KSU COURSE READER

By

KARTHICK KUMAR MALLI RAGHAVAN

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department Of Computing And Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2012

Approved by:

Major Professor
Dr.Gurdip Singh

Abstract

KSU Course Reader is an android application developed for Kansas State University students to access their course materials, listen to the audio lectures, watch the video lectures and read the lecture slides or other materials from all their enrolled courses with a single application. A student can add RSS xml link associated with the courses enrolled in the current semester in the application and can receive the course materials for each course automatically after every lecture.

The aim of this project is to develop a one stop android application for students to access the course materials of all their courses from their android smart phones and tablets. The project also provides a jsp form for course instructors to update the course materials for every lecture which automatically updates the xml file associated with the course. The application also allows students to add other RSS xml like K-State News, K-state Events etc. The same application can be used for adding public rss xml sites and can also be used as a podcast player.

It is very essential to provide the course materials readily available to the students all the time wherever they are. Since most students have smart phones or tablets and use them for accessing emails, surf internet etc , it becomes easier for them to use this KSU course reader application to access, read or view all their course materials at one place which is also customized for mobiles and tablets. As android is one of the leading and fastest growing smart phone platforms the project becomes more appropriate to develop in android.

Table of Contents

List of Figures v
List of Tables vi
Acknowledgements vii
Chapter 1 - Introduction 1
Chapter 2 - Requirements 1
  2.1. Requirements Analysis 1
  2.2. Software Requirements 2
  2.3. Hardware Requirements 2
Chapter 3 - Architecture and Design 2
  3.1. System Architecture 2
  3.2. System Design 3
    3.1.1. Use Case Diagram 3
    3.1.2. Class Diagram 5
    3.1.3. Data Flow Diagram 6
Chapter 4 - Android Components 8
  4.1 Android Manifest 8
  4.2 Activity 9
  4.3 Async Task 10
  4.4 Intent and Intent Filter 10
  4.5 SQLite 11
Chapter 5 - Implementation 11
  5.1. Android Module 12
    5.1.1. User Interface 12
    5.1.2. Splash Activity 13
    5.1.3. Channels Activity 14
    5.1.4. Parser Handler 16
    5.1.5. Navigation Activity 17
    5.1.6. Item Activity 18

5.1.7. Browser Activity 20
5.1.8. DBLayout and DBoperations 20
5.2. Web Module 21
5.2.1. JAXB Class Creation 21
5.2.2. JAXB UnMarshalling 21
5.2.3. Constructing XML tag elements from Web page 22
5.2.4. XML Marshalling 23
Chapter 6 - Testing & Logging 23
6.1 Unit Testing 23
6.2. Logging 26
Chapter 7 - Conclusion 26
Chapter 8 - References 27

List of Figures

Figure 3.1 System architecture 3
Figure 3.2 Use case Diagram for student 4
Figure 3.3 Use case Diagram for Course Instructor 5
Figure 3.4 Class Diagram 6
Figure 3.5 Data Flow Diagram 8
Figure 4.1 Android Manifest file 9
Figure 5.1 Splash screen 14
Figure 5.2 Default Channels Screen 16
Figure 5.3 Items under One Channel 18
Figure 5.4 Detailed Item View 19
Figure 5.5 Course Instructor Updating XML 23

List of Tables

Table 6.1 Test Cases.. 26

Acknowledgements

My special thanks to Dr. Gurdip Singh for giving me suggestions, encouragement, and guidance throughout the project.

I would also like to thank Dr. Xinming Ou and Dr. Torben Amtoft for graciously accepting to serve on my committee.

I would like to thank the administrative and technical support staff of the department of CIS for their support throughout my Masters program.

Chapter 1 - Introduction

The Project 'KSU Course Reader' is to develop a RSS course reader mobile application for android handsets and tablets. The main motivation of this project is to learn developing android application and understand the challenges involved in integrating server side technologies with android applications. To simplify the process of accessing course materials from android handsets and to develop an application that is useful for distance learning students is another motivation to develop this project.

This application is a proof of concept that it is possible to develop an application that provides features similar to KSOL in a more customized way for android mobile users. The project also proves that it is possible to develop similar applications in other leading mobile platforms like apple iOS and blackberry as well. The project is not just to implement an android application that could act as a course reader but also to develop a web page for course instructors to update the XML files for the courses in a simple way. This project also opens door to solve many other similar problem areas with a similar approach.

Chapter 2 - Requirements

2.1. Requirements Analysis

The project needed several materials to be gathered before designing and implementing it. For the server module to automate the xml generation from the web page, a study of JAXB plug-in and jar was needed. JAXB plug-in allows generating class files from xml schema definition. Once the preparatory materials for server side were gathered, the fields to represent the xml file for every course material and the tags for them were decided by comparing the rss version 2.0 schema definition.

For the application side, materials for debugging tools like adb were gathered and the set of features to support in the application were analyzed and documented. This document with list of features to support was later used for design and implementation phases and also to write the test cases for the possible usecases.

2.2. Software Requirements

Operating System: Android 2.2 or higher

Language: Android, Java, JSP

Database: SQLite

Technologies: Java, SQLite, RSS, SAX Parsing, XML, JAXB

Tools: Eclipse IDE, JAXB utilities, Tomcat Apache, Android Plug-in for Eclipse

Debugger: Android Device Bridge, DDMS

2.3. Hardware Requirements

- Samsung Galaxy Tablet Android 3.1
- HTC Tattoo Android 2.1

Chapter 3 - Architecture and Design

3.1. System Architecture

This section describes the architecture of this system. The diagram below shows the building blocks of the system architecture. The architecture contains both the server and client side modules. The blocks 'XML & Course Materials Server' and 'Course Webpage' are built as server side implementation. The apache tomcat server is run at the server to handle the requests from client android applications. The blocks SAX Parser, SQLite Database, Channels, Items, Detailed Item View are the blocks built as the client side implementation to be used in android.

The blocks Channels, Items and Detailed Item View act as the user interface module for android application. All these components closely interact with the SQLite database module whereas the Sax Parser interacts with SQLite database to save the parsed xml contents.

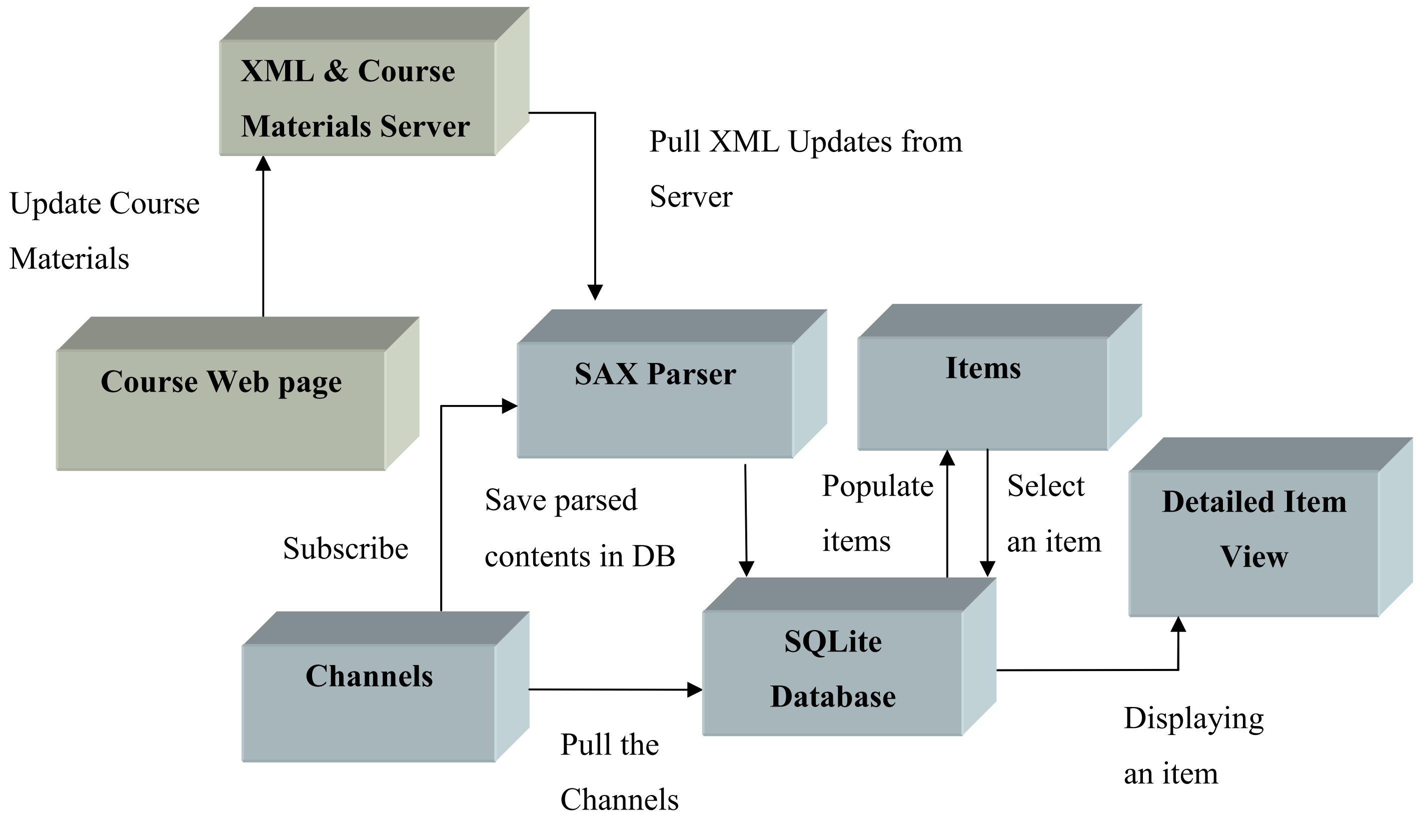


Figure 3.1 System architecture

3.2. System Design

Once the features to implement and the system architecture were drafted out the following design diagrams were necessary to implement the application. This section portrays the following three diagrams that were used in the later stages of the project to implement and test the application.

