NORTHWEST NAZARENE UNIVERSITY

A Mobile Application for John Wesley's Explanatory Notes on the Bible.

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF ARTS

Curtis Carpenter

2013

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF ARTS

Curtis Carpenter
2013

A Mobile Application for John Wesley's Explanatory Notes on the Bible.

Author: ____
Curtis Carpenter

Approved: ____
Barry Myers, Ph.D., Professor of Computer Science,
Department of Mathematics and Computer Science, Faculty Advisor

Approved: EDWIN H ROBINSON
Ed Robinson, Ph.D., Professor of Practical Theology
Department of Theology, Second Reader

Approved: ____
Barry Myers, Ph.D., Chair,
Department of Mathematics & Computer Science

ABSTRACT

A Wesley Commentary Mobile Application.

CARPENTER, CURTIS (Department of Computer Science), MYERS, DR. BARRY L. (Department of Computer Science).

A mobile application that provides access to Wesley's explanatory notes on the Biblewas created on the android operating system. This application gives users access to the same Wesley commentaries that are currently available online at the Wesley Center website. The application was created under the Eclipse development environment and uses an integrated SQLite database to store and retrieve data. A combination of ScrollViews within a ViewPager provides the user the ability to view multiple commentaries in an efficient and easy way. The framework used for searching is the (FTS3) Full Text Search within SQLite, but is currently not fully implemented in the application. With this application, the user will be able to access Wesley commentaries any time, without an internet connection.

Acknowledgments

First of all, I would like to thank my beautiful wife Tori for all of her support throughout this project. Without her, I am not sure I would have made as much progress as I did. I would also like to thankMatt Rankin for his constant willingness to discuss project ideas and programming techniques. The fact that we were both working on an android project at the same time was a great opportunity and we took full advantage of that. Lastly, I would like to thank my professors for their support before and during development.

Table of Contents

List of Figures

Overview

The primary goal of this project was to create a releasable mobile application that provides access to Wesley Center content, specifically Wesley's explanatory notes that are currently only available online at the NNU Wesley website, http://wesley.nnu.edu/. The purpose of this application is to be a tool that allows for easy access to Wesley commentaries in a format other than through the web. Users are able to view these biblical commentaries directly from their mobile devices. This application was developed under the android programming environment and could possibly be ported to iOS in the future.

Background

This project was a continuation of a previous senior's project that was partially developed last year. Although there had been some work done, the project was still mostly just an idea. The goal was to take database work done by Josh Ehrlin back in 2011, and use it for a mobile application. Josh had converted Wesley's explanatory notes from a simple text into a usable SQL database. Originally, this new application was supposed to pull content directly from the Wesley website, but instead of doing this, itwas created to be completely standalone and not need internet access to view the Wesley content. This gives the application more flexibility and increases the performance significantly.

Josh had stated in his thesis that this content might be used better as a standalone package on a personal computer or a mobile device. By doing this, the problem that might occur with having a large number of users sending requests to the server hosting our database would be removed. Having a localized application means a great deal of variability that comes with pulling information off the web can be removed.

When deciding on what to do for a senior project, doing something with the android operating system and mobile development was very important.The application had to be something that could actually be used in the future and that had components that would be challenging to develop. When the idea came up about this Wesley project that people had been working on for a couple years, it definitely grabbed my attention because it fit all the criteria of what this project needed to be.

Exploration

Before any design or coding could be done, the first priority was to determine what had already been done on this project and where the starting point was. Two seniors had worked on this already, sothere was no point in wasting time working on something that had already been completed. The assumption when starting this project was that the database was complete and that there was a decent amount of progress already made with the android application. Although the database was completed, the application had never gotten off the ground. The project would have to start at square one, which pushed the schedule back quite a bit. A huge part of this project would be to get the existing database working within the android environment. Android uses a form of SQL called SQLite, which is basically a lightweight version of SQL. The SQL file from Josh Erhlinwould work, but a method to parse that file and build the SQLite database within the application would need to be developed.

Creating a framework that allowed searching was also one the main goals of this project. Fortunately, SQLite comes with a FTS3 extension that allows for extremely fast full text searching. This is done through the use of special tables with a built in full text index. Because this application is being developed for a mobile device, making sure that the application is as efficient as possible is extremely important. In relation to working on an application that runs on a server or a personal computer, the amount of power we have at our disposal is much less.

Design

The most important part of the initial design was how the existing database was going to be integrated into the mobile application. The database was split into three main tables: books, chapters, and notes. The books table having an ID, a title, and text about the book. The chapters table contained a bookID that it was related to, a chapterID, and the text for that chapter. The notes table contained a bookID, chapterID, and a noteID, along with the text for that note.

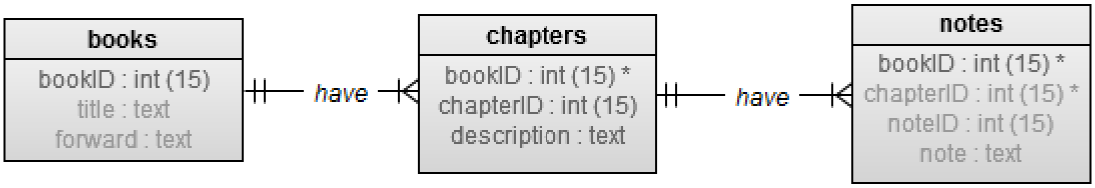
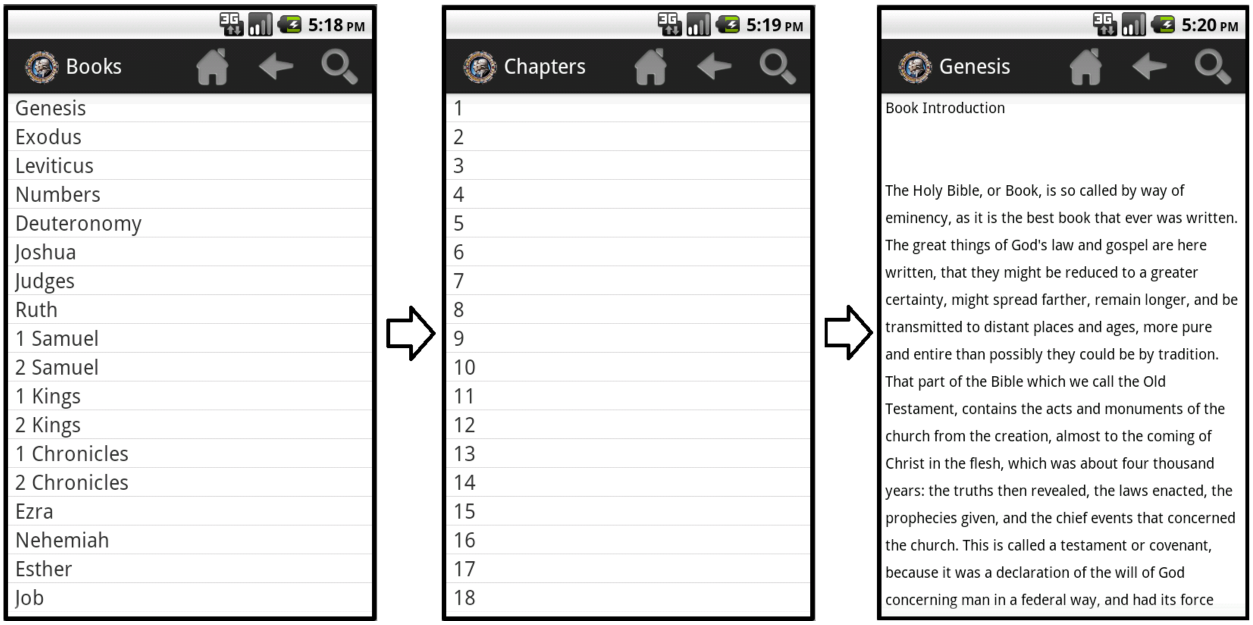


Figure 01 - Entity Relationship Diagram

The text within the books table was information about a specific book of the Bible, the text within the chapters table was information about each chapter, and the text in the notes tables contained the actual explanatory notes for a given verse. With this simple structure, the application could pull any information needed from the database.

