top of the Native libraries and Android runtime is the Android Application Framework. It provides the classes and Android APIs for creating Android applications. The APIs include, for example, Graphical User Interface (GUI), telephony, resource manager, locations, content provider, window manager and views. The Application Framework also gives a generic abstraction for the hardware access and functionalities.

On top of the Android framework is the Application Layer. All applications such as home, contact, settings, games, maps, browsers are built on this layer.

3.3 Android Phone Application Architecture

The Android application architecture is designed to promote reusability of components. Components are the architectural cornerstones or building blocks of an Android application securely managed by the Android runtime. Each of these components establishes an entry point through which the Android runtime can enter into Android applications since there is no other entry point like *main()* function as found in other development platforms.

Each component is a unique entity that does not only define the overall behavior, reponsibilities and lifecycles of a particular Android application, but also shares data and interacts with other components through intents, notifications and content resolver. The following are the four Android application components: activity, service, content provider and broadcast receiver.

An *activity* is a component that represents a single screen, which the user interacts with in a given time. It presents to the user the UI components like buttons, texts, images and navigations that are shown on an active screen [2,3].

When a long-running operation or a remote process is triggered, a *service* is the component that enables this application to run in the background without direct user interaction. To implement a service, the application must be a subclass of *service class* from the Android core library [4].

The *content provider* is used to share data within or between applications. Android applications get access to the native Android SQLite database through the content provider [2,5].

A *broadcast receiver* is a component that listens and responds to broadcast announcements from intents sent from the system or other applications. The broadcast receiver, by means of filters, detects the targeted Incoming intents [2,5].

3.4 Android Phone Application Development

The Android application development practice is focused on optimum user experience. The Android SDK (Software Development Kit) provides a set of tools and APIs libraries to write robust and sophisticated mobile applications. The Android application development steps revolve around four development phases as shown in figure 2 below.

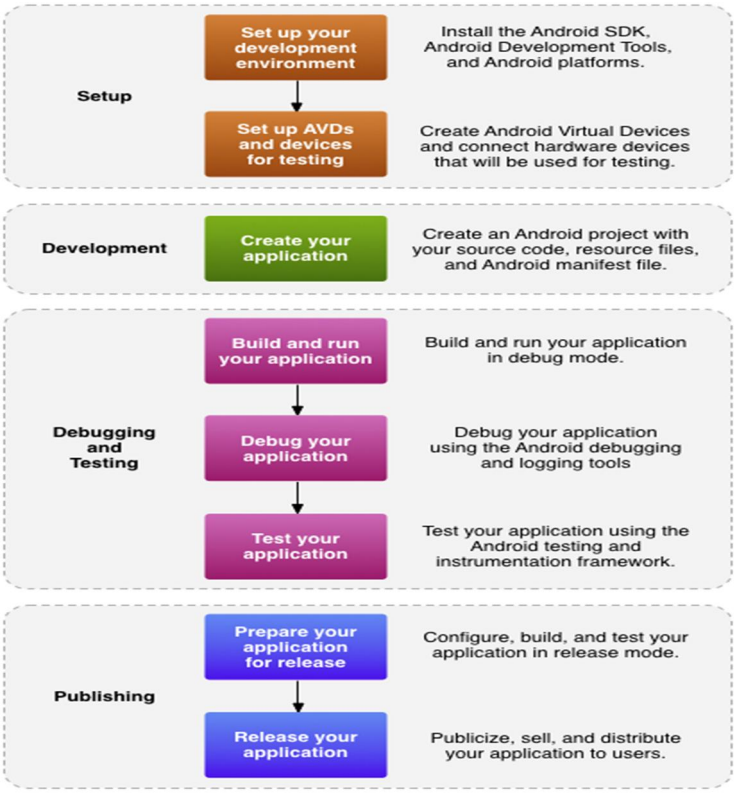

Figure 2. Development process for Android application. Reprinted from Android Open Source Project (2014) [4]

The development phases are as follows:

1. The Development Environment Setup Phase:

In this stage the free-to-download Android SDK is downloaded, installed and set up for development. The Android Virtual Devices (AVDs) on which an application can be installed for testing are also created.
In addition, the Eclipse IDE (Integrated Development Environment) and Eclipse ADT plug-in for Android development are also installed as they offer significant advantages.

2. Project Setup and Development Phase: This stage is where the setup and the actual development of Android Project are done. The Android Project folder contains all the source code, Android manifest and resource files for the application.

3. Building, Debugging and Testing: During this stage the application can be built into a debuggable Android package(s) that can be installed and run on an actual Android device or the emulator. The application is then debugged using the Android Debug Monitor and logging tools (logcat) provided by the Android SDK.
Next, the application is tested using various Android SDK testing tools and instrument framework.

4. Publishing: During this stage, the application is prepared for release by configuring, building and testing it in the release mode. Thereafter, the application is published, sold and distributed to users.

4 Requirement Analysis

In software engineering, application developers need to clearly understand the problems to be solved. It is therefore important for a developer to properly model the scenarios that can influence the solution to the problem by collecting relevant information. This process is called requirement analysis. The requirement analysis provides the opportunity for a developer to get a better understanding of the problem in question.

For effective design and development of this project, the following requirements must be met. They can be divided into functional requirements and non-functional requirements.

4.1 Functional Requirements

This section describes different requirements that are accomplished by the Online Bookstore system.
In order to achieve the desired goal of this project, the functional requirements must be met. The following are the three major actions performed by the Online Bookstore system.

4.1.1 Input Requirements

Input requirements are the requirements a user must fulfil before gaining access to use the Online Bookstore system. A registered user can use the application by providing the correct sign-in credentials. After being authorized, the user should be able to browse through the collection of books, view description of interesting books, select desired book(s) into the shopping cart, update the carts and finally purchase the book(s) using PayPal.

By making use of the *Skip* button to avoid signing in, a non-registered user can also utilize the application but with limited application functions. For example, a non-registered user may not be able to purchase an item.

4.1.2 Operational Requirements

The operations that the Online Bookstore system performs are registering users into the system and encrypting users password with SHA256, checking the availability of an Internet connection and downloading data from the server to the device's own native database (Android SQLite database) in JSON format and displaying the content of the SQLite database to the user via the UI.

4.1.3 Output Requirements

Output requirements ensure that the application is able to call the PayPal API server to complete the final process of the purchase. A receiver of the funds (i.e. the merchant) must have a PayPal account to receive the funds after the payment is complete. Figure 3 below shows how the receiver, the sender and PayPal interact with one another.

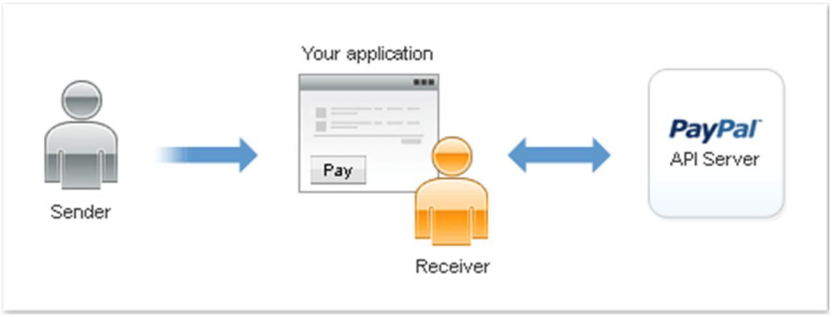

Figure 3. Relationship between a sender, receiver and the PayPal. Reprinted from PayPal (2014) [5]

The user is required to log into the PayPal account to approve the payment. The device sends a *Pay* request to PayPal. PayPal then responds, giving a key to prove the user authenticity. The user is then redirected to the PayPal website with the key for the final payment of funds. After the purchase, PayPal alerts the user of the payment and shipping information [5].

4.2 Non-Functional Requirements

Non-functional requirements are requirements that do not affect the proper running of the Online Bookstore system. However, it is worthwhile to mention and consider these requirements for the purpose of software quality and analysis.

4.2.1 Software Requirements

The Online Bookstore project, like other software engineering projects, needs well-defined specifications that must meet the software environments needed to achieve the desired goal of the project.