- Use Case Diagram
- Class Diagram
- Data Flow Diagram

3.1.1. Use Case Diagram

A Use Case diagram shows the actors and the roles they take in a system. It represents the actions performed by each actor. There are two actors for this project namely *Student* and the

Course Instructor. The diagrams below are the usecase diagrams for Student and course Instructor.

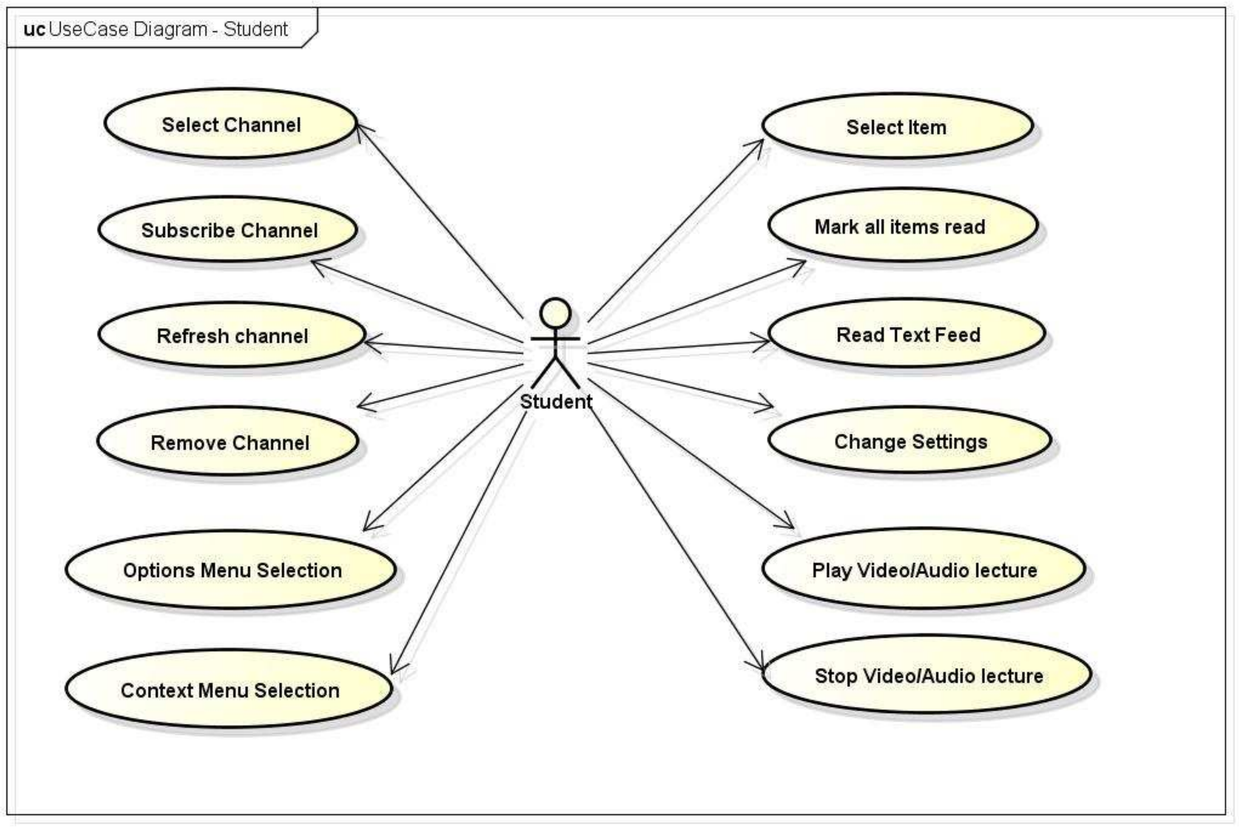


Figure 3.2 Use case Diagram for student

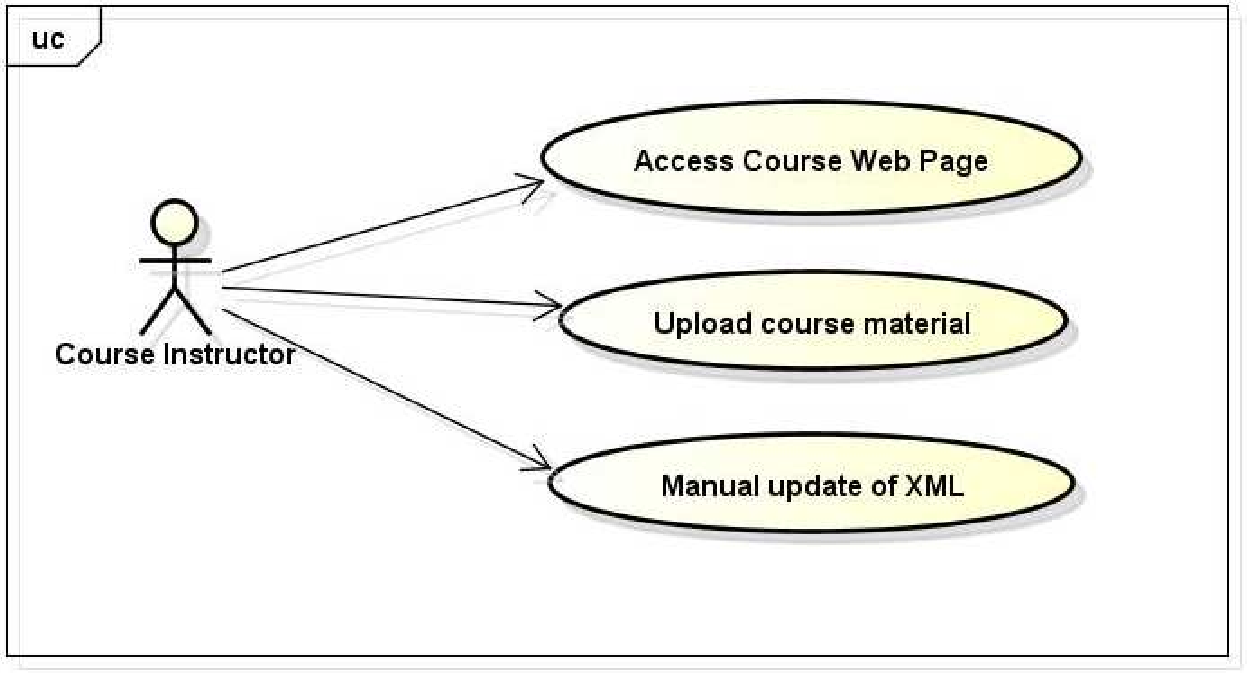


Figure 3.3 Use case Diagram for Course Instructor

3.1.2. Class Diagram

The class diagram below represents the important classes implemented for the application with their operations and return types. The diagram also shows how each class is associated with other classes in the system.