The second part that needed to be designed would be the layout. This took much longer than expected because determining the organization of the explanatory notes was very difficult. It was eventually decidedto have a book and then chapter selection page. After selecting a chapter, the user could view each note horizontally by verse.



Genesis
Exodus
Leviticus
Numbers
Deuteronomy
Joshua
Judges
Ruth
1 Samuel
2 Samuel
1 Kings
2 Kings
1 Chronicles
2 Chronicles
Ezra
Nehemiah
Esther
Job

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

Book Introduction

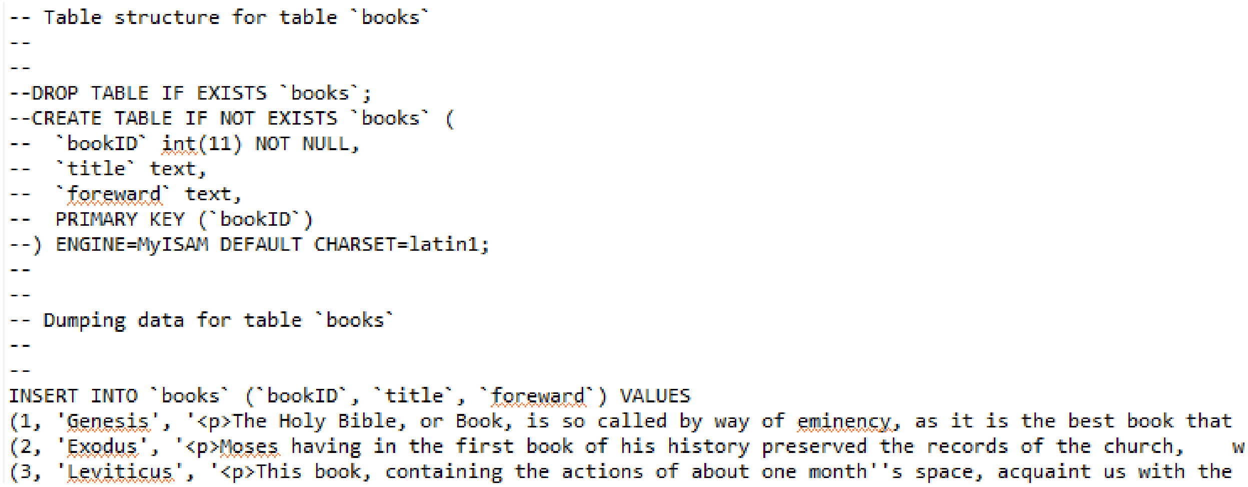
The Holy Bible, or Book, is so called by way of eminency, as it is the best book that ever was written. The great things of God's law and gospel are here written, that they might be reduced to a greater certainty, might spread farther, remain longer, and be transmitted to distant places and ages, more pure and entire than possibly they could be by tradition. That part of the Bible which we call the Old Testament, contains the acts and monuments of the church from the creation, almost to the coming of Christ in the flesh, which was about four thousand years: the truths then revealed, the laws enacted, the prophecies given, and the chief events that concerned the church. This is called a testament or covenant, because it was a declaration of the will of God concerning man in a federal way, and had its force

Figure 02 - Flow Chart

This structure gives the user access to exactly which note they are looking for quickly. After choosing a chapter, the user can then scroll horizontally to view each verse. For longer verses a vertical ScrollView is used. This in combination with the horizontal paging was very difficult to implement but paid off in the end because it gives the user a huge amount of flexibility.

Implementation

The first part of the actual implementation was creating the SQLite database within the application. The database that was given was in a .SQL file that contained the code to create the database. The problem was that it was formatted much differently than what it needed to be. The way this problem was solved was with the use of a BufferedReader to read in each line from the file and then create SQL strings which could then be executed. So what had started as a huge amount of text, Figure 03, ended up as a useable SQLite database with FTS3 search capabilities, Figure 04.



```
-- Table structure for table `books`
--
--
--DROP TABLE IF EXISTS `books`;
--CREATE TABLE IF NOT EXISTS `books` (
--  `bookID` int(11) NOT NULL,
--  `title` text,
--  `foreward` text,
--  PRIMARY KEY (`bookID`)
--) ENGINE=MyISAM DEFAULT CHARSET=latin1;
--
--
-- Dumping data for table `books`
--
--
INSERT INTO `books` (`bookID`, `title`, `foreward`) VALUES
(1, 'Genesis', '<p>The Holy Bible, or Book, is so called by way of eminency, as it is the best book that
(2, 'Exodus', '<p>Moses having in the first book of his history preserved the records of the church,    w
(3, 'Leviticus', '<p>This book, containing the actions of about one month''s space, acquaint us with the
```

Figure 03 - SQL Dump File

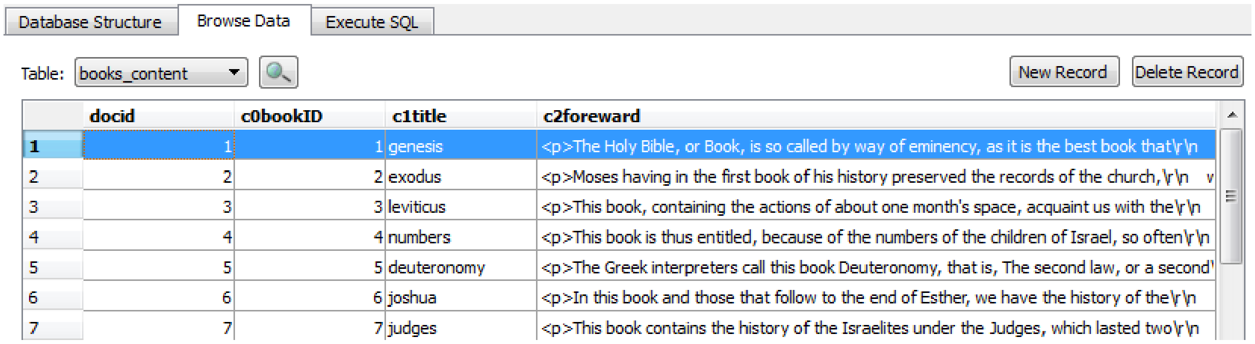


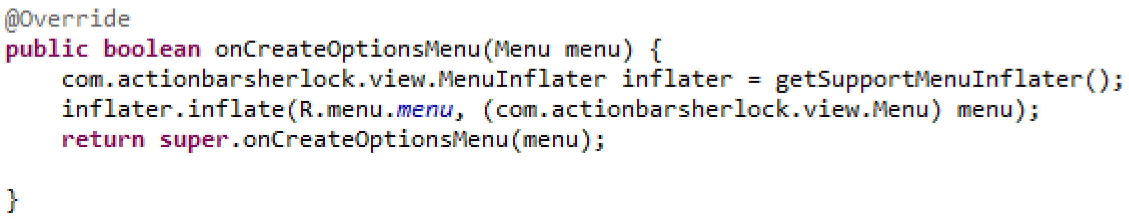
Table: books_content

	docid	c0bookID	c1title	c2foreward
1	1	1	genesis	<p>The Holy Bible, or Book, is so called by way of eminency, as it is the best book that\r\n
2	2	2	exodus	<p>Moses having in the first book of his history preserved the records of the church,\r\n v
3	3	3	leviticus	<p>This book, containing the actions of about one month's space, acquaint us with the\r\n
4	4	4	numbers	<p>This book is thus entitled, because of the numbers of the children of Israel, so often\r\n
5	5	5	deuteronomy	<p>The Greek interpreters call this book Deuteronomy, that is, The second law, or a second'
6	6	6	joshua	<p>In this book and those that follow to the end of Esther, we have the history of the\r\n
7	7	7	judges	<p>This book contains the history of the Israelites under the Judges, which lasted two\r\n

Figure 04 - Database within SQLite Database Browser

After having the database figured out, the focus was on how the user would interact with the application. Having an ActionBar, seen in Figure 02,at the top of every page was a huge priortity because it allows for fast movement through the application. Creating this action bar should have been a trivial task but it actually became quite an obstacle. The problem was that the application was designed to be compatible with the android operating system versions back to 2.1 "Eclair". This was primarily because the phone that was being used for testing was running v2.1 and it is easier to build an application with backwards compatibility during initial development then to go back afterwards and try and port a completed project. The primary problem was that the ActionBar was not added as a native element until version 3.0. If this application was to support backwards compatibility to 2.1 and have an action bar, a work around for this problem would need to be found. .