The software requirements considered in the development of this project are highlighted below:

- The application runs on Android 2.3, Android 3.0, Android 4.0 or higher (Google API level 20 – KitKat)
- Eclipse Juno Service Release 2 for Java EE Developers: Eclipse Juno Service (Release 2 is the Integrated Development Environment used in developing this project)
- Android Development Tool (ADT) is installed as a plugin from within the Eclipse IDE to provide all the Android tools needed for the application development
- Classic API for PayPal and PayPal Sandbox Account from PayPal Android SDK: It provides PayPal library for PayPal integration (for accepting credit card and PayPal services in the application)
- Server: mysql.metropolia.fi via Transmission Control Protocol/Internet Protocol (TCP/IP).
- Database Server type: MySQL.
- Protocol version: 10
- Server version: 5.0.95 – source distribution
- MySQL database server managed through PhpMyAdmin user interface.
- phpMyAdmin version: 4.1.8

4.2.2 External Interface or Hardware Requirements

In the development of this application, certain hardware requirements and specifications were considered in order for the application to be functional and result-oriented. This application is developed for Android-based phones and tablets and hence runs on an Android OS platform. With the help of the Android application framework and APIs, this application can utilize the device hardware features such as camera, sensor and touchscreen capability. The following are the hardware specifications:

- The device must support for minimum network capability (EDGE - Enhanced Data Rates for Global Evolution, HSPA – High Speed Packet Access, EV-DO – Evolution Data Optimized, 802.11g and Wi-Fi)
- The device must have at least 128MB of memory available to the Linux Kernel
- The device must have at least 1GB of non-volatile storage for user data
- The download Manager capable of downloading individual files of at least 55MB in size
- The device must implement at least a soft keyboard for user input
- The device must have a touchscreen (capacitive or resistive touchscreen)
- Support for OpenGL ES 1.0
- The device must support dynamic orientation by application to either portrait or landscape screen orientation

4.2.3 Secondary Requirements

Secondary requirements are future considerations that can be incorporated into the application for further development:

- Ability to integrate history page in the application
- Ability to receive notifications through the Android Notification service when new books are available on the server
- Ability for the user to suggest book(s) of interest
- Ability for the user to be able to read some portion of the book
- Ability for the user to comment on, recommend and rate an item. [8,24.]

4.3 Algorithm Flow Chart

A successful and dynamic application should have a flow of events that implements the system behaviour [6,12]. The term algorithm describes a solution to a problem. Algorithm is important in a problem-solving environment because it states the steps and procedures leading to the solution. In other words, an algorithm provides step-by-step procedures in solving a particular problem. [7,7.]

On the other hand, a flowchart is a diagram representing the flow of a process in a system. It combines symbols and flow lines, to visually represent the operation of the algorithm. The Algorithm Flowchart therefore is a figurative representation of the entire process, describing a set of instructions executed step-by-step to solve a given problem. [7,8.]

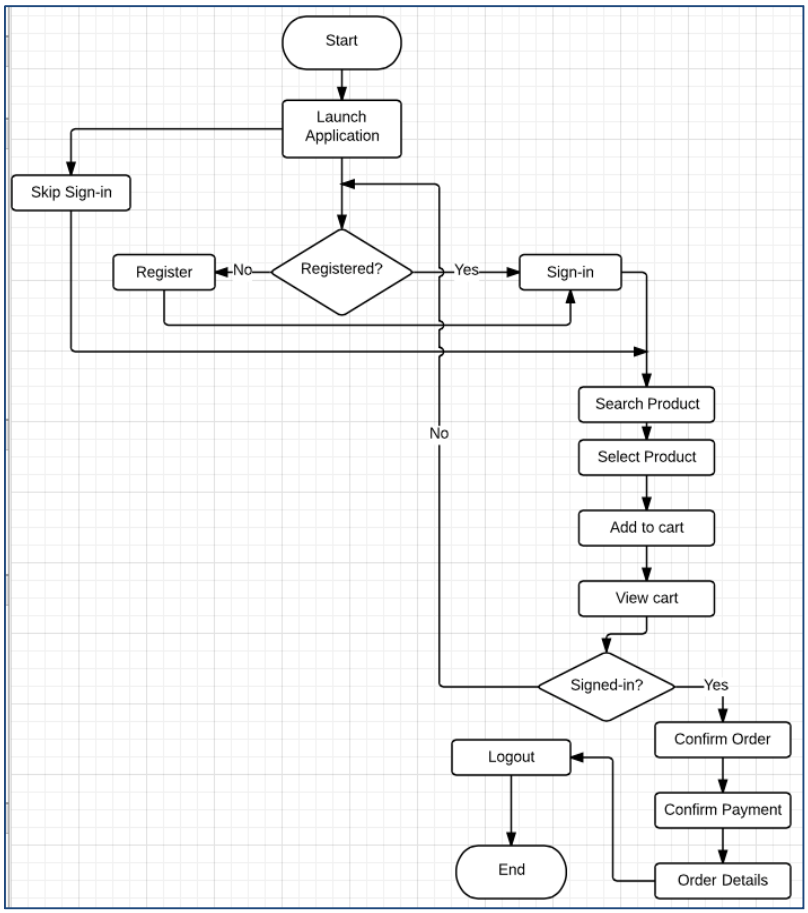

Figure 4. Normal Flow of Events in the Online Bookstore

The illustration in figure 4 above shows a complete process from start to finish, on how a user engages in the electronic transaction through the online Bookstore shopping system.

When the application is launched, the user has the option of either registering (for a complete transaction) or skipping the registration process (for quick browsing of book(s) without making a purchase). If the user choses to register, the system displays a registration form to be completed by the user. After this registration process, the user can log in with the login credentials in order to search for books, select desired book(s), add book(s) to the cart and purchase book(s).

4.4 Use Cases

A use case describes how a system behaves and responds to inputs by the primary actors. The primary actors interact with the system in a bid to accomplish a desired goal. The system, on the other hand, responds to the actors with the expected results. Figure 5 describes the use case of the Online Bookstore system. The use case brings together all possible scenarios under which a given system can act or behave. [10,1.]

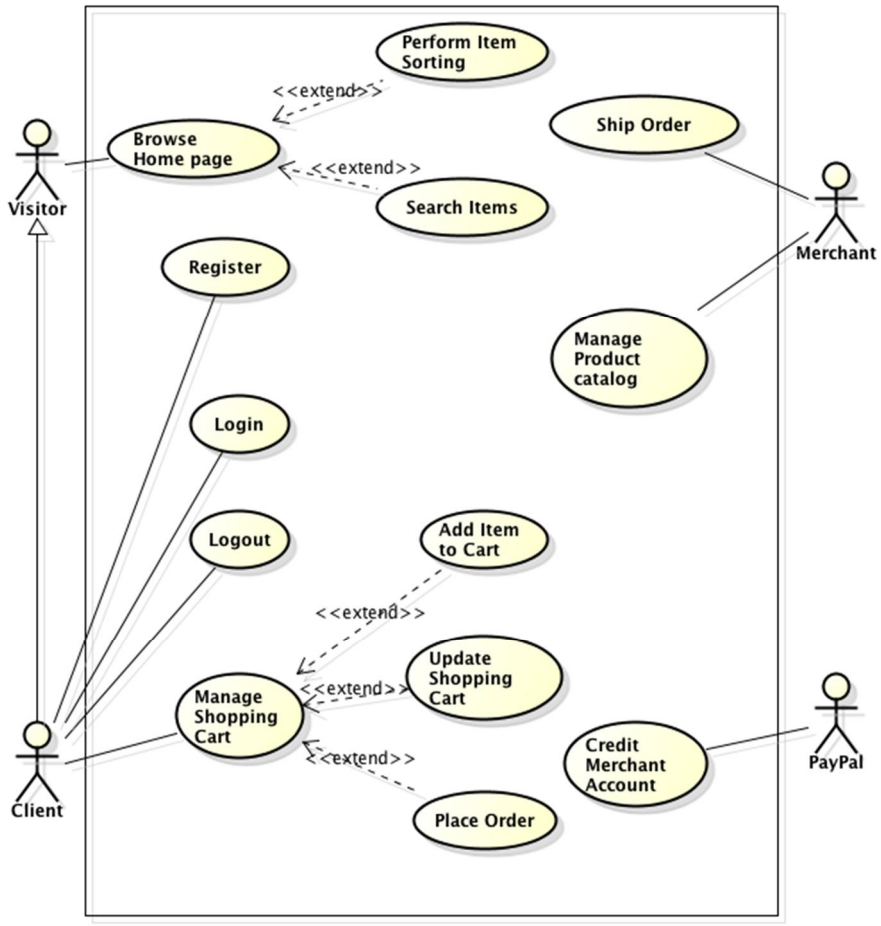

Figure 5. Use Case Diagram for the Online Bookstore

In this project, the actors are the CLIENT, VISITOR, MERCHANT and the PAYPAL service. The use cases are REGISTER, BROWSE HOME PAGE, UPDATE SHOPPING CART, LOGIN, PLACE ORDER, SHIP ORDER, MANAGE SHOPPING CART, CREDIT MERCHANT ACCOUNT, MANAGE PRODUCT CATALOG, LOGOUT, ADD ITEM TO CART, SEARCH ITEMS, and PERFORM ITEM SORTING.