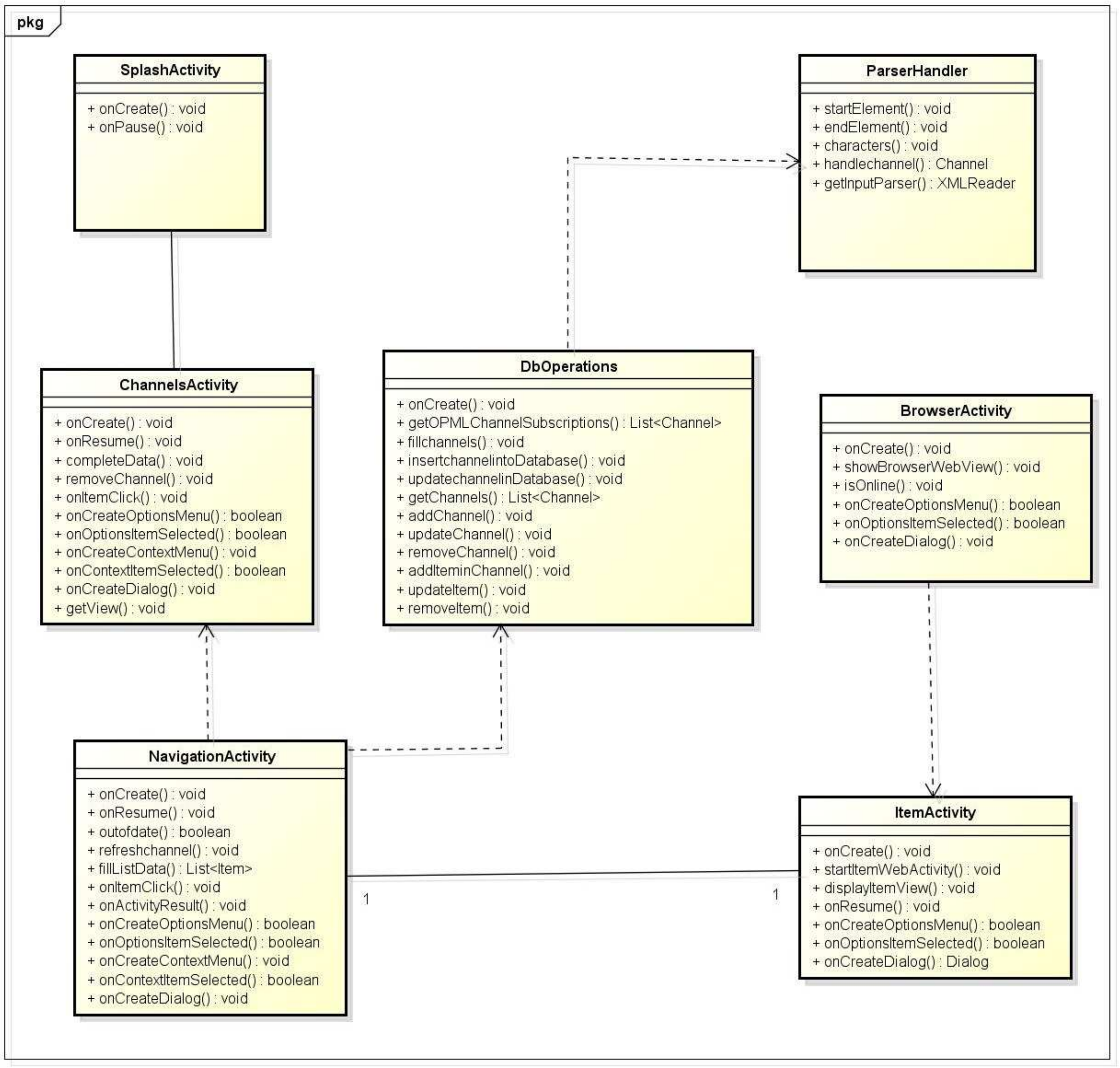
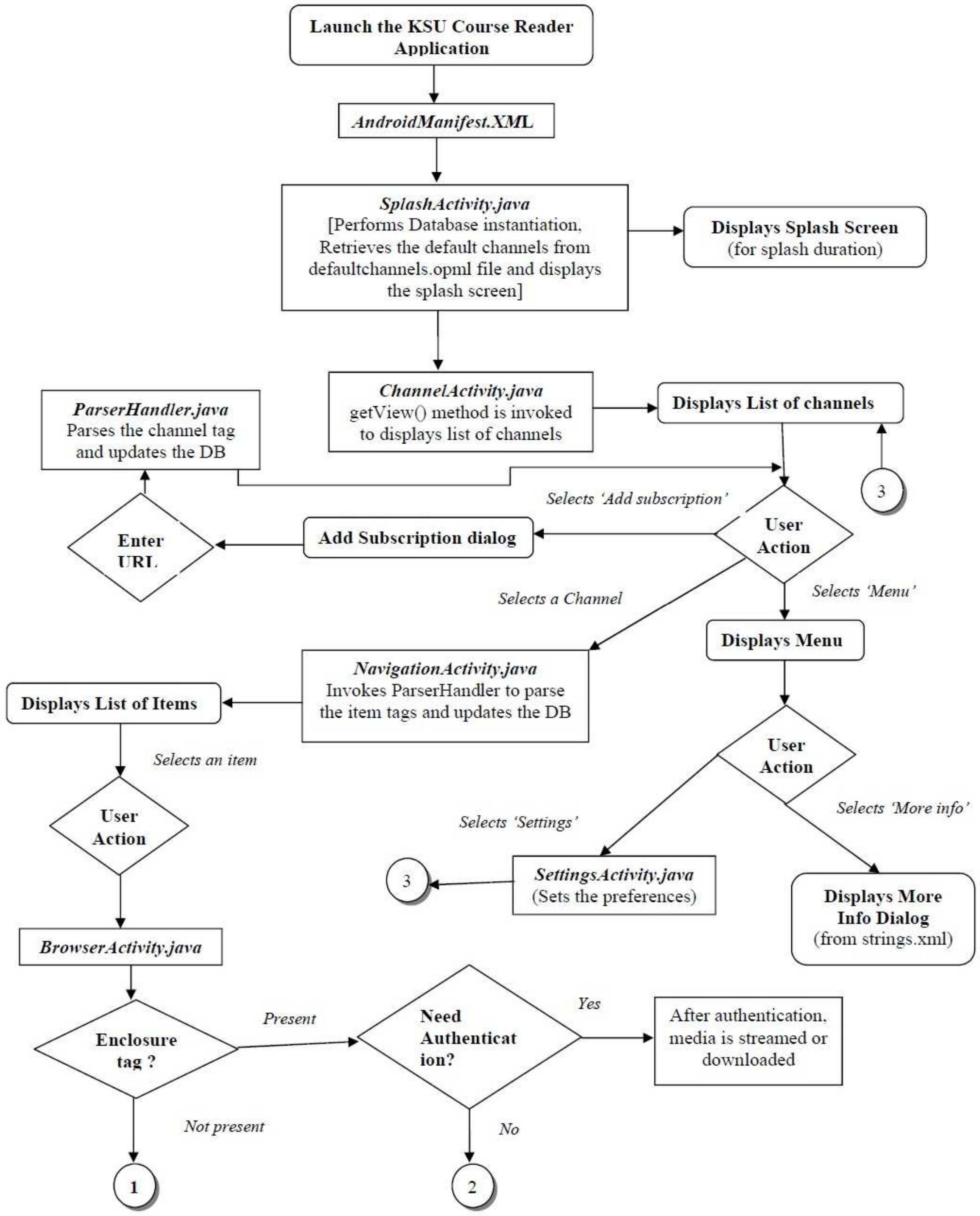

pkg
SplashActivity
+ onCreate() : void
+ onPause() : void
ParserHandler
+ startElement() : void
+ endElement() : void
+ characters() : void
+ handlechannel() : Channel
+ getInputParser() : XMLReader
ChannelsActivity
+ onCreate() : void
+ onResume() : void
+ completeData() : void
+ removeChannel() : void
+ onItemClick() : void
+ onCreateOptionsMenu() : boolean
+ onOptionsItemSelected() : boolean
+ onCreateContextMenu() : void
+ onContextItemSelected() : boolean
+ onCreateDialog() : void
+ getView() : void
DbOperations
+ onCreate() : void
+ getOPMLChannelSubscriptions() : List<Channel>
+ fillchannels() : void
+ insertchannelintoDatabase() : void
+ updatechannelinDatabase() : void
+ getChannels() : List<Channel>
+ addChannel() : void
+ updateChannel() : void
+ removeChannel() : void
+ addIteminChannel() : void
+ updateItem() : void
+ removeItem() : void
BrowserActivity
+ onCreate() : void
+ showBrowserWebView() : void
+ isOnline() : void
+ onCreateOptionsMenu() : boolean
+ onOptionsItemSelected() : boolean
+ onCreateDialog() : void
NavigationActivity
+ onCreate() : void
+ onResume() : void
+ outofdate() : boolean
+ refreshchannel() : void
+ fillListData() : List<Item>
+ onItemClick() : void
+ onActivityResult() : void
+ onCreateOptionsMenu() : boolean
+ onOptionsItemSelected() : boolean
+ onCreateContextMenu() : void
+ onContextItemSelected() : boolean
+ onCreateDialog() : void
1
1
ItemActivity
+ onCreate() : void
+ startItemWebActivity() : void
+ displayItemView() : void
+ onResume() : void
+ onCreateOptionsMenu() : boolean
+ onOptionsItemSelected() : boolean
+ onCreateDialog() : Dialog

Figure 3.4 Class Diagram

3.1.3. Data Flow Diagram

The data flow diagram below describes the flow of code in the application depending on the user action and represents the specific details of several possible data flow paths in the application. This diagram acts as a blue print for implementing the application.