The solution that was found was an extension of the support library called ActionBarSherlock. This library automatically uses the native action bar when the version is high enough to support or use a custom implementation when it does not. This allowed for the creation of a single ActionBar through this library.From there, the library would decide when to use the native or custom implementation of the ActionBar. After going through the extensive setup process for the library, it was as simple as overriding the OnCreateOptionMenu method.



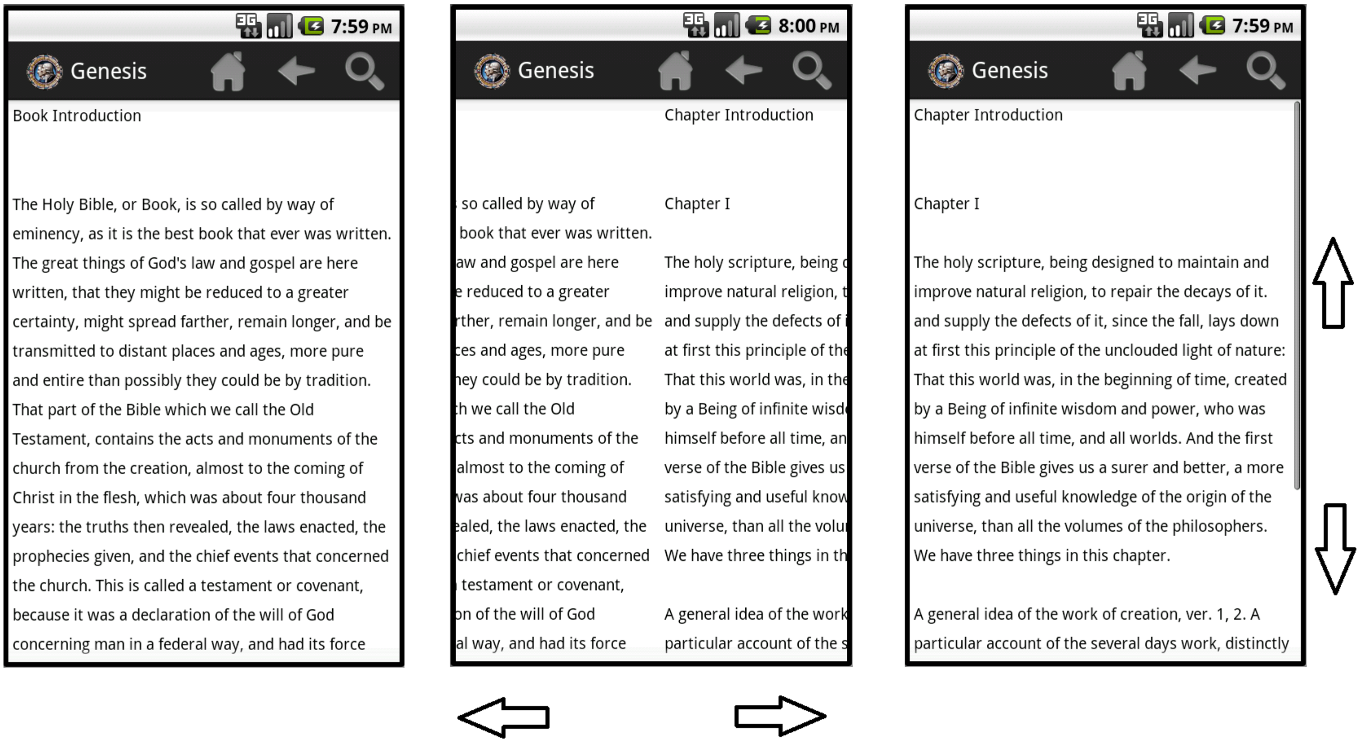
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    com.actionbarsherlock.view.MenuInflater inflater = getSupportMenuInflater();
    inflater.inflate(R.menu.menu, (com.actionbarsherlock.view.Menu) menu);
    return super.onCreateOptionsMenu(menu);

}
```

Figure 05 - SherlockActionBar Implementation

From there, the ActionBar could be customized just like a traditional ActionBar. Although finding a solution for this problem was time consuming, it is a great additionbecause it extends the usability of this application to older smart phones.

One of the last, and possibly most difficult to implement, parts of the project was the ability for a user to scroll vertically through a single verse and also page horizontally between multiple verses. To scroll vertically, typically a ScrollView is used. To scroll horizontally, a ViewPager is used to flip through multiple pages of data.



Book Introduction

The Holy Bible, or Book, is so called by way of eminency, as it is the best book that ever was written. The great things of God's law and gospel are here written, that they might be reduced to a greater certainty, might spread farther, remain longer, and be transmitted to distant places and ages, more pure and entire than possibly they could be by tradition. That part of the Bible which we call the Old Testament, contains the acts and monuments of the church from the creation, almost to the coming of Christ in the flesh, which was about four thousand years: the truths then revealed, the laws enacted, the prophecies given, and the chief events that concerned the church. This is called a testament or covenant, because it was a declaration of the will of God concerning man in a federal way, and had its force

so called by way of
book that ever was written.
aw and gospel are here
e reduced to a greater
rther, remain longer, and be
es and ages, more pure
ey could be by tradition.
h we call the Old
cts and monuments of the
almost to the coming of
as about four thousand
ealed, the laws enacted, the
chief events that concerned
testament or covenant,
n of the will of God
al way, and had its force

Chapter Introduction

Chapter I

The holy scripture, being
improve natural religion,
and supply the defects of
at first this principle of th
That this world was, in the
by a Being of infinite wisd
himself before all time, an
verse of the Bible gives us
satisfying and useful know
universe, than all the volu
We have three things in th

A general idea of the work
particular account of the s

Chapter Introduction

Chapter I

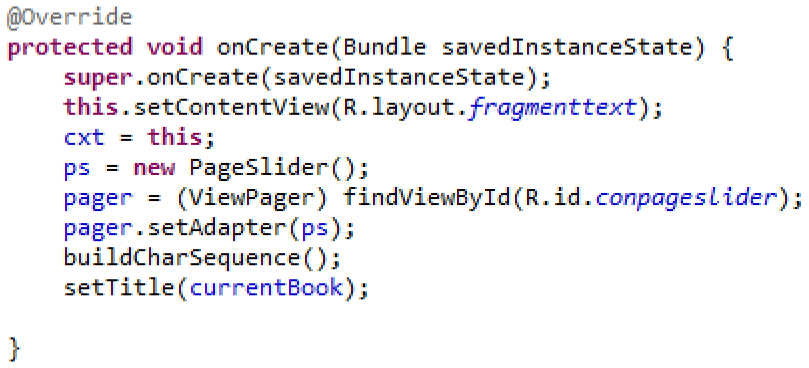
The holy scripture, being designed to maintain and improve natural religion, to repair the decays of it. and supply the defects of it, since the fall, lays down at first this principle of the unclouded light of nature: That this world was, in the beginning of time, created by a Being of infinite wisdom and power, who was himself before all time, and all worlds. And the first verse of the Bible gives us a surer and better, a more satisfying and useful knowledge of the origin of the universe, than all the volumes of the philosophers. We have three things in this chapter.

A general idea of the work of creation, ver. 1, 2. A particular account of the several days work, distinctly

Figure 06 - Horizontal and Vertical Scroll

Attempting to combine these two together was easier said than done. What made the problem even more complex was that fact that these would be created dynamically based off of the chapter that was selected. The length of a chapter or the length of a specific commentary was unknown. All of this had to be taken into account when putting this feature together.

The way the problem was solved was by creating a ViewPager and then instantiating each item dynamically as an individual ScrollView within that ViewPager. This is made possible through the use of a PagerAdapter, an adapter used to populate pages inside a ViewPager. This technique can be seen in the OnCreate method in Figure 07.

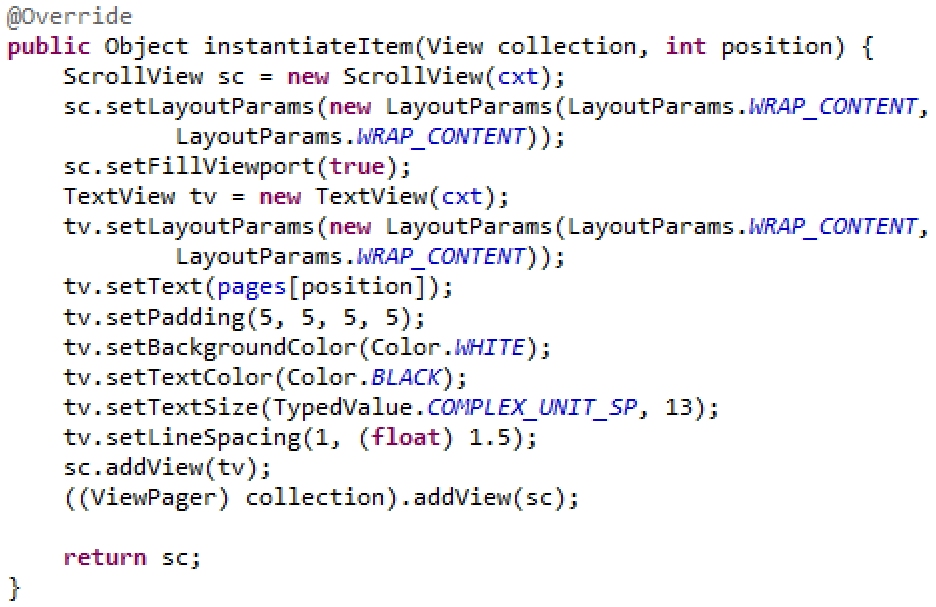


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    this.setContentView(R.layout.fragmenttext);
    cxt = this;
    ps = new PageSlider();
    pager = (ViewPager) findViewById(R.id.conpageslider);
    pager.setAdapter(ps);
    buildCharSequence();
    setTitle(currentBook);

}
```

