ANDROID QUICK BROWSER

by

ARAVIND REDDY KOTHAKAPU

B.Tech, Jawaharlal Nehru Technological University, 2013

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

2015

Approved by:

Major Professor
Dr. Daniel Andresen

Copyright

.

Abstract

People are more inclined to use mobiles instead of other computing devices like desktops and laptops due to their portability and ease of use. A number of desktop applications have become available as mobile applications in response to the increasing market demand. Android is one of the largest and most popular open source platforms that offers developers complete access to framework APIs in order to develop innovative mobile applications

The objective of this project was to develop an Android application for daily users of Android mobiles. This application advantageously allows users to search the internet on their phone, save preferred links, and prioritize outputs based on saved links. The application uses the Google search engine as its primary search engine and customizes links for the user.

Table of Contents

List of Figures

Table of Tables

Acknowledgements

I would like to express my heart-felt gratitude to my **Parents** without whom I would not have been privileged to achieve and fulfill my dreams.

I am grateful to my major professor, **Dr. Daniel A. Andresen** who had the major hand in enabling me to do this project. I profoundly thank **Dr. Torben Amtoft and Dr. Mitchell Neilsen** for their help and for serving on my committee.

I would like to express my gratitude to all the people behind the scenes who helped me to transform an idea into a real application. Last but not least I would like to acknowledge the academic support received from the staff and students of Kansas State University

Chapter 1 - Introduction

1.1 Problem Definition

Android Quick Browser is a search engine that reduces the amount of user's interaction with the search interface. This application understands the user's needs by capturing the interests of the users in their profiles and provides them with highly relevant information. The application also stores the searches for later reference.

1.2 Existing System

In the existing system, a user must browse for a particular website and then search for particular content on that website. Android mobiles currently do not contain a search engine that gives multiple links for each website at a single instance. Main challenges to these systems include handling the enormous amount of dynamic data (*data storage problem*), presenting results in real time (*time efficiency problem*), and fetching relevant ranked results to the queries (*relevancy decision problem*).

1.3 Proposed System

The *Android Quick Browser* retrieval system performs a word-by-word match of a query, and then a feature personalized search aims to customize search results for each user according to local context. Location intent queries are also studied. Android quick search is a very simple search engine consisting of a simple indexing system and query processing system is presented by capturing the user's interest in their profiles and provides highly relevant information. The application also stores the searched data for reusability.

1.3.1 Modules

The following are the modules in this application

1.3.1.1 Global Search Module

In this module, a user can search for any website using a keyword and then results are displayed horizontally. Each result has one link with the actual URL and another link with the

title of that page. The URL link is to view the corresponding content or webpage and the title of the page, or keywords link, is to save a particular link for future reference.

1.3.1.2 Local Search Module

In the local search module, a user can access URLs that are saved during the global search by searching the same string or keyword. The local search module also supports multiple link openings.

1.3.1.3 Link Deposit Module

In the link deposit module the user conducts a global search and results are displayed horizontally. If a user wants to save a link for future reference, the user may save that link by clicking the upper link so that it is stored in the link deposit module. It acts as temporary storage for viewing multiple links during a global search, which is done internally. When a user wants to extract the saved link from the link repository, the local search module is utilized.

1.3.1.4 Web Integration Module

The web integration module connects all outputs of all activities and components essential for carrying out the web project as a whole. The web integrator generates the client that the project will comply with the agreed specifications. The web integrator is also responsible for coordinating all necessary activities. The web integrator, the client's primary contact in the project, must show sufficient capability in project management.

Chapter 2 - Related Work

2.1 Proposed System over Traditional Existing System

This section describes how the proposed system effectively solves problems that exist in the current system.