The actors of the system are described below:

CLIENT: The client is a registered and authorised user also known as a customer. The client is able to log in to the system to purchase items from the online store.

VISITOR: The visitor is unregistered or an anonymous user that is able to search for and view items from the online store system. The visitor cannot log onto the system and make a purchase unless he/she is registered.

MERCHANT: The merchant represents a person selling the product through the online system, populates the product catalog and updates any information related to the products in the system. Also, the merchant ships orders to the client.

PAYPAL: The PayPal is responsible for validating a client's credit card information, debits the client's account and credits the merchant's account during transaction.

Furthermore, the use cases of the system are described in detail below:

REGISTER: A new user that wants to purchase a book must register into the database prior to a transaction. The application displays the registration form to the user. Table 1 shows the *use case* describing this activity.

Table 1. Use case description for REGISTER

Use case	REGISTER
Description	The application displays the registration form to the user for the user to register
Pre-condition	Unregistered user
Standard flow	1. The user clicks the REGISTER button from the home page 2. The application's registration page is displayed 3. The user fills in the form with appropriate information 4. The user clicks the REGISTER button 5. The system checks if all the required fields are entered and if the user's name does not exist already to avoid duplications. 7. If true, the system saves the new user's record in the *Users table* on the database.
Post condition	The user should be able to log in

The user should be able to log in if the system has successfully registered the user credentials into the database.

LOGIN: A registered user that wants to buy book(s) needs to LOGIN before any transaction can take place. Table 2 below illustrates the activity of the use case.

Table 2. Use case description for LOGIN

Use case	LOGIN
Description	A registered user signs in to allow access to the application
Pre-condition	The sign-in credentials provided during registration should be valid and the user must exist.
Standard flow	1. The user clicks the LOGIN button on the homepage of the application. 2. The system displays the login page 3. The user enters username and password. 4. The user clicks the LOGIN button 5. The system authenticates the user's record against the ***User Table*** in the database
Alternate flow	1. The user clicks the LOGIN button on the home page. 2. The system displays the LOGIN page 3. The user enters user name and password. 4. The user clicks the LOGIN button.
Post condition	1. If the user is an authorised user, the system displays the products page containing the list of books. 2. If user is not an authorised user, the system displays an error message, informing the user of 'incorrect password of user name'

The login credentials of the user are encrypted with SHA256 random generator encoding.

SEARCH ITEMS: The SEARCH ITEMS use case *extends* BROWSE HOME PAGE and allows the user to search the Product and Promotions pages for a particular item, browse through items and perform sorting of the items. The user can find an item quickly by the name or title of the book using the search button on these pages. The *skip* button on the application makes it possible for an unregistered user to search through the collection of books. In this way, the *skip* button makes the application available to all users but offers limited access to the Online Bookstore system re-

sources. For example, an unregistered user cannot buy books from the store if the *skip* button is used.

PERFORM ITEM SORTING: The PERFORM ITEM SORTING use case *extends* the BROWSE HOME PAGE use case. In addition to searching books by name or title, a user can also search items by sorting. The sorting can be in various orders that include: ascending or descending order of price, category, and title.

ADD ITEM TO CART: The ADD ITEM TO CART use case extends MANAGE SHOPPING CART and provides the user with the opportunity to place an order. The user must be registered and successfully logged in before adding items to Cart. Table 3 below illustrates the activity of the ADD ITEM TO CART use case.

Table 3. Use case description for ADD ITEM TO CART

Use case	ADD ITEM TO CART
Description	The user places selected item(s) into the CART and confirms order
Pre-condition	The user has signed in
Standard flow	1. The user searches by TITLE, NAME or CATEGORY 2. The system displays the search results on the Products page. 3. The user selects book(s) and clicks the BUY button. 4. The system adds the book(s) to the CART page for an order
Post condition	The user is able to view the CART for review

The user has the opportunity to review or edit the contents of the CART before placing an order. The user must have signed in with the correct username and password before making a purchase.

PLACE ORDER: The PLACE ORDER use case describes how the client completes a purchase by checking out from the shopping cart. It provides the user with the opportunity to make a payment for an order. Thus, a user must be registered before making a payment using the PayPal paying system. Table 4 below illustrates the activity of the *PLACE ORDER* use case.

Table 4. Use case description for PLACE ORDER

Use case	PLACE ORDER
Description	The user can buy the books in the CART using PayPal
Pre-condition	1. The user has registered, signed in, and has at least one item in the CART.
Standard flow	1. The user clicks the CART button. 2. The system displays the list of books in the CART. 3. The user checks the list for accuracy. Then, the user clicks the PAY WITH PAYPAL button 4. The system displays the PayPal login window. 5. The user enters the PayPal email address and password and clicks the LOGIN button 6. The system displays the PayPal REVIEW window containing the shipping address and the balance. 7. The user clicks the PAY button if everything goes fine and clicks the CANCEL button if not
Post condition	1. The user views or updates the shopping cart 2. The user views the shipping cost and taxes 3. The user views the sum total 4. The user makes payment with the PayPal account.

As seen in Table 4 above, the PayPal service provides a method of payment, which makes it easier to accept credit cards online and hence, provides an effective way for a user to have a smooth checkout experience. Before the payment is accepted, a user must have a PayPal account with login credentials. After PayPal accepts the payment, the system confirms the payment and provides the client with order status. This means that there must be at least one item in the cart before placing an order.

SHIP ORDER: The SHIP ORDER use case describes how the merchant supplies the products ordered to the client. The products are shipped only if the payment has been

accepted and the merchant account has been credited. The products are shipped to the client's shipping address.

BROWSE HOME PAGE: The BROWSE HOME PAGE use case allows both the visitor and the client to view a list of products on the product page, which is stored in a relational database. Before this can be performed, a database populated with a list of products must exist. User first requests a list of products in the database, then the web server connects to the database and displays all available products on the product page.

UPDATE SHOPPING CART: The UPDATE SHOPPING CART use case allows the client to update the shopping cart. The client has a user account and must have logged into the system. The client can remove item(s), add item(s) or empty the cart.

MANAGE SHOPPING CART: The shopping cart is implemented on the application. This use case describes the activities the client can perform on the shopping cart. These include browsing the product page, viewing the cart, adding to or removing items from the cart and editing the quantities of items in the cart.

CREDIT MERCHANT ACCOUNT: The CREDIT MERCHANT ACCOUNT use case is utilised by the *actor* PAYPAL in order to credit the merchant account when a client has successfully placed an order. The PayPal service validates the client's credit card information before crediting the merchant account. The system displays the transaction details.

MANAGE PRODUCT CATALOG: The MANAGE PRODUCT CATALOG use case allows the merchant to manage the products page. The merchant can add to, remove or update product prices in the catalog. The merchant can also manage the inventory and sales.

LOGOUT: The LOGOUT use case requires that a client has been log into the system. The client is able to click *logout* after a successful transaction or at any point of the process to interrupt the system. The system displays a prompt to confirm if a user intends to log out. If the client selects *logout*, the system returns the client to the *login* page and ends the session. Otherwise, the client continues from the last page before clicking *logout.*

5 The Online Bookstore Graphical Design

A Graphical User Interface (GUI) is a user-friendly interface, which allows a user to use icons or other visual elements to interact with the application. In this section, the different GUI elements and their implementations are analysed in detail.