Figure 07 - ViewPager Creation

The custom PagerAdapter containing our ScrollViewsis set as the adapter for ourViewPager. The ViewPager is partially defined ahead of time within an XML file and each scroll view is created dynamically as each page is instantiated, as seen in Figure 08.



```
@Override
public Object instantiateItem(View collection, int position) {
    ScrollView sc = new ScrollView(cxt);
    sc.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT,
            LayoutParams.WRAP_CONTENT));
    sc.setFillViewport(true);
    TextView tv = new TextView(cxt);
    tv.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT,
            LayoutParams.WRAP_CONTENT));
    tv.setText(pages[position]);
    tv.setPadding(5, 5, 5, 5);
    tv.setBackgroundColor(Color.WHITE);
    tv.setTextColor(Color.BLACK);
    tv.setTextSize(TypedValue.COMPLEX_UNIT_SP, 13);
    tv.setLineSpacing(1, (float) 1.5);
    sc.addView(tv);
    ((ViewPager) collection).addView(sc);

    return sc;
}
```

Figure 08 - ScrollView Creation

In previous attempts to solve this problem,a ScrollView was defined ahead of time within an XML layout. This worked in some situations but caused some problem when tring to use it inside the ViewPager. There were various formatting and touch problems when this technique was used, so that is why it was decided to just create the ScrollView programmatically.

Unfinished Portions and Future Work

The main portion of this project that remains unfinished is the search functionality. Although the framework is in place to make a searching system possible, it was unable to be completed because of time constraints. It is unfortunate that the search function was not completed because the user really needs the ability to search these explanatory notes. If someone was to pick this project up, the primary focus should probably be to get the search functionality working in this application. As said previously, the FTS3 extension is already in place to make searching possible within the database. Only the actual use of that extension needs to be implemented. Part of getting this to work would probably mean going back and researching how Josh Erhlin had implemented his search functionality with PhP. From research, it looks like he had a web interface working that could search this same database structure. It might be possible to combine what he has done in his work with the FTS3 framework here to create a working search option.

Another part of this project that needs to be completed is the actual formatting of the explanatory notes. All of the text is there and it is readable, but there needs to be some work done on making those chunks of text better looking and easier to read. Currently, line spacing is the same throughout all the notes, there is no custom font size, no bold or italicized words, and there is no option for the user to change the font size. All of these are possible; they would just take some time to implement. Because this is an application with a primary purpose of viewing text notes, this part of the project really needs to be completed before it can be considered usable.

Conclusion

When I began on this project I had very little experience working with the android environment. Part of the reason why I decided to work on this project was because I knew I would have to learn a lot to make it work. After all of my development, I am so glad I chose this as my senior project because it has opened my eyes to mobile development and how much I enjoy doing it. It would be great to continue with mobile development as a future career.

Throughout this project there were many ups and downs and I struggled with this project more times than I care to remember. That is one reason why it turned out to be such a great project. I struggled, but through that struggle I learned more than I ever could otherwise. Although there are many things still left to do on this project, I still see it as an accomplishment because the application has come so far. It would be great to see this project continued on by another senior in the future. The Wesley Center could definitely benefit from the completion and release of this application.

References

"Introduction to FTS3 and FTS4."*SQLite FTS3 and FTS4 Extensions*.N.p., n.d. Web. 30 Apr. 2013.

"Develop | Android Developers." *Develop | Android Developers*. N.p., n.d. Web. 30 Apr. 2013.

Ehrlin, JoshuaH.*Wesley Notes: Development of a Searchable Database*. Thesis.Northwest Nazarene University, 2002. Print.

"SQLite Home Page."*SQLite Home Page*.N.p., n.d. Web. 30 Apr. 2013.

"Usage."*ActionBarSherlock*.N.p., n.d. Web. 30 Apr. 2013.

"Wesley Center Online."*The : Home*. N.p., n.d. Web. 30 Apr. 2013.