Launch the KSU Course Reader Application
AndroidManifest.XML
SplashActivity.java [Performs Database instantiation, Retrieves the default channels from defaultchannels.opml file and displays the splash screen]
Displays Splash Screen (for splash duration)
ChannelActivity.java getView() method is invoked to displays list of channels
Displays List of channels
ParserHandler.java Parses the channel tag and updates the DB
3
Selects 'Add subscription'
User Action
Enter URL
Add Subscription dialog
Selects a Channel
Selects 'Menu'
Displays Menu
NavigationActivity.java Invokes ParserHandler to parse the item tags and updates the DB
Displays List of Items
Selects an item
User Action
User Action
Selects 'Settings'
Selects 'More info'
3
SettingsActivity.java (Sets the preferences)
Displays More Info Dialog (from strings.xml)
BrowserActivity.java
Present
Yes
Enclosure tag ?
Need Authenticat ion?
After authentication, media is streamed or downloaded
Not present
No
1
2

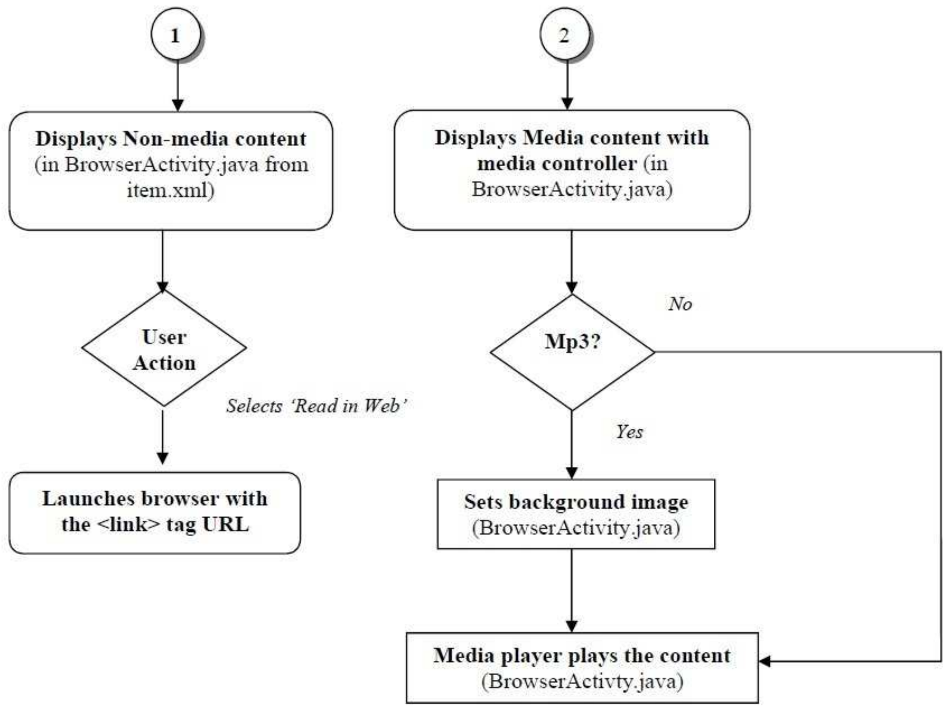

1
Displays Non-media content (in BrowserActivity.java from item.xml)
User Action
Selects 'Read in Web'
Launches browser with the <link> tag URL
2
Displays Media content with media controller (in BrowserActivity.java)
Mp3?
No
Yes
Sets background image (BrowserActivity.java)
Media player plays the content (BrowserActivty.java)

Figure 3.5 Data Flow Diagram

Chapter 4 - Android Components

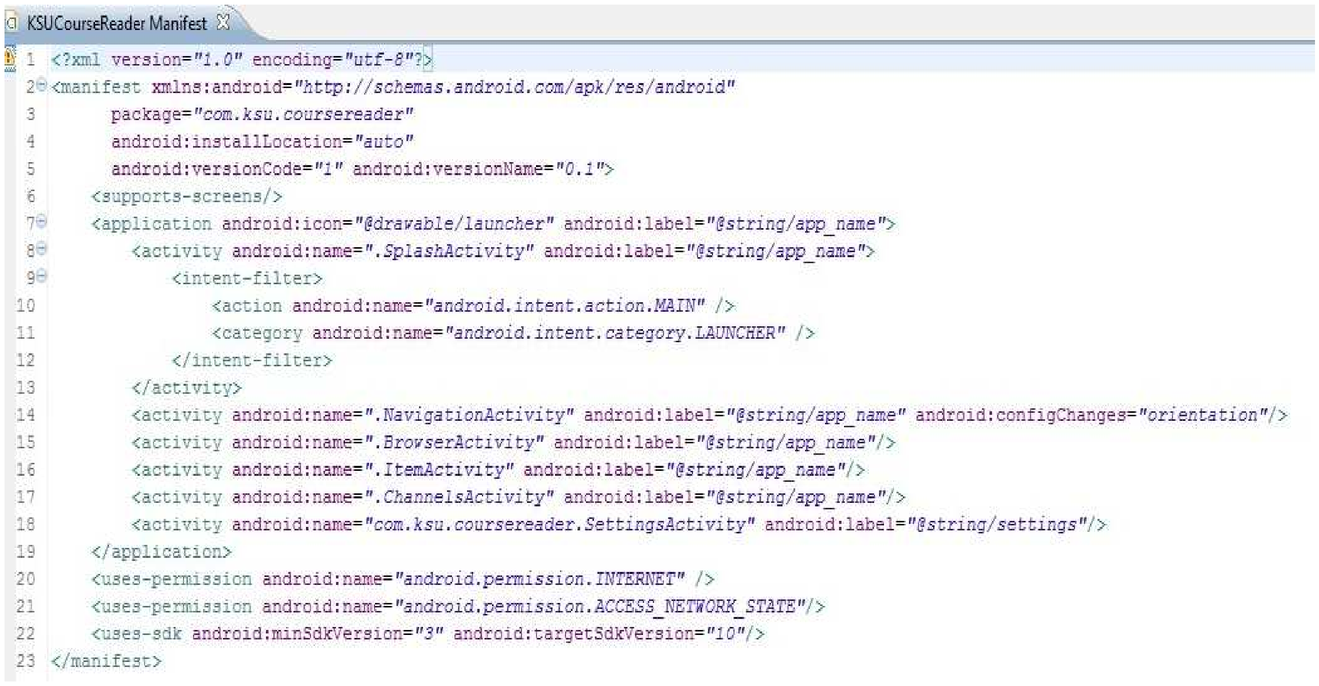
This chapter briefly discusses several android concepts and components that were used to implement the application.

4.1 Android Manifest

AndroidManifest.xml file is the starting point of any android application. When an android application is launched the android system looks for the application's androidmanifest.xml file. In specific, the android system looks for the first component that has to be loaded for the application. This component can be an activity or a service. Activities and services are later described in this chapter.(XML in Android)

AndroidManifest.xml file contains all android components that this application uses. This file specifies the list of activities, services, broadcast receivers, Content providers, intents and intent filers associated with each component and the set of permissions required to access each component in the application. An application's security is defined with the set of permissions listed in this file. The permissions listed here are the permissions that will be shown to the user to accept to proceed with the installation of application.

The snapshot below shows the KSU Course reader's android manifest file.