5.1 Design View of Start-up Window

In this application, the first *activity* that launches is the start-up window that presents the *Sign-in*, *Register* and *Skip* buttons logically on the GUI. The start window is shown in figure 6 below. The layout of this *View* is created with the *activity_login.xml* file from the layout folder of the Android application project.

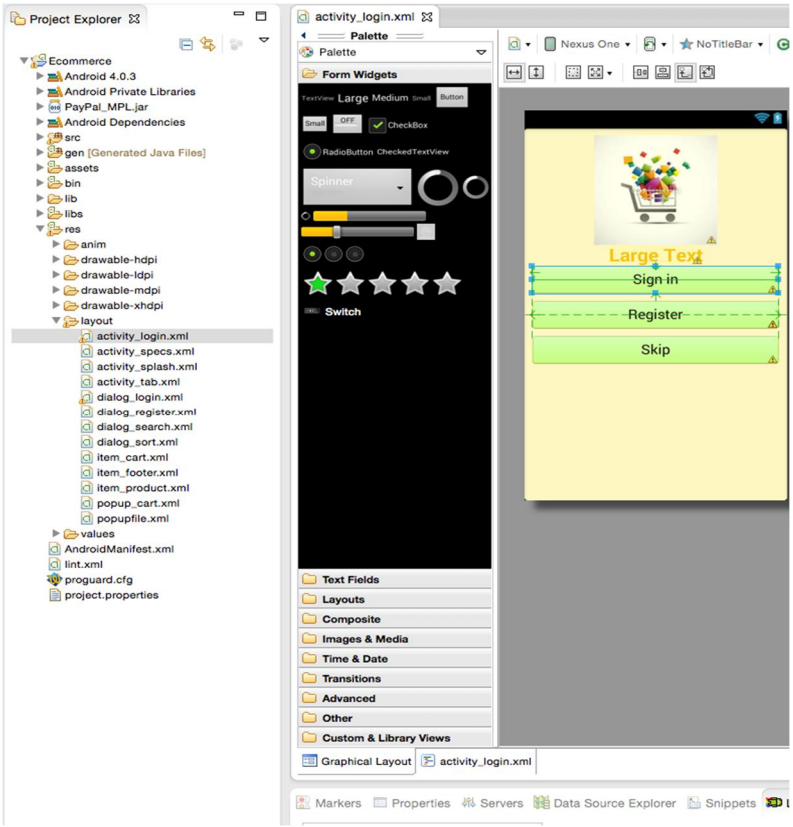

Figure 6. Start-up Window

The layout is a *Relative Layout* that consists of three buttons (Sign in, Register and Skip), one *TextView* (contains the “Hello username” text) and an *ImageView* (contains the online store logo). The button *Views* are from the *Form Widgets* folder and the *Im-*

age and *Media* folder provides the *View* that holds the logo. All the three buttons have “clickable” attributes set to *true* to make the buttons provide an action when clicked.

The registration form layout is displayed with the help of the *dialog_register.xml* file. This form contains all required information that should be filled in by the user. The user should fill in the form appropriately and submit by clicking on the *Register* button. *Figure 7* below shows the registration page of the application.

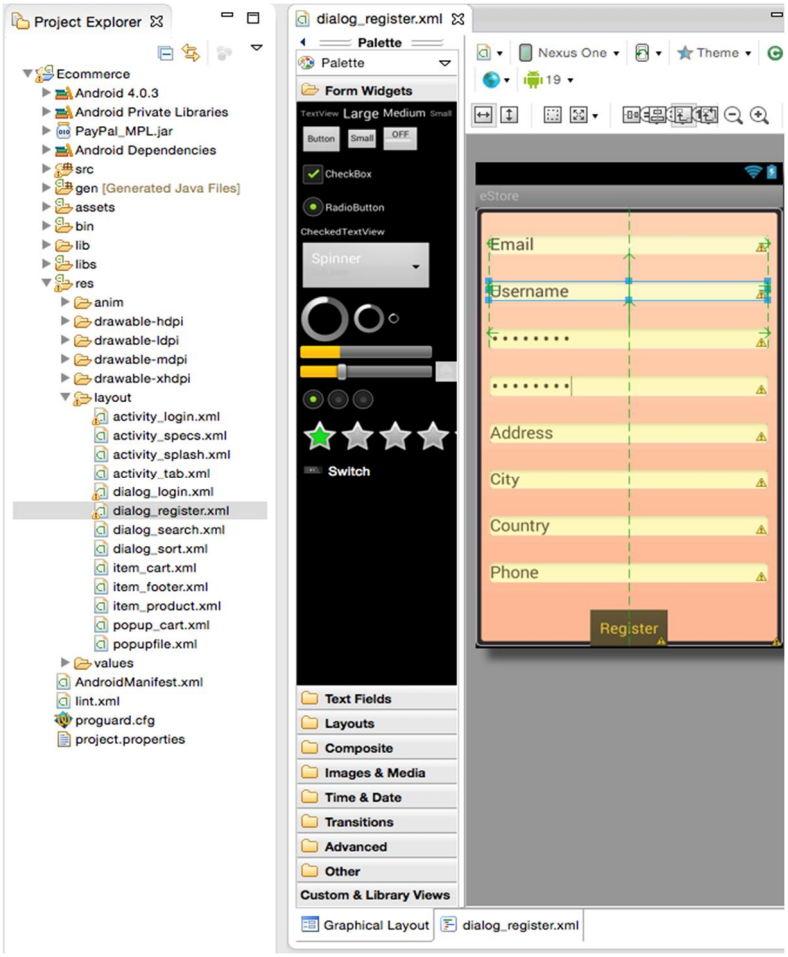

Figure 7. Registration Page

The registration page contains eight *TextViews* and a *register* button. The *Email* field has ”textEmailAddress” *Input Type* that validates the user input whether it follows the correct email format. Also, the *password* field has a ”textPassword” *Input Type* with *Visibility* attribute set to ”true”. This makes the input from the user unreadable.

5.2 Application Control Buttons

The four control buttons are: *Product tab*, *Promo tab*, *SendTo tab* and *Cart tab*. These buttons are important in this application because they provide the medium for the user to interact with the application. The following describe these control buttons in details:

1. Products tab: This tab is connected to the *Product* ListView, which displays the list of items on the GUI. This *Product* ListView is populated by items from the *Products table* on the database. If the *Products tab* is clicked, the application checks if there is an Internet connection. If this is true, data from the *Products table* that are not in the *Promotion field* (WHERE pPromotion=false) are parsed from MySQL database. Figure 8 illustrates the *products list* page when the *Products tab* is pressed. Each row parsed from the Products Table creates a new row on the mobile phone sqlite local database. On the other hand, if there is no Internet connection, the items information is parsed from the sqlite database instead. A PHP script form the server is used to access the *Products table* on the database.

Figure 8. Products list page

Ten rows containing products data are parsed at a time. When the end of the list is encountered, a new request is made through the Hypertext Transfer Protocol (HTTP)

to the database. This process is repeated when the user scrolls down for more lists until all the rows from the database are successfully parsed to the mobile phone.

2. Promo tab: This tab is connected to the *Promo* ListView, which displays the list of *Promo* items on the GUI. The *Promo* ListView is populated by products from the *Products table* into the database as shown in figure 9 below.

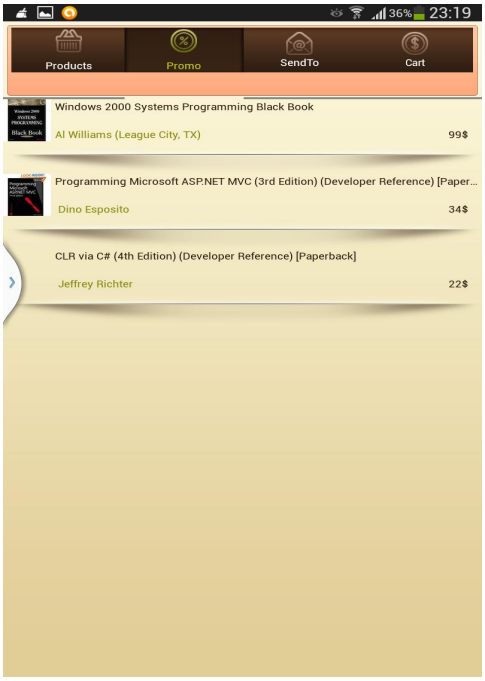

Figure 9. Promo list page

If the Promo tab is clicked, the application will check if there is an Internet connection. If this is true, only data from the Products Table (WHERE pPromotion=true) are parsed from MySQL database to the phone. This data creates a new row on the mobile phone sqlite local database.