Appendix A

```
ChaptersDatabaseActivity.java

package android.sqllite;

import my.android.sqllite.R;
import android.os.Bundle;
import java.util.ArrayList;
import java.util.List;
import com.actionbarsherlock.app.SherlockListActivity;
import com.actionbarsherlock.view.Menu;
import com.actionbarsherlock.view.MenuItem;
import android.content.Intent;
import android.text.Html;
import android.text.Spanned;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

// ChaptersDatabaseActivity provides the interface of displaying our book
chapters
// SherlockListActivity provides access to the action bar on older versions
of android
publicclass ChaptersDatabaseActivity extends SherlockListActivity {
private CommentsDataSource datasource;
privateint repeatVerseCount =0;
private String REPEAT;
privateboolean wasLastCheckTrue;
privateboolean addDuplicate;

    @Override
publicvoid onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setTitle("Chapters");
setContentView(R.layout.main);

datasource=new CommentsDataSource(this);
datasource.open();

        Bundle extras =getIntent().getExtras();
        String value =extras.getString("databasetext");

        List<Comment> values =datasource.getAllChapters(value);

// Use the SimpleCursorAdapter to show the
// elements in a ListView
        ArrayAdapter<Comment> adapter =new ArrayAdapter<Comment>(this,
      R.layout.listlayout, values);

setListAdapter(adapter);

        ListView list =(ListView)findViewById(android.R.id.list);
list.setOnItemClickListener(new AdapterView.OnItemClickListener(){
```

```
// onItemClick our database is queried based of that selection
// text is formatted and stored in an intent that is then passed
      to our viewpager
// In the future, this class might need to be separated into
      multiple methods.
publicvoid onItemClick(AdapterView<?> parent, View view,int
      position,long id){


                List<Comment> chapterIntro =
                  datasource.getTextComments(position);
                List<Comment> bookIntro =datasource.getBookNotes(position);

                String hey =chapterIntro.toString();
                String header =String.valueOf(position +1);

                List<String> CharItems =new ArrayList<String>();

                String value;
                Spanned marked_up_notes =
            Html.fromHtml(bookIntro.toString());
value= marked_up_notes.toString();
value= value.substring(1, value.length()-1);
value="Book Introduction"+"\n"+ value;
CharItems.add(value);

                marked_up_notes =Html.fromHtml(chapterIntro.toString());
value= marked_up_notes.toString();
value= value.substring(1, value.length()-1);
value="Chapter Introduction"+"\n"+ value;
CharItems.add(value);

int count =0;

int totalnotes = datasource.getNoteCountForChapter(position);
                List<Comment> notes =datasource.getTextNotes(position);

while(count < totalnotes){

                    marked_up_notes = Html
.fromHtml(notes.get(count).toString());
value= marked_up_notes.toString();

if(checkForDuplicateVerse(value, CharItems, count)==
      true){
++count;

}else{

if(addDuplicate ==true){
CharItems.remove(CharItems.size()-1);
CharItems.add(REPEAT);
addDuplicate=false;

}

++count;
```

```
                        REPEAT = value;
value="Verse "+ count +"\n\n"+ value;
CharItems.add(value);

}

}

CharSequence[] Chars = CharItems.toArray(new
      CharSequence[CharItems.size()]);


                Intent i =new
      Intent(android.sqllite.ChaptersDatabaseActivity.this,
      ScrollingViewPager.class);
i.putExtra("charseq", Chars);
i.putExtra("SelectedItem", position +1);
i.putExtra("databasetext", hey);
i.putExtra("databaseheader", header);
if(position ==0){
                    String intro =bookIntro.toString();
i.putExtra("databasenotes", intro);
}else{
i.putExtra("databasenotes","Null");
}

//
i.putExtra("title",           datasource.bookIDtoBookName().toString());//
grabs the current book

startActivity(i);

}

});

}

    @Override
protectedvoid onResume(){
datasource.open();
super.onResume();
}

    @Override
protectedvoid onPause(){
datasource.close();
super.onPause();
}
```

```
// database contains duplicate verses where an entry was a combination of
      verses - ex. 1,2,3
// previous developer for the web interface left verses in entry spots of
      1,2,3 instead of splitting them up
// this function finds duplicates and combines them into  a single entry
      for the fragment
privateboolean checkForDuplicateVerse(String value, List<String>  charItems,
Integer count){

if(value.equals(REPEAT)){
++repeatVerseCount;
wasLastCheckTrue=true;
returntrue;
}

if(wasLastCheckTrue ==true){

                String verseCombo ="";
while(repeatVerseCount >-1){
                    Integer flip = count - repeatVerseCount;
                    verseCombo = verseCombo +","+flip.toString();
--repeatVerseCount;
}

verseCombo= verseCombo.substring(1, verseCombo.length());
                REPEAT ="Verses "+ verseCombo +"\n\n"+ REPEAT;
addDuplicate=true;
++repeatVerseCount;

}

wasLastCheckTrue=false;

returnfalse;
}

    @Override
publicboolean onCreateOptionsMenu(Menu menu){
com.actionbarsherlock.view.MenuInflater inflater =
      getSupportMenuInflater();
inflater.inflate(R.menu.menu,(com.actionbarsherlock.view.Menu)
      menu);
returnsuper.onCreateOptionsMenu(menu);

}

    @Override
publicboolean onMenuItemSelected(int featureId, MenuItem item){

int itemId = item.getItemId();
switch(itemId){
case R.id.menu_home:
            Intent i
=newIntent(android.sqllite.ChaptersDatabaseActivity.this,
android.sqllite.Menu.class);// Search.class
startActivity(i);
case R.id.menu_back:
finish();
break;

case R.id.menu_search:

break;

}

returntrue;
}
}
```

Comment.java

```
package android.sqllite;

publicclass Comment {
privatelong id;
private String comment;

publiclong getId(){
return id;
}

publicvoid setId(long id){
this.id = id;
}

public String getComment(){
return comment;
}

publicvoid setComment(String comment){
this.comment = comment;
}

// Will be used by the ArrayAdapter in the ListView
      @Override
public String toString(){
return comment;
}
}
```

CommentDataSource.java

```
package android.sqllite;

import java.util.ArrayList;
import java.util.List;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

//CommentsDataSource is the layer in between our activity classes and
database..
//Provides us access to all of our database related operations.
publicclass CommentsDataSource {

    String CurrentBook;
    Integer BookCount;

// Database fields
private SQLiteDatabase database;
private MySQLiteHelper dbHelper;
private String[] allBookColumns ={ MySQLiteHelper.COLUMN_BOOKID,
      MySQLiteHelper.COLUMN_BOOKTITLE, MySQLiteHelper.COLUMN_BOOKFOREWARD };



public CommentsDataSource(Context context){
dbHelper=new MySQLiteHelper(context);
}

publicvoid open()throws SQLException {
database= dbHelper.getWritableDatabase();
}

publicvoid close(){
dbHelper.close();
}

public Comment createComment(String comment){
    ContentValues values =newContentValues();
values.put(MySQLiteHelper.COLUMN_COMMENT, comment);