KSUCourseReader Manifest

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.ksu.coursereader"
      android:installLocation="auto"
      android:versionCode="1" android:versionName="0.1">
    <supports-screens/>
    <application android:icon="@drawable/launcher" android:label="@string/app_name">
        <activity android:name=".SplashActivity" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".NavigationActivity" android:label="@string/app_name" android:configChanges="orientation"/>
        <activity android:name=".BrowserActivity" android:label="@string/app_name"/>
        <activity android:name=".ItemActivity" android:label="@string/app_name"/>
        <activity android:name=".ChannelsActivity" android:label="@string/app_name"/>
        <activity android:name="com.ksu.coursereader.SettingsActivity" android:label="@string/settings"/>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="10"/>
</manifest>
```

Figure 4.1 Android Manifest file

4.2 Activity

Activity is the most used component in a user interface based android application. An activity component with the resource files represents the user interface in android. It provides a screen which users can interact to do something. The following five activities are developed for KSU course reader application. Each activity represents a user interface screen.

- Splash Activity
- Channels Activity
- Navigation Activity

- Item Activity
- Browser Activity

4.3 Async Task

AsyncTask enables proper and easy way of accessing the UI thread in android. A background operation should not block the user interface for more than few seconds in a mobile application. It is important that any task blocking the UI for several seconds have to be separated from the UI. Android works in two thread model where user interface is the main thread and the background task is the worker thread. Accessing User Interface thread from the worker thread is handled in android by the use of handlers. But when there are many instances of worker threads that access user interface, the handler gets complicated. Hence Android provides a mechanism called Async Task to manipulate the handlers.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. It is defined by 4 steps called onPreExecute, doInBackground, onProgressUpdate and onPostExecute. In this application AsyncTask is used to retrieve the contents from the xml and parse those in the background while the user interface shows the progress of the background process with a little progress image.

4.4 Intent and Intent Filter

Intent helps for communication between two android components. The communicating components can be in the same application or they can be in two different applications. The calling component should have the permissions to access the target component if the target component sets any permission. The component raising an intent can either specify a target component class directly or can broadcast the intent. If the intent was broadcasted, any component that has an intent filter matching one of the actions, data category of the intent can respond to the intent. If there are multiple components that can respond to that intent, they are shown to the user to select an appropriate or favorite component.

When an intent is raised specifically to a component it is termed as Explicit Intent. If the intent was broadcasted, it is termed as Implicit Intent. Action represents the general action to be performed and Data represents the information to act on when an intent is raised. The KSU reader application uses both explicit and implicit intents at several usecases of the application.

4.5 SQLite

A database operation in android is supported with a built-in SQlite database framework. Though SQLite does not support all the features of a normal SQL for computers, it provides the essential features to do any database operation for handsets and tablets. Android SDK provides API to work with SQLite and simplifies the database operations. As each android applications run in a separate sandbox, the database and tables created by one application is not accessible to other application. This ensures security of data stored in the databases. Data can be inserted, queried, removed or updated in the tables with simple queries. The components Content provider and content resolvers also provide mechanism to retrieve or insert data into the tables. (SQLite in Android)

The KSU Course reader application has the following three tables to perform database operations:

- Channel Schema
- Item Schema
- Enclosure Schema

Chapter 5 - Implementation

The implementation of this project is mostly in the android application. However the XML in the server corresponding to each course have to be updated by the course instructors. This project also provides a jsp web page for the lecturers to automatically update the xml file associated with the course they teach. Course Instructors can use this web page to update the xml page or even edit them manually to update after every lecture. This chapter discusses the implementation details of both the android module and the web module in detail.

5.1. Android Module

This section discusses the implementation details of several activities, other classes implemented and the user interface details of the application.(Developers Guide, 2011)

5.1.1. User Interface

Android separates designing the user interface from the code handling the user interface elements nicely. The modeling of several components within a screen is designed using xml. Android also allows dragging and dropping the text box, labels, buttons and other possible elements to generate the corresponding xml automatically. Android supports several kinds of layout for user interface like Linear layout, Relative layout, frame layout, table layout, etc. Our KSU course reader application mostly uses linear layout and in few places use relative layout. The xml user interfaces developed for this application are as follows:

Splashscreen.xml: This xml corresponds to the initial loading screen when this application is launched from the application tray. This user interface contains a text view to display the application title 'KSU Course Reader' followed by the logo image and another textview. This xml is set by the view of SplashscreenActivity in the code.

Channels.xml: The channels.xml file represents the screen that displays the list of channels in a linear layout format. This screen also contains a button 'Subscribe' to add a new channel to the list.

Item.xml: This xml file represents the user interface to display the text content of the feeds. It just displays the contents of description tag. There is also a button 'Read in Web' to read the same description from the web where this feed was posted. The read in web when pressed redirects the user to the url mentioned in the <link> tag. The detailed view of the item contains

Audio.xml: This represent the user interface showing how the screen looks like when an audio is listened from the application. This screen just contains an icon to show when the audio is being heard.

Settings.xml: This xml file show the list of items that must be shown to the user when the menu option ‘Settings’ is selected from channels screen or items screen. This xml contains 3 list preference elements to set or modify the time interval to update the feeds automatically, number of items to keep in each channel and the number of days to keep the items before removing them from the application.

There are also other user interface xml to show the popup dialogs for several usecases.

5.1.2. Splash Activity