3. SendTo tab: This tab allows user to share this application with friends simply via an email address or by mobile phone number. When the *SendTo* tab is pressed, the application displays a page with a *Text Area* View for the user to enter a valid email address or valid mobile phone number as shown in figure 10 below.

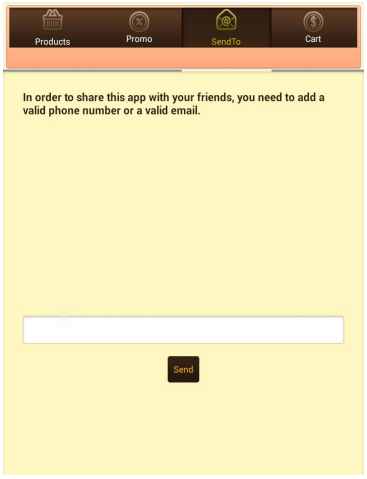

Figure 10. SendTo page

When the *send* button is pressed, the web link to the application on the Android Play Store is sent to the email address or the mobile phone number provided.

4. Cart tab: The cart tab allows the user to view for item(s) in the cart. The PayPal button for the payment is only available when a registered user is logged in as shown in figure 11.

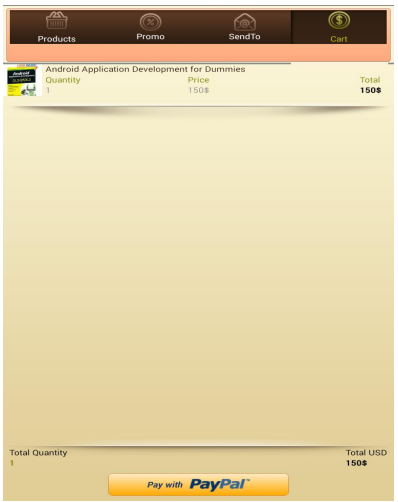

Figure 11. Cart page for registered user

With the cart page, a user can view the summary of the cart contents, the quantity of item(s) and the sum total of item(s).

On the other hand, if the user decides to use the Skip button to view items in the store, the cart page displays the content of the cart with the *Register* button and the *login* button but not the PayPal button as shown in figure 12.

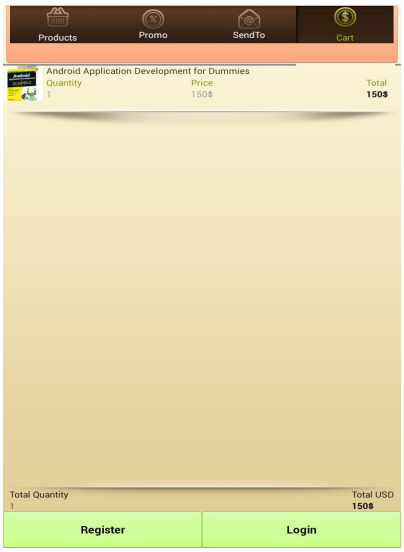

Figure 12. Cart page for unregistered user

As shown in *figure 12* above, the *Register* button, when pressed, will take an unregistered user to the application's *registration* page while the *login* button takes a registered user to the *login* page for signing in. When the user makes a *long-press* on any item in the *Products*, *Promo* or the *Cart* page, a popup window is displayed for adding items to the cart or removing items from the cart as shown in figure 13 below.

When an item is *long-pressed* on the cart page, a new window is popped up. The *minus sign* icon on this window is used to reduce the number of items or to remove items from the cart. The *buy* button on the window confirms the action.

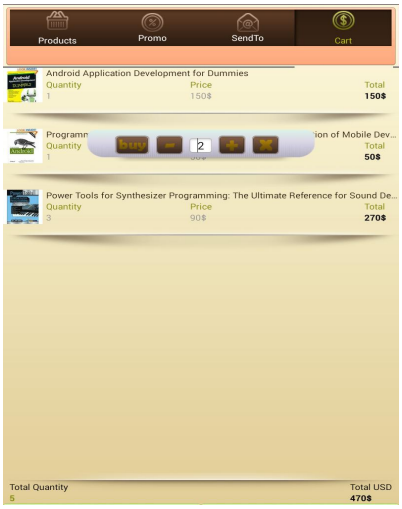

Figure 13. Adding or removing items to Cart

On the contrary, the *plus sign* is used to add more items to the cart. The same process is implemented for adding item(s) to the cart from the *Products* and *Promo* pages.

In addition, if a user clicks on any of the item in the Products, Promo or Cart page, a new *activity,* which displays a full description of the item will be launched as shown in figure 14 below.

Figure 14. Page with item description

Figure 14 does not only show the description of the item clicked but also the price, name and author of the book.

On the other hand, if a registered user logs in with the username and password, the user cannot only view all the items in the store but can also buy the items using the PayPal button. The page with the PayPal button is shown in figure 15 below.

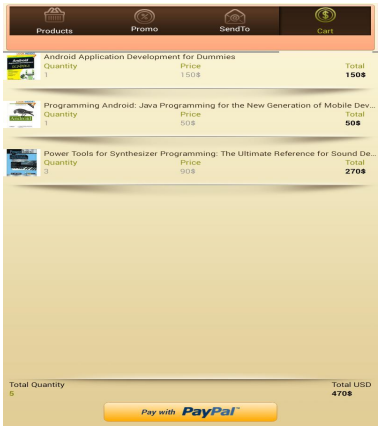

Figure 15. Cart page with PayPal Button

The PayPal button for payment of selected item(s) is available only on the cart page. This button, when clicked, will display the PayPal authentication page where the user is prompted to log in to the PayPal account for payment. The PayPal authentication page is shown in figure 16 below.

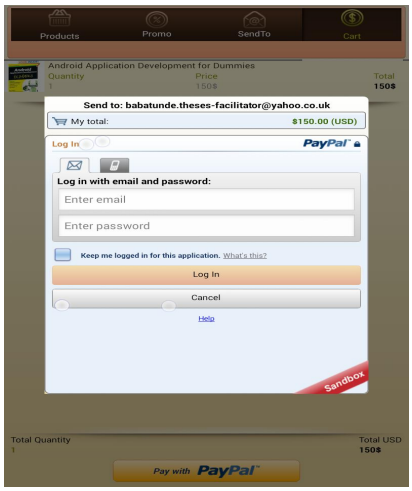

Figure 16 PayPal authentication page

The user must be registered and must have been successfully logged in before using the PayPal button. As shown in figure 16, the page displays the sum total of the items in the cart. The user can either logs in to the PayPal account with an email address and password or a phone number and password. The user can also choose to cancel the transaction and return to the cart.

6 Implementation of the Online Bookstore

In this section, all the steps taken to implement the business logic of the Online Bookstore System is explained.

6.1 User Registration

When the user presses the registration button in order to register, a dialogue form that allows the user to send the required information to the MySQL server pops up. The first event that happens during the registration process is that the required fields are checked if empty or if the user already exists.

The first line of the code shows how the *email* field is checked if empty and if it already exists on the server. The same goes for all the fields of the registration form. Other fields are *username, password, address, city, country and phone number*. The user data is now ready to be transferred and added to the *Users table* on the database server. When the *Register* button is pressed, the application will establish an Internet connection with the server before parsing the data to the server and will store this information in the database if user does not already exist. This application utilizes a high level of security by protecting user's password with an SHA256 encryption. The user's password is therefore not exposed on the server when added to the *Users table***.**

On the contrary, if the user has already registered, the GUI pops up a window that notifies the user. Items from the *Products table* can be displayed without first signing in by clicking on the *skip* button. The *skip* button will skip the registration page completely. This option is important as it allows user to search through the store for item(s) without ordering.