long insertId = database.insert(MySQLiteHelper.TABLE_COMMENTS,null, values);

    Cursor cursor =database.query(MySQLiteHelper.TABLE_COMMENTS,
allBookColumns, MySQLiteHelper.COLUMN_ID +" = "+
insertId,null,null,null,null);

cursor.moveToFirst();
    Comment newComment =cursorToComment(cursor);
cursor.close();
return newComment;
}

// search is still a work in progress
public String search(String search){
      Cursor cursor =null;
//cursor = this.database.query(true, MySQLiteHelper.TABLE_WORDS_FTS,
      new String[] { "bookID", "title", "foreward" }, "WORDS_FTS" + " MATCH
      ?", new String[] { search }, null, null, null, null);

      String select ="SELECT * FROM books WHERE foreward MATCH '"+ search
+"';";
cursor= database.rawQuery(select,null);
if(cursor !=null){
cursor.moveToFirst();
}
if(cursor.getCount()==0){

return"Invalid Input";
}
int results = cursor.getColumnIndex("foreward");
      String rez =cursor.getString(results);
cursor.close();
return rez;
}



publicvoid deleteComment(Comment comment){
long id = comment.getId();
System.out.println("Comment deleted with id: "+ id);
database.delete(MySQLiteHelper.TABLE_COMMENTS, MySQLiteHelper.COLUMN_ID +" =
"+ id,null);

}

// returns the full list of books
public List<Comment> getAllComments(){
    List<Comment> comments =new ArrayList<Comment>();

    Cursor cursor =database.query(MySQLiteHelper.TABLE_WORDS_FTS,
allBookColumns,null,null,null,null,null);

cursor.moveToFirst();
while(!cursor.isAfterLast()){
      Comment comment =cursorToComment(cursor);
comments.add(comment);
cursor.moveToNext();
}

// Make sure to close the cursor
cursor.close();
setChapterCount(cursor.getCount());
return comments;
}

// returns a list of chapters for a specific book
public List<Comment> getAllChapters(String value){

String[] allChapterColumns ={ MySQLiteHelper.COLUMN_CHAPTERID,
value, MySQLiteHelper.COLUMN_DESCRIPTION };
```

```
      String Where = MySQLiteHelper.COLUMN_BOOKID +" = "+ value +" ";

      CurrentBook = value;

        List<Comment> comments =new ArrayList<Comment>();

        Cursor cursor =database.query(MySQLiteHelper.TABLE_CHAPTERS,
allChapterColumns, Where,null,null,null,null);

cursor.moveToFirst();
while(!cursor.isAfterLast()){
          Comment comment =cursorToChapterComment(cursor);
comments.add(comment);
cursor.moveToNext();
}

setChapterCount(cursor.getCount());// sets the total number of    chapters in
a book

// Make sure to close the cursor
cursor.close();
return comments;
}


private Comment cursorToComment(Cursor cursor){
    Comment comment =newComment();
comment.setId(cursor.getLong(0));
comment.setComment(cursor.getString(1));
return comment;
}

private Comment cursorToChapterComment(Cursor cursor){
        Comment comment =newComment();
comment.setId(cursor.getLong(0));
comment.setComment(cursor.getString(0));
return comment;
}

// returns the chapter introduction for a specific chapter
public List<Comment> getTextComments(Integer value){
        List<Comment> comments =new ArrayList<Comment>();

        String Where = MySQLiteHelper.COLUMN_CHAPTERID +" = "+(value +1)+"
AND "+ MySQLiteHelper.COLUMN_BOOKID +" = "+ CurrentBook;

String[] allChapterColumns ={ MySQLiteHelper.COLUMN_CHAPTERID,
                  MySQLiteHelper.COLUMN_BOOKID,
MySQLiteHelper.COLUMN_DESCRIPTION };

        Cursor cursor =database.query(MySQLiteHelper.TABLE_CHAPTERS,
allChapterColumns, Where,null,null,null,null);


cursor.moveToFirst();
while(!cursor.isAfterLast()){
```

```
          Comment comment =cursorToText(cursor);
comments.add(comment);
cursor.moveToNext();
}
// Make sure to close the cursor
cursor.close();
return comments;
}

//returns text for a specific chapter
public List<Comment> getTextNotes(Integer value){
        List<Comment> comments =new ArrayList<Comment>();

        String Where = MySQLiteHelper.COLUMN_CHAPTERID +" = "+(value +1)+"
AND "+ MySQLiteHelper.COLUMN_BOOKID +" = "+ CurrentBook;

String[] allChapterColumns ={ MySQLiteHelper.COLUMN_NOTEID,
MySQLiteHelper.COLUMN_CHAPTERID,
MySQLiteHelper.COLUMN_BOOKID ,MySQLiteHelper.COLUMN_NOTE};

        Cursor cursor =database.query(MySQLiteHelper.TABLE_NOTES,
allChapterColumns, Where,null,null,null,null);


cursor.moveToFirst();
while(!cursor.isAfterLast()){
          Comment comment =noteCursorToText(cursor);
comments.add(comment);
cursor.moveToNext();
}
// Make sure to close the cursor
cursor.close();
return comments;
}

// returns the book introduction for a specific book
public List<Comment> getBookNotes(Integer value){
        List<Comment> comments =new ArrayList<Comment>();

        String Where = MySQLiteHelper.COLUMN_BOOKID +" = "+ CurrentBook;

String[] allChapterColumns ={ MySQLiteHelper.COLUMN_BOOKID,
                  MySQLiteHelper.COLUMN_BOOKTITLE,
MySQLiteHelper.COLUMN_BOOKFOREWARD };

        Cursor cursor =database.query(MySQLiteHelper.TABLE_BOOKS,
allChapterColumns, Where,null,null,null,null);


cursor.moveToFirst();
while(!cursor.isAfterLast()){
          Comment comment =cursorToText(cursor);
comments.add(comment);
cursor.moveToNext();
}
// Make sure to close the cursor
cursor.close();
```

```
return comments;
}

private Comment cursorToText(Cursor cursor){
        Comment comment =newComment();
comment.setId(cursor.getLong(0));
comment.setComment(cursor.getString(2));
return comment;
}

private Comment noteCursorToText(Cursor cursor){
            Comment comment =newComment();
comment.setId(cursor.getLong(0));
comment.setComment(cursor.getString(3));
return comment;
}

//sets the book count from the most recently selected book
privatevoid setChapterCount(Integer value){

            BookCount = value;

}

//returns the book count from the most recently selected book
publicint getChapterCount(){

return BookCount;

}

// returns the number of chapters for a given book
publicint getNoteCountForChapter(Integer value){
          List<Comment> comments =new ArrayList<Comment>();

            String Where = MySQLiteHelper.COLUMN_CHAPTERID +" = "+(value
+1)+" AND "+ MySQLiteHelper.COLUMN_BOOKID +" = "+ CurrentBook;

String[] allChapterColumns ={ MySQLiteHelper.COLUMN_NOTEID,
MySQLiteHelper.COLUMN_CHAPTERID,
MySQLiteHelper.COLUMN_BOOKID ,MySQLiteHelper.COLUMN_NOTE};

            Cursor cursor
=database.query(MySQLiteHelper.TABLE_NOTES,//table_books
allChapterColumns, Where,null,null,null,null);


cursor.moveToFirst();
while(!cursor.isAfterLast()){
              Comment comment =cursorToText(cursor);
comments.add(comment);
cursor.moveToNext();
}

value= cursor.getCount();
// Make sure to close the cursor
cursor.close();
```

```
return value;

}

// converts a book ID to its string name
public List<Comment> bookIDtoBookName(){

          List<Comment> comments =new ArrayList<Comment>();

          String Where = MySQLiteHelper.COLUMN_BOOKID +" = "+ CurrentBook;

            Cursor cursor =database.query(MySQLiteHelper.TABLE_WORDS_FTS,
allBookColumns, Where,null,null,null,null);

cursor.moveToFirst();
while(!cursor.isAfterLast()){
              Comment comment =cursorToComment(cursor);
comments.add(comment);
cursor.moveToNext();
}

// Make sure to close the cursor
cursor.close();


return comments;
}

}
```