This is the first activity that is launched when this application is launched from application tray. The android system loads AndroidManifest.xml to find the component that will be launched first by checking which activity has an intent filter matching ‘Main’ action and ‘launcher’ category. In this application Splash Activity contains the intent filter with ‘Main’ action and ‘Launcher’ category. This activity sets the view with splashscreen.xml. The splash activity is responsible for doing the following tasks in this application:

- Showing the initial loading screen
- Initiating the database connection
- Launching the Channels activity

When the database is initiated for the first time it loads each xml url in the defaultchannels.opml file and sends to the parser handler file to parse them. The parsed contents are stored in database.

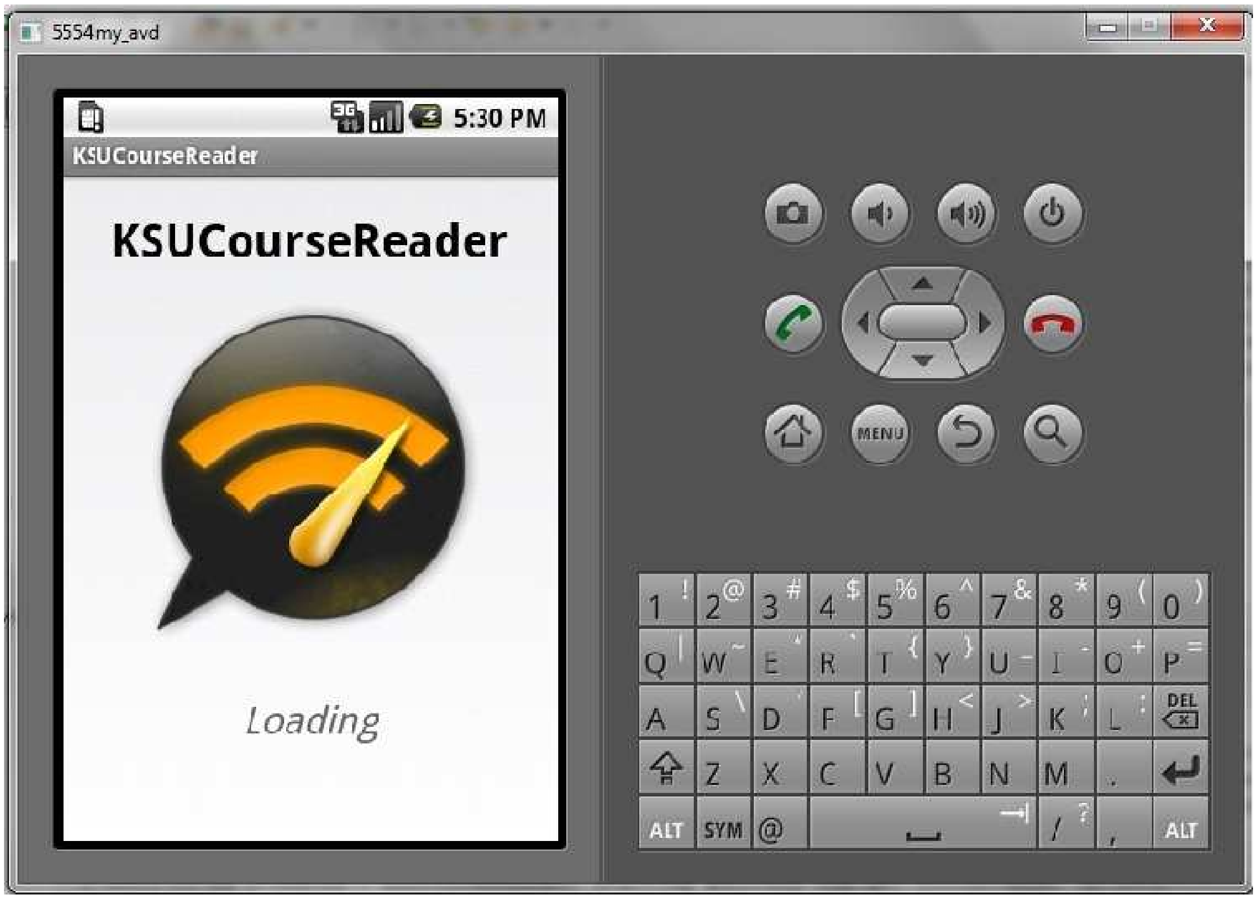


Figure 5.1 Splash screen

5.1.3. Channels Activity

The Channels Activity is initially launched by an intent from splash activity. The userinterface for this activity is set from Channels.xml. The default channels loaded from defaultchannels.opml are initially listed in the channels activity. However none of these channels have their items loaded yet. If all the items in each channel are loaded when this activity is launched for the first time, then it takes several seconds to give the control to the user. Hence to avoid this scenario, only the title of all the channels in the default channels are loaded initially and displayed in the channels activity.

The ‘subscribe’ button at the bottom of the screen is tied to an onClickListener in channels activity. When the subscribe button is clicked, it launches the Popup with Input text to enter the xml link of the channel the user wish to subscribe. Once the xml URL is captured, this activity passes this link to ParserHandler by instantiating the ParserHandler class.

If the user selects one of the channels listed in this activity and if the items of this channel have not been loaded yet then the channel's xml URL is passed to parser handler. If the items of the selected channel have been loaded already, then parser handler is not called but an intent is raised to Items activity to display the list of items in that channel. The channels activity also maintains the total number of items in each channel and keeps track of the number of items unread in each channel. This count is displayed adjacent to the channel title of each channel.

There are two context menu options set for the channels. Context menu is the list of menu items that are shown when an item is selected and pressed for few seconds. The context menus for each channel in this screen are "Mark all read" and "Remove Channel". When selecting 'Mark all' option, the unread count for that particular channel is set to 0 and the color of each item is changed to represent it as read content. When selecting 'Remove channel' all the items in that channel are removed from the user interface and the database.

The Options menu is the menu that is shown to the user when the menu hard key is selected from the handset or tablet. The two menu options for this screen are 'Settings' and 'About'. When 'Settings' is selected an intent is raised to launch Settings Activity and Channels activity is pushed to the back stack of the task automatically. When 'About' option is selected, a popup is displayed to show what this application is for.

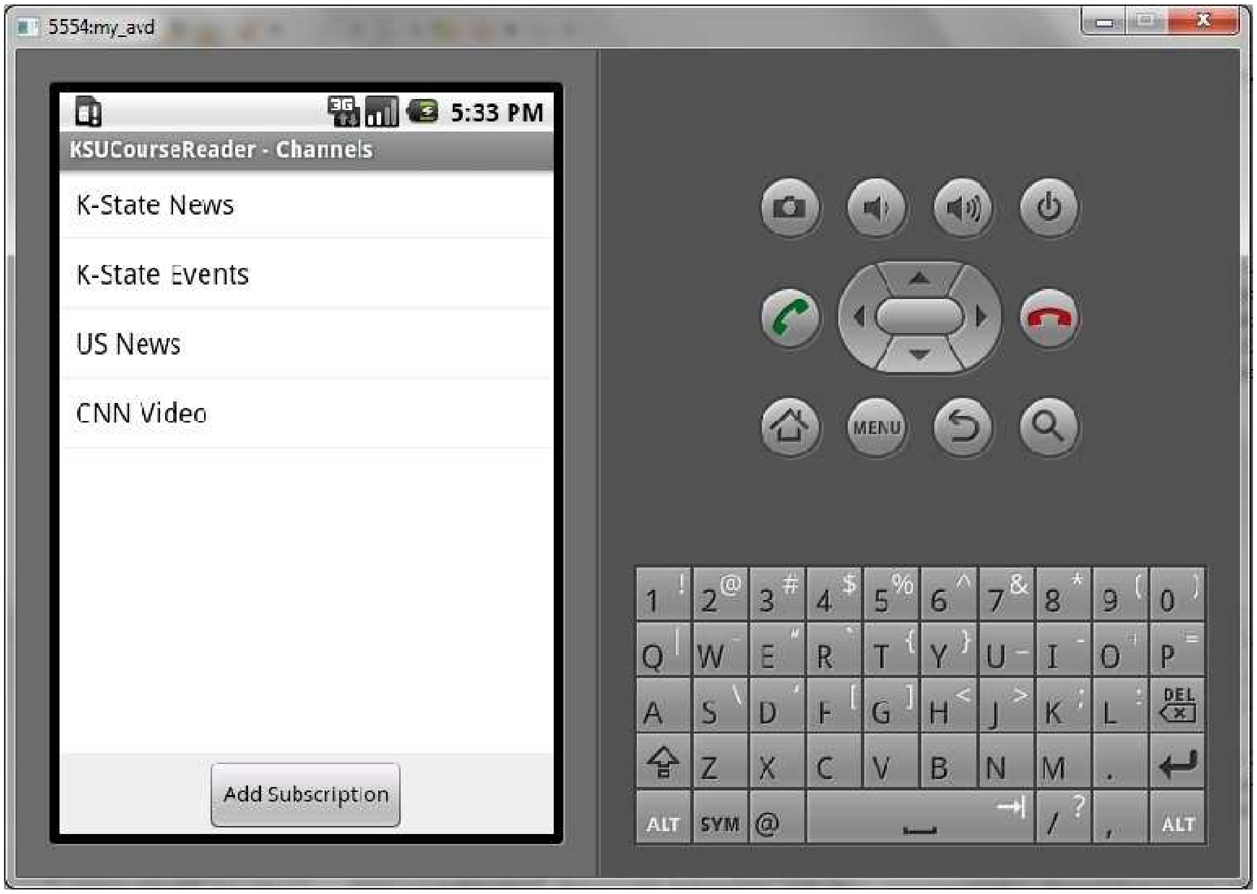


Figure 5.2 Default Channels Screen

5.1.4. Parser Handler

The Parser handler class does an important task of parsing the contents of each channel subscribed in the application. Parser handler class uses a built in Simple Api for Xml parser (SAX Parser) to parse each tag in the xml file. SAX parser was preferred over DOM parser in the application because DOM parser needs to have the entire xml file in memory to parse even a single tag from the xml file. As this is a course reader application, size of the xml would grow every week while the course instructor keeps updating the xml file. Hence there is a chance of performance degradation of the application. On the other hand, SAX parser is an event based parser and it doesn't require memory to have the whole xml while parsing. The SAX parser scans the file and for every matching token it performs some action. It can either trigger some events or can save the contents between every matching tag to some data structure for some processing later or even ignore the matched token depending on the logic. SAX parser contains 4 methods that need to be overridden for doing some useful processing of xml data. Those 4 methods are:(XML in Android)

- startDocument
- endDocument
- startElement
- endElement

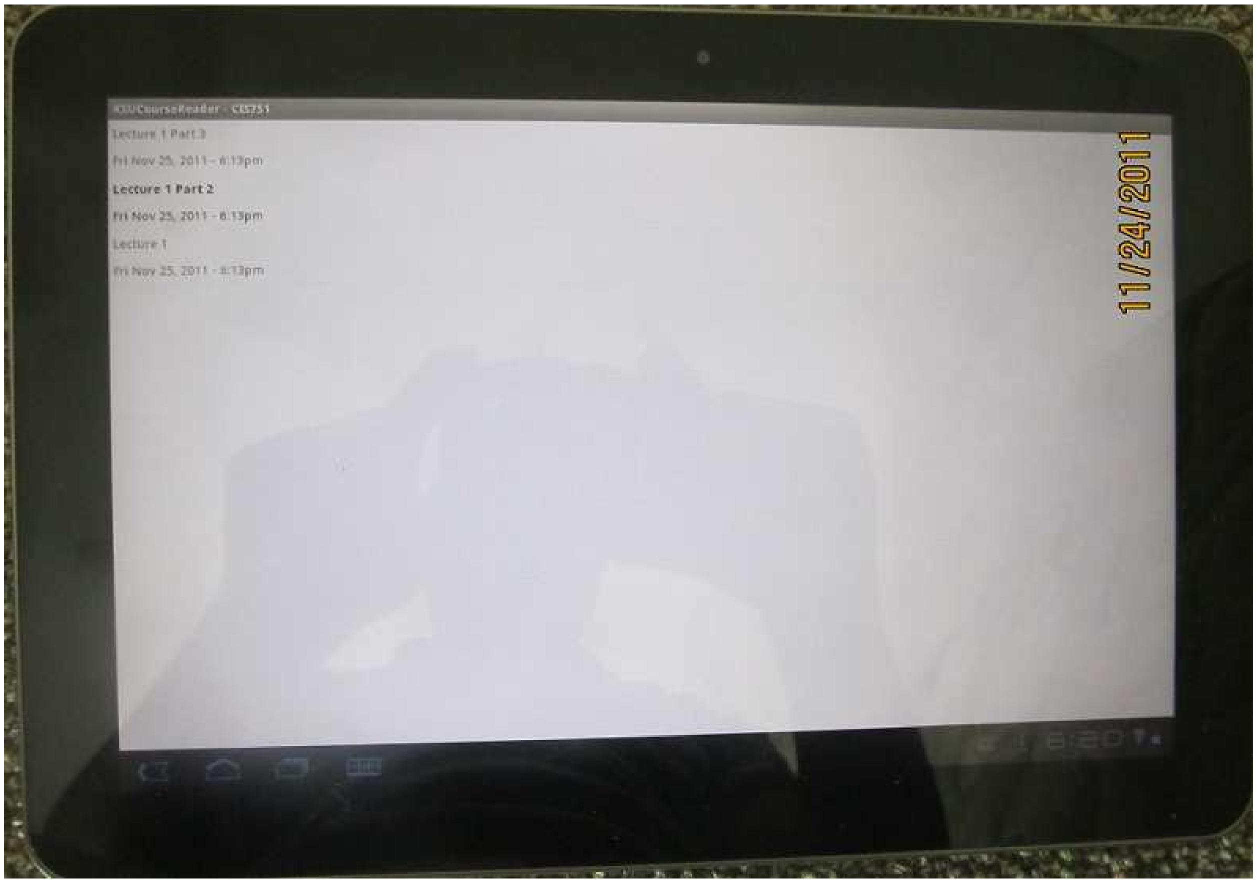
In our course reader application startElement and endElement matches every open tag and close tags and saves the content between each tag in a predefined class object. For example the presence of <enclosure> tag represents that the content is of media type and the URL content between the open and close tags of an enclosure correspond to the link of the media file. This particular link will be saved in the constructed object and will also be saved in the enclosure schema of this application's database.

5.1.5. Navigation Activity

Navigation Activity is an activity that acts as a bridge for Channel activity and Item activity. When a channel is clicked from the channels activity, an intent is passed to navigation activity with the channel name. The onResume method of navigation activity calls refreshChannel method. This activity checks if the Internet connection is available or not. When there is no internet connection at that moment before pulling the data from xml, it pops up with a messagebox saying that internet connection is not available.

If the channel for which this intent was raised has not loaded any items, navigation activity starts an Async task to pull the items from the xml file. Here the navigation activity works with Parser handler in parsing and saving the data to the database table and the object. The 'Item Schema' gets updated after the async task is complete. The Aysnc task publishes to UI the progress of downloading the content that is parsed in the background. Once the async task executes its onPostExecute method, the control comes back to navigation activity. Navigation activity then updates the UI with the parsed content and displays the list of items in each channel of this screen.

The navigation activity provides an onClick method which implements what to happen when an item displayed in this activity is selected. Here it actually raises an intent to Item activity by passing the item id. The navigation activity applies itemnotselected.xml to set the content view and itemselected.xml when an item is selected. Navigation activity screen contains two options in its menu namely ‘refresh’, ‘Channels’ and ‘Settings’. Selecting ‘refresh’ option updates the user interface with the updated items. Selecting Channels takes back to the Channels activity whereas selecting settings takes to Settings Activity with the help of intents.



KSUCourseReader - CIS751

Lecture 1 Part 3

Fri Nov 25, 2011 - 6:13pm

Lecture 1 Part 2

Fri Nov 25, 2011 - 6:13pm

Lecture 1

Fri Nov 25, 2011 - 6:13pm

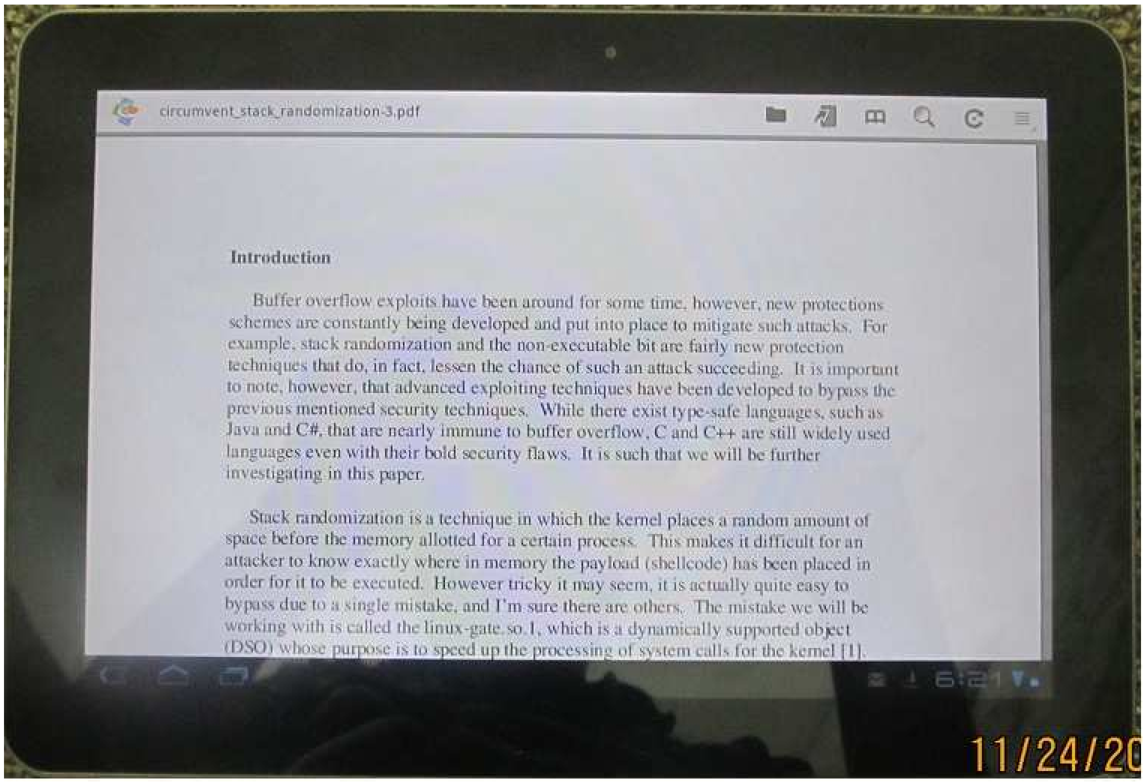
Figure 5.3 Items under One Channel

5.1.6. Item Activity

Item activity is launched when an item is selected from the navigation activity screen. As soon as the item activity is launched, the color field associated with the item from which the intent was launched is changed to represent that it is read. The unread/total count of items in that particular channel is also updated. Then, from the database, item activity checks if there is an Enclosure field associated with this item in Enclosure schema.