6.2 User Login

A registered user is able to log in with the username and password provided during the registration process. If the user presses the *Sign-in* button, a dialogue form pops up that allows the user to enter the username and password. The form accepts the username and password of the user when the *Login button* is clicked. The Login window is showed in figure 17 below.

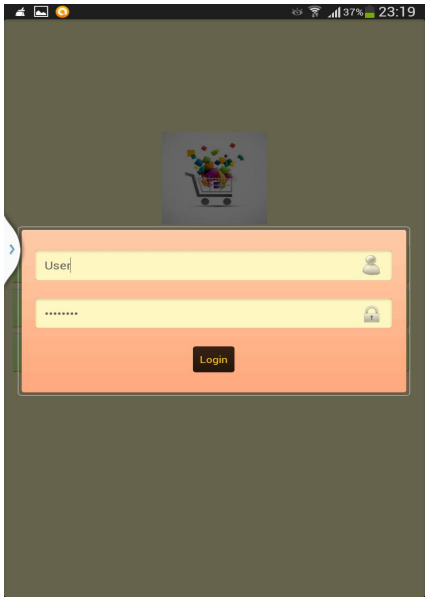

Figure 17 Login Window

The application does not interact with the server directly but uses a PHP code remotely to authenticate user's credentials with the user data already stored on the *Users table*. The PHP code interacts with the database in order to implement the validation. The script returns '1' if the username and password entered by the user from the mobile phone are the same as the one in the *Users table* in the database.

On the other hand, if the user enters an incorrect password or username, the PHP script returns '0' and the GUI displays, " sorry!! Incorrect password or username " as a *toast* to the user. The user is then offered another chance to re-enter the password and the username.

6.3 Parsing HTTP response to JSON Object

Since the application needs to access a RESTful API over HTTP protocol to interact with a remote server, it is therefore important to discuss how the application sends and retrieves data over the network from the server.

Android provides the Apache HTTP components library which includes methods for creating connections to remote host APIs. The API acts as an interface between the application and the server as shown in figure 18 below.

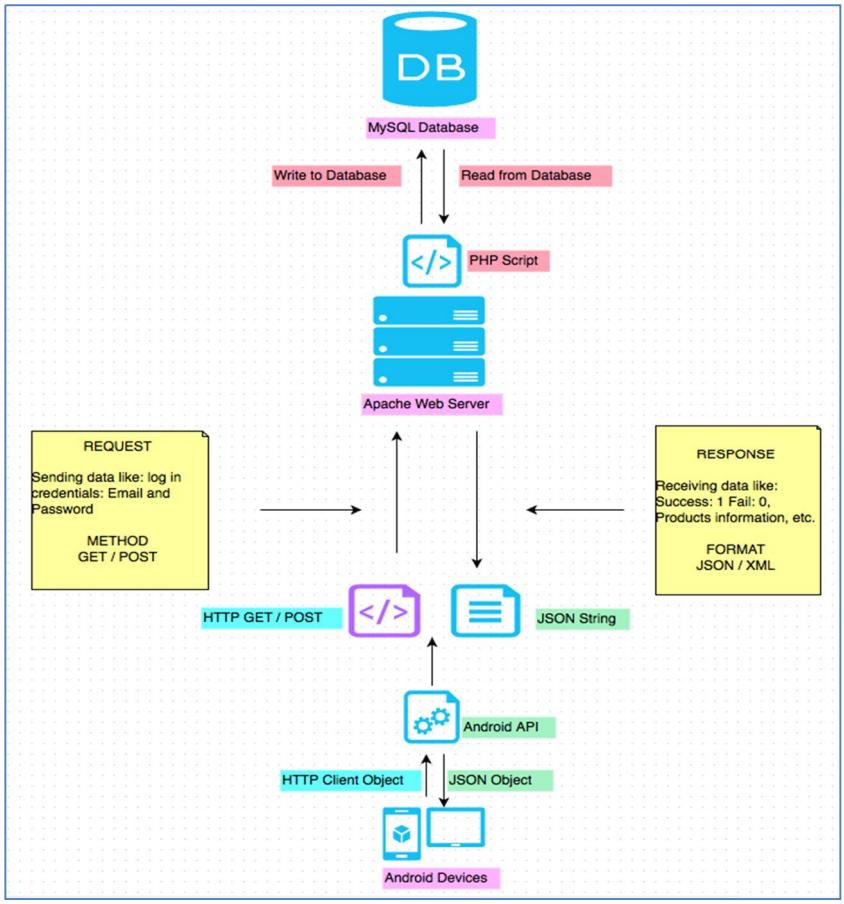

Figure 18. Android devices connecting to database

With Android support for the Apache HTTP library, it is easy to perform GET, POST, PUT and DELETE requests via the RESTful APIs. When the HTTP requests are made through the API, the responses are returned in a structured document format, JSON string.

The raw JSON response from the remote server cannot be read directly by users as it contains only strings of key-value pair characters. Hence, there is a need for the JSON strings to be converted to human-readable text that is understood by the application. The JSON string must therefore be parsed to the JSON object that the application understands. Fortunately, the Android SDK includes the ***org.json*** parser classes for efficient parsing of JSON formatted strings.

7 Database Design

The design of the database of this project is of profound importance as it is the fundamental building block to understanding the operation of the project. It presents, in an organised manner, the collection of data used in this project. The project utilises two sets of database, namely the internal database (SQLite) that is generic to the Android device and the external MySQL database located at mysql.metropolia.fi. The administering of the MySQL database is achieved with the phpMyAdmin program that is installed at https://users.metropolia.fi/phpMyAdmin.

The entire project database revolves around three major tables: the users table, the sales table and the products table. These tables in the database are managed through the PHP script. The database schema is shown in figure 19 below.

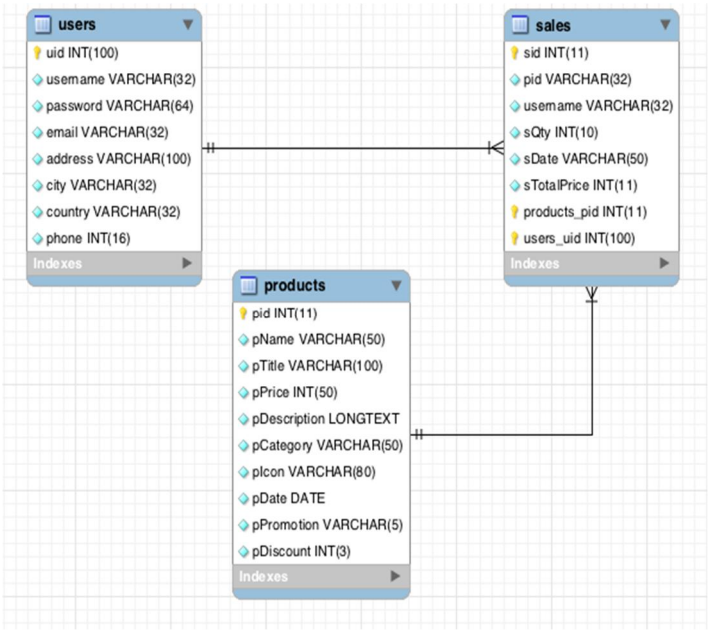

Figure 19 Database schema

The *Users* table contains the user's information and the login details such as: username, password, address, email and phone number. The *Products* table stores the product name (pName), title (pTitle), price (pPrice), description (pDescription), category (pCategory), icon (pIcon), date (pDate), products that are on discount prices (pPromotion) and discount price (pDiscount). Lastly, the *Sales* table contains the username, sales quantity (sQty), sales date (sDate) and sale total price (sTotalPrice).

In this project, a new database called ***babatud*** was created on the MySQL server (mysql.metropolia.fi). The products, sales and users tables were created on the ***babatud*** database and populated with the SQL scripts. Every transaction made using the online bookstore system is stored in these database tables including the users login credentials and information, product and sales details. The following describes their creation:

The *Products table* is created in the database by an SQL script. This code creates a table with 10 fields in the table. With the database table in place, an SQL script is used to populate data into the table as shown in figure 20 below.