Menu.java

```
package android.sqllite;

import my.android.sqllite.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

//Menu serves as our main activity
publicclass Menu extends Activity {

    Button startbutton;
    Button searchbutton;
    Button aboutbutton;

publicvoid onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.menulayout);

addListenerOnStartButton();
addListenerOnSearchButton();
addListenerOnAboutButton();

}

privatevoid addListenerOnStartButton(){

startbutton=(Button) findViewById(R.id.button3);

startbutton.setOnClickListener(new OnClickListener(){

publicvoid onClick(View arg0){

                Intent i =newIntent(android.sqllite.Menu.this,
                        TestDatabaseActivity.class);
startActivity(i);

}

});

}

privatevoid addListenerOnSearchButton(){

searchbutton=(Button) findViewById(R.id.button2);

searchbutton.setOnClickListener(new OnClickListener(){

publicvoid onClick(View arg0){

                Intent i =newIntent(android.sqllite.Menu.this,
                  Search.class);
startActivity(i);
}
});
}
privatevoid addListenerOnAboutButton(){
aboutbutton=(Button) findViewById(R.id.button1);
aboutbutton.setOnClickListener(new OnClickListener(){
publicvoid onClick(View arg0){
}
});
}
}
```

MySQLiteHelper.java

```
package android.sqllite;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

// MySQLiteHelper is used to create our database from our
baseBackup_041311.sql file
publicclass MySQLiteHelper extends SQLiteOpenHelper {

publicstaticfinal String TABLE_COMMENTS ="comments";
publicstaticfinal String COLUMN_ID ="_id";
publicstaticfinal String COLUMN_COMMENT ="comment";

publicstaticfinal String TABLE_BOOKS ="books";
publicstaticfinal String COLUMN_BOOKID ="bookID";
publicstaticfinal String COLUMN_BOOKTITLE ="title";
publicstaticfinal String COLUMN_BOOKFOREWARD ="foreward";

publicstaticfinal String TABLE_CHAPTERS ="chapters";
publicstaticfinal String COLUMN_CHAPTERID ="chapterID";
publicstaticfinal String COLUMN_DESCRIPTION ="description";

publicstaticfinal String TABLE_NOTES ="notes";
publicstaticfinal String COLUMN_NOTEID ="noteID";
publicstaticfinal String COLUMN_NOTE ="note";

publicstaticfinal String TABLE_WORDS_FTS ="books";//TABLE_WORDS_FTS =
"WORDS_FTS";

privatestaticfinal String DATABASE_NAME ="commments.db";
privatestaticfinalint DATABASE_VERSION =1;


privatestaticfinal String VDATABASE_CREATE_BOOKS ="CREATE VIRTUAL TABLE "
+ TABLE_WORDS_FTS +" USING fts3("+ COLUMN_BOOKID +", "+ COLUMN_BOOKTITLE +",
"+ COLUMN_BOOKFOREWARD +" "+");";

privatestaticfinal String VDATABASE_CREATE_CHAPTERS ="CREATE VIRTUAL TABLE "
+ TABLE_CHAPTERS +" USING fts3("+ COLUMN_CHAPTERID +", "+ COLUMN_BOOKID +",
"+ COLUMN_DESCRIPTION +" "+");";

privatestaticfinal String VDATABASE_CREATE_NOTES ="CREATE VIRTUAL TABLE "
+ TABLE_NOTES +" USING fts3("+ COLUMN_NOTEID +", "+ COLUMN_CHAPTERID +", "+
COLUMN_BOOKID +" "+", "+ COLUMN_NOTE +" "+");";

public MySQLiteHelper(Context context){
super(context, DATABASE_NAME,null, DATABASE_VERSION);
}

    @Override
```

```
publicvoid onCreate(SQLiteDatabase database){

database.execSQL(VDATABASE_CREATE_BOOKS);//create our books table
database.execSQL(VDATABASE_CREATE_CHAPTERS);//create our chapters table
database.execSQL(VDATABASE_CREATE_NOTES);//create our notes table
importSQL(database);

}

    @Override
publicvoid onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion){
Log.w(MySQLiteHelper.class.getName(),
"Upgrading database from version "+ oldVersion +" to "
+ newVersion +", which will destroy all old data");
db.execSQL("DROP TABLE IF EXISTS "+ TABLE_COMMENTS);
onCreate(db);
}


// this function parses the baseBackup_041311.sql file and creates the
database
publicstaticvoid importSQL(SQLiteDatabase database){

        String currentLine;
        String outputLine;
        String ret =null;

        File appBase =newFile(".");
        String path =appBase.getAbsolutePath();
System.out.println(path);
        InputStream stream =null;
stream= MySQLiteHelper.class.getClassLoader().getResourceAsStream(
"baseBackup_041311.sql");

if(stream !=null){

            BufferedReader br =newBufferedReader(
new InputStreamReader(stream));

try{
while((currentLine = br.readLine())!=null){

if(currentLine.length()==0
||currentLine.charAt(0)=='-'){

}else{
if(currentLine.contains("INSERT")){
ret= currentLine;

}else{

if(currentLine.charAt(currentLine.length()-1)
      ==','){

currentLine= currentLine.substring(0,
      currentLine.length()-1);
currentLine= currentLine +';';
outputLine= ret + currentLine;

database.execSQL(outputLine);

}
else{

outputLine= ret + currentLine;
database.execSQL(outputLine);

}

}

}

}
}catch(IOException e){
// TODO Auto-generated catch block
e.printStackTrace();
}

}

}

}
```

```
ScrollingViewPager.java

package android.sqllite;

import com.actionbarsherlock.app.SherlockActivity;
import com.actionbarsherlock.view.MenuItem;
import com.actionbarsherlock.view.Menu;
import my.android.sqllite.R;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.ViewPager;
import android.util.TypedValue;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.ScrollView;
import android.widget.TextView;

//ScrollingViewPager is used to display our text from the database
//It combines vertical scroll views within the viewpager and fragment
framework
//SherlockActivity provides access to the action bar on older versions of
android
publicclass ScrollingViewPager extends SherlockActivity {

private ViewPager pager;
private Context cxt;
private CharSequence[] pages;
private PageSlider ps;
private String currentBook;

    @Override
protectedvoid onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
this.setContentView(R.layout.fragmenttext);
cxt=this;
ps=new PageSlider();
pager=(ViewPager) findViewById(R.id.conpageslider);
pager.setAdapter(ps);

buildCharSequence();
setTitle(currentBook);

}

privatevoid buildCharSequence(){

        Bundle extras =getIntent().getExtras();
if(extras !=null){
pages= extras.getCharSequenceArray("charseq");
currentBook= extras.getString("title");
currentBook= currentBook.substring(1, currentBook.length()-1);
}

}

publicclass PageSlider extends PagerAdapter {

        @Override
publicint getCount(){
return pages.length;
}
```

```
        @Override
public Object instantiateItem(View collection,int position){
            ScrollView sc =newScrollView(cxt);
sc.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT,
                    LayoutParams.WRAP_CONTENT));
sc.setFillViewport(true);
            TextView tv =newTextView(cxt);
tv.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT,
                    LayoutParams.WRAP_CONTENT));
tv.setText(pages[position]);
tv.setPadding(5,5,5,5);
tv.setBackgroundColor(Color.WHITE);
tv.setTextColor(Color.BLACK);
tv.setTextSize(TypedValue.COMPLEX_UNIT_SP,13);
tv.setLineSpacing(1,(float)1.5);
sc.addView(tv);
((ViewPager) collection).addView(sc);

return sc;
}

        @Override
publicvoid destroyItem(View collection,int position, Object view){
((ViewPager) collection).removeView((ScrollView) view);
}

        @Override
publicboolean isViewFromObject(View view, Object object){
return view ==((ScrollView) object);
}

}

    @Override
publicboolean onCreateOptionsMenu(Menu menu){
com.actionbarsherlock.view.MenuInflater inflater =
      getSupportMenuInflater();
inflater.inflate(R.menu.menu,(com.actionbarsherlock.view.Menu) menu);
returnsuper.onCreateOptionsMenu(menu);

}

    @Override
publicboolean onMenuItemSelected(int featureId, MenuItem item){

int itemId = item.getItemId();
switch(itemId){
case R.id.menu_home:
            Intent i =newIntent(android.sqllite.ScrollingViewPager.this,
                    android.sqllite.Menu.class);// Search.class
startActivity(i);

case R.id.menu_back:
finish();
break;

case R.id.menu_search:
```

```
break;

}

returntrue;
}

}
```

Search.java

```
package android.sqllite;

import my.android.sqllite.R;
import android.app.ListActivity;
import android.app.SearchManager;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

//Search is currently a class that is not being implemented.
//This class will be completed and used later in development.
publicclass Search extends ListActivity {

publicvoid onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.searchlayout);

}

publicvoid myClickHandler(View v){

onSearchRequested();

}

protectedvoid onNewIntent(Intent intent){
setIntent(intent);
handleIntent(intent);
}

privatevoid handleIntent(Intent intent){
if(Intent.ACTION_SEARCH.equals(intent.getAction())){
            String query =intent.getStringExtra(SearchManager.QUERY);
Query(query);
}
}

publicvoid Query(String find){

        CommentsDataSource datasource;
datasource=new CommentsDataSource(this);
datasource.open();
        String ret =datasource.search(find);

        Intent i =newIntent(android.sqllite.Search.this, Text.class);//
Text.class
i.putExtra("databasetext", ret);
startActivity(i);

}

}
```

TestDatabaseActivity.java

```
package android.sqllite;

import my.android.sqllite.R;
import android.os.Bundle;
import java.util.List;
import com.actionbarsherlock.app.SherlockListActivity;
import com.actionbarsherlock.view.Menu;
import com.actionbarsherlock.view.MenuItem;
import android.content.Intent;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

//TestDatabaseActivity (name will be changed before release) provides the
interface of displaying our list of books
//SherlockListActivity provides access to the action bar on older versions of
android
publicclass TestDatabaseActivity extends SherlockListActivity {
private CommentsDataSource datasource;

    @Override
publicvoid onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);

datasource=new CommentsDataSource(this);
datasource.open();

        List<Comment> values =datasource.getAllComments();// returns our
            list of books

        ArrayAdapter<Comment> adapter =new ArrayAdapter<Comment>(this,
      R.layout.listlayout, values);

setListAdapter(adapter);

        ListView list =(ListView)findViewById(android.R.id.list);
list.setOnItemClickListener(new AdapterView.OnItemClickListener(){

publicvoid onItemClick(AdapterView<?> parent, View view,
int position,long id){

int passer = position +1;

                String hey =Integer.toString(passer);

                Intent i
=newIntent(android.sqllite.TestDatabaseActivity.this,
                        ChaptersDatabaseActivity.class);
i.putExtra("databasetext", hey);
startActivity(i);

}

});

}

    @Override
protectedvoid onResume(){
datasource.open();
super.onResume();
}
```

```
    @Override
protectedvoid onPause(){
datasource.close();
super.onPause();
}

    @Override
publicboolean onCreateOptionsMenu(Menu menu){
com.actionbarsherlock.view.MenuInflater inflater = getSupportMenuInflater();
inflater.inflate(R.menu.menu,(com.actionbarsherlock.view.Menu) menu);
returnsuper.onCreateOptionsMenu(menu);

}

    @Override
publicboolean onMenuItemSelected(int featureId, MenuItem item){

int itemId = item.getItemId();
switch(itemId){
case R.id.menu_home:
            Intent i =newIntent(android.sqllite.TestDatabaseActivity.this,
                    android.sqllite.Menu.class);
startActivity(i);

case R.id.menu_back:
finish();
break;

case R.id.menu_search:

break;

}

returntrue;
}
}
```