If there was a non empty enclosure field then it is of media type and if there were no enclosure tags associated with that item, it is of text type. If the item is of media type then an intent is passed to Browser Activity with the URL within the enclosure tag. If the item is of text type, then the detailed item view of that item is displayed by setting the item.xml with a button at the bottom 'Read Online'. The idea is to show the content within the ,description> tag in this detailed view and take the user to the url within <link> tag when 'Read Online' button was clicked with an intent to Browser activity.

There are four menu options available in this activity screen namely 'Channels', 'Settings', 'Home' and 'About'. 'Channels' take to channels screen , settings take to settings activity, 'Home' takes to the Navigation activity to show the list of items within the channel associated with the current item and 'About' shows a popup to describe the purpose of this application.



Introduction

Buffer overflow exploits have been around for some time, however, new protections schemes are constantly being developed and put into place to mitigate such attacks. For example, stack randomization and the non-executable bit are fairly new protection techniques that do, in fact, lessen the chance of such an attack succeeding. It is important to note, however, that advanced exploiting techniques have been developed to bypass the previous mentioned security techniques. While there exist type-safe languages, such as Java and C#, that are nearly immune to buffer overflow, C and C++ are still widely used languages even with their bold security flaws. It is such that we will be further investigating in this paper.

Stack randomization is a technique in which the kernel places a random amount of space before the memory allotted for a certain process. This makes it difficult for an attacker to know exactly where in memory the payload (shellcode) has been placed in order for it to be executed. However tricky it may seem, it is actually quite easy to bypass due to a single mistake, and I'm sure there are others. The mistake we will be working with is called the linux-gate.so.1, which is a dynamically supported object (DSO) whose purpose is to speed up the processing of system calls for the kernel [1].

Figure 5.4 Detailed Item View

5.1.7. Browser Activity

Browser activity is launched from the Item activity for both the media and non-media types. The intent to launch the browser from item activity is an implicit intent. So the application shows all the browsers that can handle this intent. The user will be allowed to choose the browser of their choice. They can also set a default browser that needs to be launched automatically every time when similar intents are passed by some components within the application.

If the URL that was launched needs some authentication, the login screen will be prompted to the user. If the content is of media type (video, audio, image, flash, ppt, etc) the browser decides which is the best component to play that media. If it is of video or audio type, then media players installed within the android handset or tablet will be invoked to stream them and play. If streaming is not supported the browser attempts to download the file and open it. If the item is not of video or audio type, then browser can play those media and show to the user in the user interface.

5.1.8. DBLayout and DBoperations

The KSU course reader application needs a database to save the parsed data from each channel subscribed by the student using the application. Following are the three database schema that this application creates when the application is launched first after installing.

- Channel schema
- Item Schema
- Enclosure Schema

Channel schema contains the fields URL, title, type, refresh fields.
Item schema contains channel_id, link, title, description, content, image, pubdate, read fields.
Enclosure schema contains item_id, mime and URL.

DBOperations class provides methods to insert, update, and delete entries into the schema mentioned above from various activities within the application. (Developers Guide, 2011)

5.2. Web Module

This module discusses the server side implementation of this project. The android application subscribes the xml links associated with each course in their application. The xml could be manually updated by the instructor after every lecture. But to automate the xml update with minimal effort this module has been implemented for the use by course instructors. Each course will have an associated web page where the course instructor can update the materials and links after lecture. From this web page the xml page of this course will be automatically updated. Hence any student enrolled with the course and using KSU course reader application can refresh the associated channel manually or by auto update option in android application to receive the updated course material after every lecture. This module briefly discusses this automatic xml generation from course webpage.

5.2.1. JAXB Class Creation

Java Architecture for XML Binding allows Java classes to XML representation and vice versa. The JAXB Plug-in for eclipse is needed for this module and the JAXB jar is also placed in the library of this project. The xml files that we have used in this KSU course reader are based on RSS version 2.0 and hence the XML schema definition for RSS version 2.0 is downloaded and kept in the project folder. This rss-2.0.xsd file is the representation of the rss version 2.0. It contains all possible xml tags in rss version 2.0 and their format within the xml file.

As the JAXB plug-in for eclipse has been downloaded, class representation of this XSD file can be generated by Selecting 'Generate JAXB class' option in the JAXB plug-in for this xsd.
These classes are generated just once from this xsd unless there are some updates to the xsd or if we upgrade or downgrade our rss version for the xml files.(JAXB Reference)

5.2.2. JAXB UnMarshalling

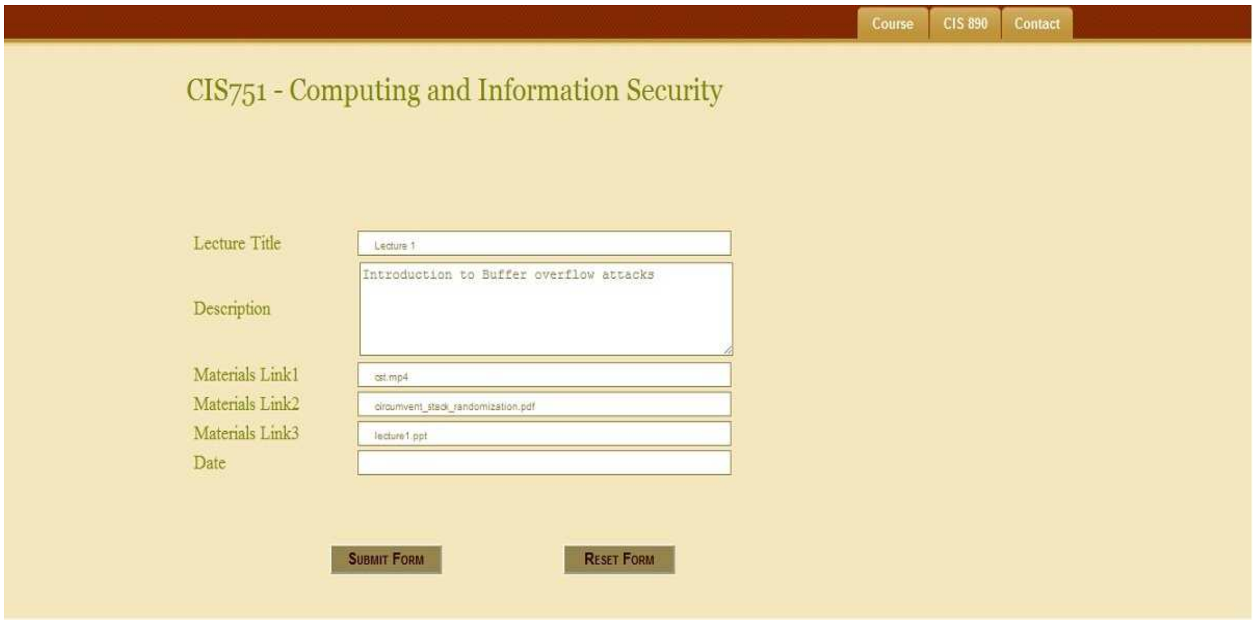
Automating the XML update is easier if the xml file is represented as a class and fields. This is because if each tag is represented as a class and the attributes of the tags are represented as fields in a class, it becomes simpler to instantiate each class when we need to introduce similar tag with associated attributes and contents in the xml. For this purpose JAXB API provides *unmarshall* method which helps to convert an XML file to object representation. The *unmarshall* method works with the classes that were generated in 4.2.1 and with this xml to convert to its object representation.

5.2.3. Constructing XML tag elements from Web page

JAXB jar provides API to add and remove instances of each type of tag to the Object representation of unmarshalled xml file. The web page for a course as in the screenshot below allows the course instructor to update the materials for the lecture from the web page. For every submission from this link one <item> tag is created. The title field in the web page corresponds to the <title> tag of the item. The description field in the web page represents the <description> tag of item. The material link where a course instructor can paste the url link of the lecture slide or video lecture or audio file represent the <enclosure> tag of item.

A course instructor can attach more than one material for every submission. In this case, for keeping it simple, if there are more than one material link attached in the web page separate items are generated for each material link as <enclosure> and the title appended with the part number. For example, if there are 3 materials attached, then 3 <item> tags are generated with title tags appended with part1 part2 and part3 respectively for each item and each item with one <enclosure> tag.