Host: mysql.metropolia.fi
Database: babatud
Generation Time: Dec 03, 2014 at 02:18 AM
Generated by: phpMyAdmin 4.1.8 / MySQL 5.0.95
SQL query: SELECT * FROM `products` LIMIT 0, 25 ;
Rows: 18

pid	pName	pTitle	pPrice	pDescription	pCategory	pIcon	pDate	pPromotion	pDiscount
1	Donn Felker	Android Application Development for Dummies	150	Learn to: • Create apps for hot smartphones like DroidTM X, Galaxy S, and MyTouch® • Download the SDK and get Eclipse up and running • Code Android applications • Submit your apps to the Android Market	PROGRAMMING	img202138-27122011162947-0_150x150c_7hio.jpg	2012-05-28	false	0
2	Al Williams (League City, TX)	Windows 2000 Systems Programming Black Book	99	Windows 2000 Systems Programming Black Book: The Only Reference Needed to Successfully Deploy Applications Within the Windows NT Operating System!	PROGRAMMING	img217627-07032012144434-0_150x150c_js8g.jpg	2012-02-29	true	20

Figure 20. Sample data from the Products Table

The data in this table represent a book unique number(*pid*), the name of the book (*pName*), it price (*pPrice*), description of the book (*pDescription*), book category (*pCategory*), the book image or icon (*pIcon*), date published (*pDate*), promotion flag (*pPromotion*) and discount on price if any (*pDiscount*).

The *Sales table* is created in the database with an SQL script. The table contains six fields. The *Sales table* with sample data is shown in figure 21 below.

Host: mysql.metropolia.fi
Database: babatud
Generation Time: Dec 03, 2014 at 02:26 AM
Generated by: phpMyAdmin 4.1.8 / MySQL 5.0.95
SQL query: SELECT * FROM `sales` LIMIT 0, 25 ;
Rows: 3

sid	pid	username	sQty	sDate	sTotalPrice
7 [->7]	2	babatud	3	Apr 29, 2014 3:28:38 PM	297
6 [->6]	1	babatud	3	Apr 29, 2014 3:28:37 PM	450
8 [->8]	7	babatud	5	Apr 29, 2014 3:50:12 PM	900

Figure 21. Sales Table

The fields of the *Sales table* are: *sid*, which represents the sales unique number; *pid*, which represents the products unique number; *username* field which contains the username of the user; *sQty* field which represents quantity of sales; *sDate*, which represents the sale date and *sTotalPrice*, which describes the sale total price.

The *Users table* contains user information. The user data like username, password, email address, name of city, country and phone number are stored securely in the *Users table* as shown in figure 22 below.

Host: mysql.metropolia.fi
Database: babatud
Generation Time: Dec 03, 2014 at 02:23 AM
Generated by: phpMyAdmin 4.1.8 / MySQL 5.0.95
SQL query: SELECT * FROM `users` LIMIT 0, 25 ;
Rows: 1

uid	username	password	email	address	city	country	phone
10 [->10]	babtee	132da666eb5c890d97c9847c190e1541d560e90e9afdc6eef2ad6d348a6666b7	babsoniuk@yahoo.co.uk	Ratalaaksonkuja	espoo	finland	466691463

Figure 22. Users Table

The *Users* table has eight fields: The *uid* field represents the user's unique number. The *pid* field represents the products unique number. The *username* field contains the username of the user. The *sQty* field represents quantity of sales. The *sDate* field represents the sale date while the *sTotalPrice* field describes the sale total price. [2, 589.] The online bookstore system stores the user's password in this table. The password is encrypted using the SHA256 random code generator algorithm.

The MySQL database design needs to include more tables like Cart Table (that contains items in the cart), Promotions Table (a separate table from the Products Table that contains the promotion items) and Delivery Table (that keeps the status of shipped items and date of delivery). In the future, these option could be considered in order to improve the database.

However, the transaction details can be viewed from the merchant's PayPal test account and the PayPal test account of the user. Figure 23 shows the user payment confirmation page generated by the user PayPal test account.

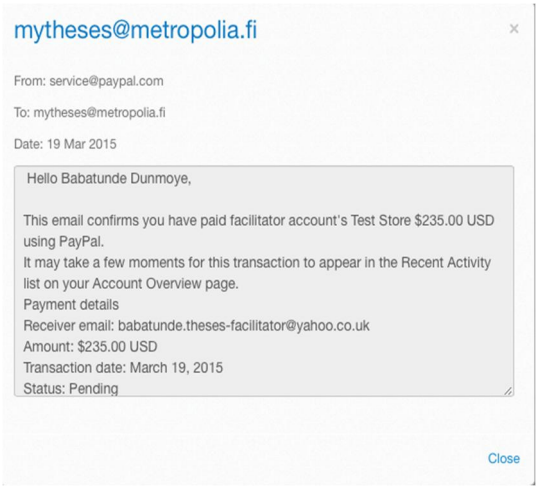

Figure 23. User payment confirmation page

The payment confirmation page sent to the user email contains the name of the user (customer), payment details, merchant email address, amount paid for the item(s), transaction date and status of the transaction.

Similarly, the merchant can also view transaction details from the PayPal test account. Figure 24 below shows the merchant transaction page.

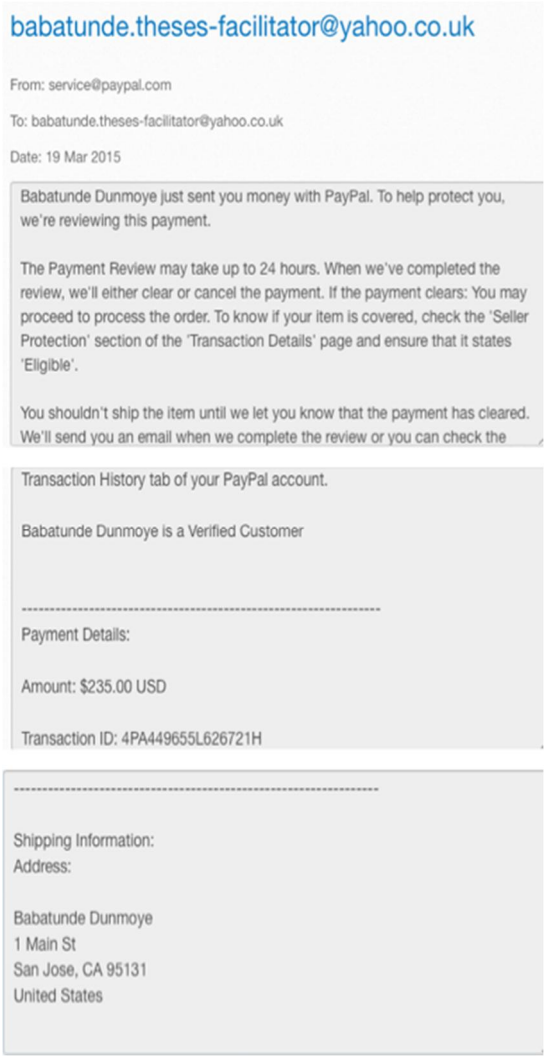

Figure 24. Merchant transaction page

This page confirms to the merchant that a user (customer) has made a payment for item(s). It also provides other information such as: user verification status, payment details, shipping address, date of transaction and name of the user.

8 PayPal Integration

In this project, PayPal is used as a method of payment, which makes it easier to accept credit cards online. PayPal payments provide an effective way for merchants to offer a smooth checkout experience to customers using mobile devices, such as Android devices. The PayPal features are used in a sandbox for the sake of testing this project. This means that PayPal Sandbox is rather fictitious compared to the live PayPal website. The Sandbox test accounts transactions are linked to *www.sandbox.paypal.com* while the live PayPal transactions are linked to the common *www.paypal.com*.

8.1 PayPal Sandbox Overview

The PayPal Sandbox is a self-contained environment, which allows developers to test and experiment with PayPal features and APIs. The PayPal Sandbox is similar to the live PayPal website. It mirrors the features of the PayPal live servers and so, helps developers to work within a development enclosure for testing and integration purposes before migrating to the live PayPal website. Since applications using PayPal features must work according to set guidelines and standards, PayPal Sandbox offers a testing platform to ensure the applications work within the set contract. While the Sandbox PayPal does not reflect all the live PayPal features (for example: fraud alert, account statements generation and credit card verification), the sandbox has support for a full PayPal live environment. This means that testing a PayPal features in the Sandbox environment will bring the same results as in the live PayPal servers. [9,9.]