Text.java

```
package android.sqllite;

import com.actionbarsherlock.app.SherlockActivity;
import com.actionbarsherlock.view.Menu;
import com.actionbarsherlock.view.MenuItem;
import my.android.sqllite.R;
import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.text.Spanned;
import android.text.method.ScrollingMovementMethod;
import android.util.TypedValue;
import android.view.View;
import android.widget.TextView;
import android.graphics.Color;

//Text is currently a class that is not being implemented.
//Early in development this was a prototype class that displayed text
//from our database queries.
publicclass Text extends SherlockActivity {

publicvoid onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.text);



            Bundle extras =getIntent().getExtras();
if(extras !=null){
                String value =extras.getString("databasetext");
                String notes =extras.getString("databasenotes");
                String header =extras.getString("databaseheader");


setTitle("Chapter "+ header);// "Chapter " + header

                TextView note =(TextView)findViewById(R.id.textView1);
note.setBackgroundColor(Color.rgb(237,237,237));
note.setTextSize(TypedValue.COMPLEX_UNIT_SP,12);
note.setTextColor(Color.BLACK);
note.setMovementMethod(new ScrollingMovementMethod());

                TextView chapter =(TextView)findViewById(R.id.textView2);
chapter.setBackgroundColor(Color.WHITE);
chapter.setTextColor(Color.BLACK);
chapter.setTextSize(TypedValue.COMPLEX_UNIT_SP,12);
chapter.setMovementMethod(new ScrollingMovementMethod());

if(notes.equals("Null")){
                    TextView viewToHide =(TextView)
                  findViewById(R.id.textView1);
viewToHide.setVisibility(View.INVISIBLE);
}
else{
notes= notes.substring(1, notes.length()-1);
                    Spanned marked_up_notes =Html.fromHtml(notes);
((TextView)findViewById(R.id.textView1)).setText(marked_up_notes.toString());

}

value= value.substring(1, value.length()-1);// remove [  ]

                Spanned marked_up_chapter =Html.fromHtml(value);
((TextView)findViewById(R.id.textView2)).setText(marked_up_chapter.toString()
);
//((TextView) findViewById(R.id.textView2)).setText()


}

}

        @Override
publicboolean onCreateOptionsMenu(Menu menu){
com.actionbarsherlock.view.MenuInflater inflater =
      getSupportMenuInflater();
inflater.inflate(R.menu.menu,(com.actionbarsherlock.view.Menu)
      menu);
returnsuper.onCreateOptionsMenu(menu);

}


        @Override
publicboolean onMenuItemSelected(int featureId, MenuItem item){

int itemId = item.getItemId();
switch(itemId){
case R.id.menu_home:
                    Intent i =newIntent(android.sqllite.Text.this,
            android.sqllite.Menu.class);// Search.class
startActivity(i);

case R.id.menu_back:
finish();
break;

case R.id.menu_search:

break;

}

returntrue;
}
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:debuggable="true"
package="my.android.sqllite"
android:versionCode="1"
android:versionName="1.0">

<!-- Build Target -->
<uses-sdk android:targetSdkVersion="14" android:minSdkVersion="8"/>
```

```
<application
        android:icon="@drawable/logo2"
        android:label="@string/app_name">
<activity
android:name="android.sqllite.TestDatabaseActivity"
            android:label="@string/books"
            android:theme="@style/Theme.Sherlock.Light.DarkActionBar">

</activity>

<activity
android:name="android.sqllite.Menu"
            android:label="@string/app_name"
android:maxLines="10"
android:scrollbars="vertical"
            android:theme="@android:style/Theme.Black.NoTitleBar">

<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>

</activity>

<activity
android:name="android.sqllite.Text"
            android:label="@string/app_name"
android:maxLines="10"
android:scrollbars="vertical"
            android:theme="@style/Theme.Sherlock.Light.DarkActionBar"
>
</activity>

<activity
android:name="android.sqllite.TextViewer"
            android:label="@string/app_name"
android:maxLines="10"
android:scrollbars="vertical"
            android:theme="@style/Theme.Sherlock.Light.DarkActionBar"
>
</activity>

<activity
android:name="android.sqllite.ScrollingViewPager"
            android:label="@string/app_name"
android:maxLines="10"
android:scrollbars="vertical"
            android:theme="@style/Theme.Sherlock.Light.DarkActionBar"
>
</activity>


<activity android:name="android.sqllite.Search"
android:launchMode="singleTop">
<intent-filter>
<action android:name="android.intent.action.SEARCH"/>
</intent-filter>
```

```
<meta-data android:name="android.app.searchable"
                   android:resource="@xml/searchable"/>
</activity>

<activity
android:name="android.sqllite.ChaptersDatabaseActivity"
            android:label="@string/app_name"
android:maxLines="10"
android:scrollbars="vertical"
            android:theme="@style/Theme.Sherlock.Light.DarkActionBar">
</activity>



</application>

</manifest>
```

fragmenttext.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

<android.support.v4.view.ViewPager
    android:id="@+id/conpageslider"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

```
</LinearLayout>
```

listlayout.xml

```
<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
android:layout_height="wrap_content"
android:layout_width="fill_parent"
android:textSize="18sp"
android:paddingLeft="6dip"
android:paddingRight="6dip"
android:gravity="center_vertical">

</TextView>
```

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<LinearLayout
        android:id="@+id/group"
android:layout_width="wrap_content"
android:layout_height="wrap_content">

</LinearLayout>

<ListView
        android:id="@android:id/list"
android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:text="@string/hello"/>

<android.support.v4.view.ViewPager
        android:id="@+id/pager"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
/>

</LinearLayout>
```

menulayout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="match_parent"
android:layout_gravity="center"
android:orientation="vertical"
android:paddingTop="150dp"
    android:background="@drawable/mainbackground">
```

```
<Button
        android:id="@+id/button3"
android:layout_width="200dp"
android:layout_height="65dp"
android:layout_gravity="right"
        android:text="@string/menu_start"
android:layout_marginRight="10dp"
android:layout_marginTop="10dp"/>


<Button
        android:id="@+id/button2"
android:layout_width="200dp"
android:layout_height="65dp"
android:layout_gravity="right"
        android:text="@string/search"
android:layout_marginRight="10dp"
android:layout_marginTop="10dp"/>
<Button
        android:id="@+id/button1"
android:layout_width="200dp"
android:layout_height="65dp"
android:layout_gravity="right"
        android:text="@string/about"
android:layout_marginRight="10dp"
android:layout_marginTop="10dp"/>



</LinearLayout>
```

searchlayout.xml

```
<?xml version="1.0" encoding="UTF-8"?>


<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<ListView
        android:id="@android:id/list"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:text="@string/hello"/>

<Button android:text="Click me!"
     android:id="@+id/BtnToClick"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:onClick="myClickHandler">
</Button>


</LinearLayout>
```

text.xml

```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:scrollbars="horizontal" android:id="@+id/ScrollView">

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<TextView
        android:id="@+id/textView1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:text="@string/testing"
/>

<TextView
        android:id="@+id/textView2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
        android:text="@string/testing"
/>

</LinearLayout>

</ScrollView>
```

textfragmentlayout.xml

```
<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tv"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center">

</TextView>
```

menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/menu_home"
          android:icon="@drawable/home"
          android:title="@string/menu_test"
android:showAsAction="always"/>
<item android:id="@+id/menu_back"
          android:icon="@drawable/back"
          android:title="@string/menu_test"
android:showAsAction="always"/>
<item android:id="@+id/menu_search"
          android:icon="@drawable/search"
          android:title="@string/menu_test"
android:showAsAction="always"/>
</menu>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

<string name="hello">Hello World, testingActivity!</string>
<string name="app_name">Wesley Commentaries</string>
<string name="books">Books </string>
<string name="testing">This is where the text goes</string>
<string name="search_hint">Please Enter Search</string>
```

```
<string name="app_name2">Android.sqllite.Search</string>
<string name="menu_test">testing</string>
<string name="menu_start">Commentaries</string>
<string name="search">Search</string>
<string name="about">About</string>

<style name="ButtonText">
<item name="android:layout_width">fill_parent</item>
<item name="android:layout_height">wrap_content</item>
<item name="android:textColor">#ffffff</item>
<item name="android:gravity">center</item>
<item name="android:layout_margin">3dp</item>
<item name="android:textSize">30dp</item>
<item name="android:textStyle">bold</item>
<item name="android:shadowColor">#000000</item>
<item name="android:shadowDx">1</item>
<item name="android:shadowDy">1</item>
<item name="android:shadowRadius">2</item>
</style>

</resources>
```