Once these item tags are generated from the course web page, each item tags are prepended to the existing object representation of the xml. This is to make sure that a new material updated recently shows as the first feed in the course channel of the android application when refreshed. After all these items are prepended, a single object representing the xml with new contents is available.



CIS751 - Computing and Information Security

Lecture Title	Lecture 1
Description	Introduction to Buffer overflow attacks
Materials Link1	cst.mp4
Materials Link2	circumvent_stack_randomization.pdf
Materials Link3	lecture1.ppt
Date	

Figure 5.5 Course Instructor Updating XML

5.2.4. XML Marshalling

The modified xml with the recent updates in the object representation can be converted back to XML format using JAXB *marshall* method. The xml generated can now replace the old xml at same location in the server.

The same process in 4.2.3 and 4.2.4 are repeated automatically when the course instructor updates new material from the web page. The course instructors can also update the xml file manually if they prefer to.

Chapter 6 - Testing & Logging

6.1 Unit Testing

In Unit testing every usecase possible in the application is tested as an independent entity to verify its correctness. All the unit testcases were tested in Emulator and real android devices. The application is tested with the following devices to ensure the compatibility and uniformity across different versions.

- Android HTC Tattoo OS Version 2.2
- Samsung Galaxy tablet Android OS version 3.1
- Android Emulator simulating OS versions 2.3.4 and 2.1

The table below shows all the test cases that were executed to test the application.

S.No	Test Case	Expected Behavior	Result
1	Launch the application from application tray	The splash screen should be shown for 3 seconds	Pass
2	Launch the splash screen and wait for 3 seconds	Default channels should be displayed	Pass
3	Select any of the available default channels	Channel must load with the items	Pass
4	Go to channels screen and add one of the xml of enrolled courses	The course feeds are loaded and the list of items are shown	Pass
5	Go to channels screen and hit menu button	Settings and About options are shown	Pass
6	Select settings menu from channels screen	settings activity is loaded with Update items periodically option, Remove items option and No. of items to keep for each channel	Pass
7	Change the No. of items for each channel option in settings screen	Every channel is updated to keep at most the no.of items set in the settings activity	Pass
8	Change update periodically from once a day to every one hour	Once the channel is updated with few items in the server , the application must be updated automatically updated with new items for every channel	Pass
9	Change Update periodically option in settings to Update manually	Auto update of items must be stopped and the items are refreshed only after selecting Update option manually from Menu	Pass
10	Change Remove items option from 'Never' to older than 1 week	The items older than 1 week must be removed from each channel	Pass
11	Select 'About' option in Menu from channels screen	A popup is shown describing what the application is all about	Pass
12	Check Unread/Total items count for a channel that is already loaded	Before any item is read from a loaded channel unread and total count should be the same	Pass
13	Check Unread/Total count for a channel that is not yet loaded in channels screen	The unread/Total count field must not be displayed and should be empty	Pass
14	Read an item from one of the channel having unread/total values field	Unread count is reduced and color of the item is changed	Pass
15	Select 'Mark all Read' option from context menu of a channel	Color of all items of the channel is changed and unread count must change to 0	Pass
16	Select 'Remove channel' option from context menu of a channel	The channel and its contents are removed from the display	Pass
17	Select 'Remove channel' option when there is only one channel in the application	The application doesn't remove the channel and pops up a message 'Sorry, the app requires atleast one channel. We cannot remove it'	Pass
18	Select a video item from one of the channel items	Browser is launched and favorite media player is asked to be chosen by the user and the video is played	Pass
19	Select an audio item from one of the channel items	Media player is launched and the background is set with an image	Pass

20	Select a Text feed from one of the channel items	The text feed is displayed properly and 'Read Online' button is shown	Pass
21	Select 'Read Online' button for a text feed	Browser is launched and the user is taken to the url where this text feed has been posted.	Pass
22	Add 3 material links in the course web page for one submission	Three individual feeds are added with one course material each with the same title appended with Part numbers	Pass

Table 6.1 Test Cases

6.2. Logging

Android SDK has a built in tool called Android Device Bridge (adb) that helps to check the logs and check user defined logs. ADB toolkit has several commands to monitor the system logs. It also provides a sqlite3 tool to display the entries of all the tables in the application's database and execute queries. ADB tool kit works well with both emulator and real devices. The devices need to be connected with the system via USB cable to read the logs in adb shell. Android also comes with DDMS tool that would display android system and application logs in the eclipse IDE.(Developers Guide, 2011)

Chapter 7 - Conclusion

The project attempts to prove how the KSU course access reader kind of applications can be used to simplify accessing course materials for students. The application can also become a podcast player when the students using this application subscribe to podcast feed sites. This application development has given me an opportunity to understand the challenges and limitations in developing an android application and has also taught how server side technologies can be integrated with mobile applications. The project also bolsters the fact that it is possible to add more values to the software when it is designed well before implementation. Same application can also be developed in iOS for iPhones and iPads.

The application can be improved in several ways and can be extended or modified to solve many similar problem areas. Below are some of the possible ways of improving the application or modifying them for solving other problems:

- The application can be extended to be used by course instructors as well to update the course materials from their android handsets or tablets.

- The application can be extended to send emails to course instructors or to start a discussion by adding more social values.

- It is possible to modify the application to receive the course update alerts in the form of SMS messages or a WAP Push message alerts.

- The application can also support adding social networking feeds from twitter or facebook etc.

Chapter 8 - References

Developers Guide. (2011). Retrieved September 2011, from Android Developers: http://developer.android.com/guide/index.html

Hipp, R. (2011). *sqlite*. Retrieved September 2011, from documentation: http://www.sqlite.org/docs.html

JAXB Reference. (n.d.). Retrieved September 2011, from http://www.oracle.com/technetwork/articles/javase/index-140168.html#xmp1.

Reilly, O. (n.d.). *Android Cookbook*. Retrieved 2011, from http://androidcookbook.com/home.seam.

SQLite in Android. (n.d.). Retrieved from http://www.screaming-penguin.com/node/7742.

XML in Android. (n.d.). Retrieved from http://www.ibm.com/developerworks/opensource/library/x-android/.