8.2 Accessing the PayPal Sandbox

PayPal Sandbox is accessed by first creating a test account at *https://www.developer.paypal.com*. The Sandbox creates non-real bank accounts and credit card numbers, and simulates credit card number verification. Fraud detection is not enabled for the Sandbox PayPal. After signing up, the developer can sign in using the sign-in credentials provided at signing up. The developer then needs to set up two separate test accounts: The PayPal Personal test account and the PayPal Business test account.

- PayPal Personal Account: The user of this account can represent a buyer, a sender or a customer. When a transaction is instantiated on a Sandbox test ac-

counts, it will create a sham transaction that acts similar to a transaction in the live PayPal. This account sends the transaction funds to the merchant. Multiple PayPal personal accounts can be created if the application uses multiple users.

- PayPal Business Account: The user of this account can represent a seller, a receiver, a merchant or an API Caller. With the PayPal business account, user does not only provide goods and services to the buyer, but can also receive the transaction funds and represents the merchant in the Sandbox transactions. [11]

When a test account is created, the following fictitious information will be auto generated:

- The mailing address
- Bank account and credit card number
- Email address and password (real email address and password should never be used)
- Security question and answer. [9,17.]

8.3 Adding Funding Sources

To test how a transaction behaves, there must be a source of funds added to the buyer PayPal test account. The other account is a business account representing the developer as a merchant. This section describes how to add funding to the test account by either adding a bank account or by adding a credit card or both. It is important to note that no money or funds are actually transferred to the Sandbox PayPal accounts from these funding sources; therefore, to protect confidentiality, real credit card numbers or bank accounts should not be registered on the test accounts.

The bank account is a source of funds for the buyer test account, which allows the transaction between that test account and the business (merchant) account. When a bank account is added to a test account, the PayPal Sandbox will automatically generate a fictitious bank name, bank account number and sort code numbers.

Furthermore, adding a credit card is another source of funding for the buyer's PayPal account, and thus can be used to test transactions between a buyer's test account and

other test accounts. Of course, test credit card numbers cannot be used to make payments for real-world transactions. [9,28.]

9 Submitting the Application to the Android Market Place

An Android application can be submitted to the Android Market for distribution after being developed and tested both in the emulator and on a real Android device. Publishing an Android application on the Android Market requires that the developer register an account at *www.market.android.com/publish/*. There is a one-time registration fee attached to the registration. Developers must adhere strictly to the *Android Market Content Policy for Developers* during registration.

Furthermore, the developer must sign up for a *Google Checkout Merchant Account*, which provides payment services for online transactions and helps to protect developers from fraudulent sales, similar to PayPal. [1,48.]

Before publishing the application, the following points should be considered:

1. The *AndroidManifest.xml File* must be edited: This includes information about the application such as: package name, version numbers, supported screen size, SDK version, hardware configuration (like Touch screen, Hard keyboard, Keyboard type, and Navigation) and permission for certain features (like accessing location, sending SMS and setting wallpapers). [1,49.]
2. Icons for the application: The application must have an icon that represents it on the Android Market and on the user's device. The icon should be at least *512 x 512 pixels* and support multiple screen densities. [1,51.]
3. Turning off logging and debugging: The application's debugging features must be turned off before publishing. This is achieved by clicking the *Application tab* in the Android Manifest Editor and setting *Debugging* attribute of the *application* to false.
4. Application Versioning: It is a good practice to include a version name in the application as this makes future updates easy and simple to track by users.
5. Shrinking, Optimizing and Obfuscating the Application: It is important to shrink and optimize the application codes by reducing the size to optimize performance. Obfuscating application codes is the term used to prevent application codes uploaded to Android Market from reverse engineering. The *ProGuard* tool developed by Google helps to shrink the size of the application's package file *(.apk file)*. It also optimizes and obfuscates the application codes while *Zipalign* is the tool for optimizing the application's memory usage. [1,52.]

6. Digitally signing the Application: Android application's package file *(.apk file*) must be digitally signed before uploading to the Android Market or other application marketplaces. Using a digitally signed application identifies the developer of the application as the original owner and author of the application. The digital certificate includes, for example, the developer's name, contact information and date the application was signed. The Java Development Kit (JDK) contains the tools for digitally signing an Android application. The tools are: *Keytool* for generating a private key and *Jarsigner* for signing the Android application package. [1,53.]
7. Uploading Assets: When the application has been digitally signed, it is ready to be uploaded to the Android Market. This is done by logging into the Android Market at www.market.android.com/publish and clicking the *Upload Application* button on the page that opens. This starts the process of uploading the application's package (.apk file) file, which includes the application's code file (.dex files), assets, resources and the manifest file. The upload will start immediately when the *upload* button is clicked. [1,55.]

10 Conclusion

The project was carried out to develop an Android phone application for an online bookstore system with PayPal Integration. The goal of the project was to create an Android application that would allow users to register an account, login, search for particular books of interest, sort books in ascending or descending order of price and purchase book(s) in the cart with a PayPal account.

The objectives of the project were achieved by observing software development procedures and principles for software designs and implementation. In achieving the goal of this project, three major parts were designed and implemented. Firstly, the design of the UI is attractive, intuitive, responsive and with good user experience in mind. This was achieved and implemented by following the Android design guidelines for Android devices. Secondly, the design of the MySQL database was realized, and the contents of the database can be easily accessed indirectly from a web server instead from the Android device. This makes the database secure and easy to manage. Thirdly, the design and the implementation of the Android phone local SQLite database was achieved. This allows users to use the application offline.

Furthermore, all required functionalities were implemented accordingly and, hence, a fully operative and functional Android phone application was developed. The application is able to register an account for users, while protecting and encrypting a user's password. The users can log in conveniently with the username and password, search for available books on the online bookstore, order books and make purchases using the integrated PayPal services. In addition, users can log out from the online bookstore and can even recommend the application to friends within the application.

In conclusion, it is important to know that this application could still be improved upon by adding more interesting features. For example, a feature that makes the user to save the purchase history could be added. In addition, the application could demonstrate a feature that allows the user to not only rate books, but also recommend book(s) of interest to friends.

References

1 Mewer R. Professional Android 2 Application Development. Indianapolis: Wiley Publishing, Inc.; 2010.

2 Göransson A. Efficient Android Threading. Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc.; 2014.

3 Android-App. Android architecture–The Key Concepts of Android OS [serial online]. February 2012.
URL: http://www.android-app-market.com/android-architecture.html.
Accessed December 11, 2014.

4 Android Open Source Project. Application Fundamentals – App Components [online]. Google; May 2014.
URL: http://developer.android.com/guide/components/fundamentals.html.
Accessed December 11, 2014.

5 PayPal. Introducing Adaptive Payments [online]. PayPal Inc.; August 12, 2014.
URL: https://developer.paypal.com/webapps/developer/docs/classic/adaptive-payments/integration-guide/APIntro.
Accessed December 12, 2014.

6 Gilmore J. Beginning PHP and MySQL from Novice to Professional 4th edition. New York: Apress; 2008.

7 Lim J. Algorithms, Flowcharts, Data Types and Pseudo Code [online]. CA: scribed; January 2011. URL: http://www.scribd.com/doc/46981114/Algorithms-Flowcharts-Data-Types-and-Pseudo-Code. Accessed 13 November 2014.

8 Google. Android 2.3 Compatibility Definition [online]. Google Inc; 2010.
URL: http://static.googleusercontent.com/media/source.android.com/en//compatibility/2.3/android-2.3-cdd.pdf
Accessed December 12, 2014.

9 PayPal. Sandbox User Guide. Luxembourg: PayPal Inc.; 2010.

10 Alistair C. Writing Effective Use Cases. USA: Pearson Education; 2011.

11 PayPal. Planning Your First Account [online]. PayPal Inc.; 1999 - 2014.
URL: https://developer.paypal.com/webapps/developer/docs/classic/lifecycle/sb_planning-accounts.
Accessed March 3, 2015.