A common problem experienced by current browser users is difficulty in navigating between webpages or a webpage and search results. Users must return to the search results by pressing the back button or they must start a new search. Many users may not know the exact desired URL, requiring continuous surfing in order to obtain relevant results. Because end users would benefit from some glance of what exactly that particular result contains, the concept of the Android Quick Browser was developed, which provides a glance of every link on a single page to increase user understanding of links without navigation problems. Figures 1 and 2 illustrate this concept using screenshots of the traditional existing system and the proposed system. As shown in Figure 1, home screens of both systems are similar, with the exception that the proposed system has two searches: search on web (i.e., content search) and search locally from the database (i.e., local search). The local search option is helpful when a lot of saved links or bookmarks are present in the browser, which is lacking in the current system. The proposed system also is divided into search results and glances of selected results.

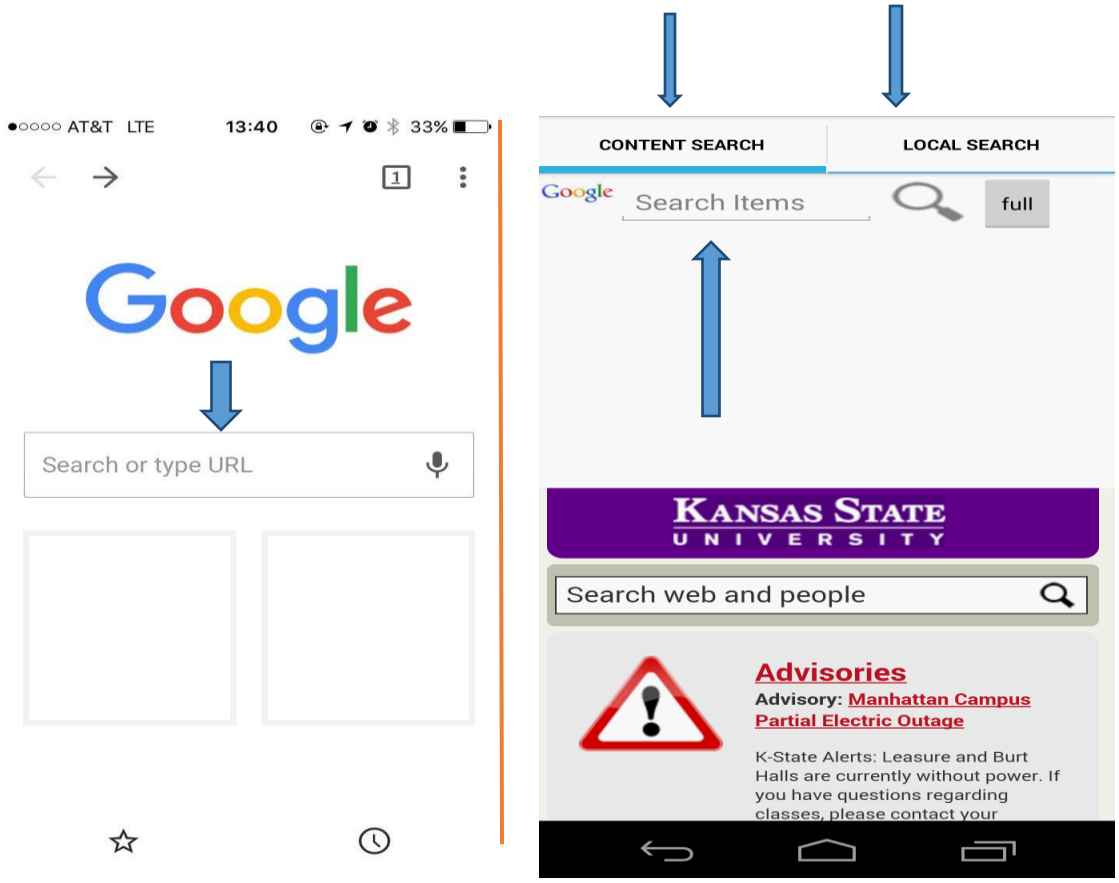

Figure 1 Comparison of home screens

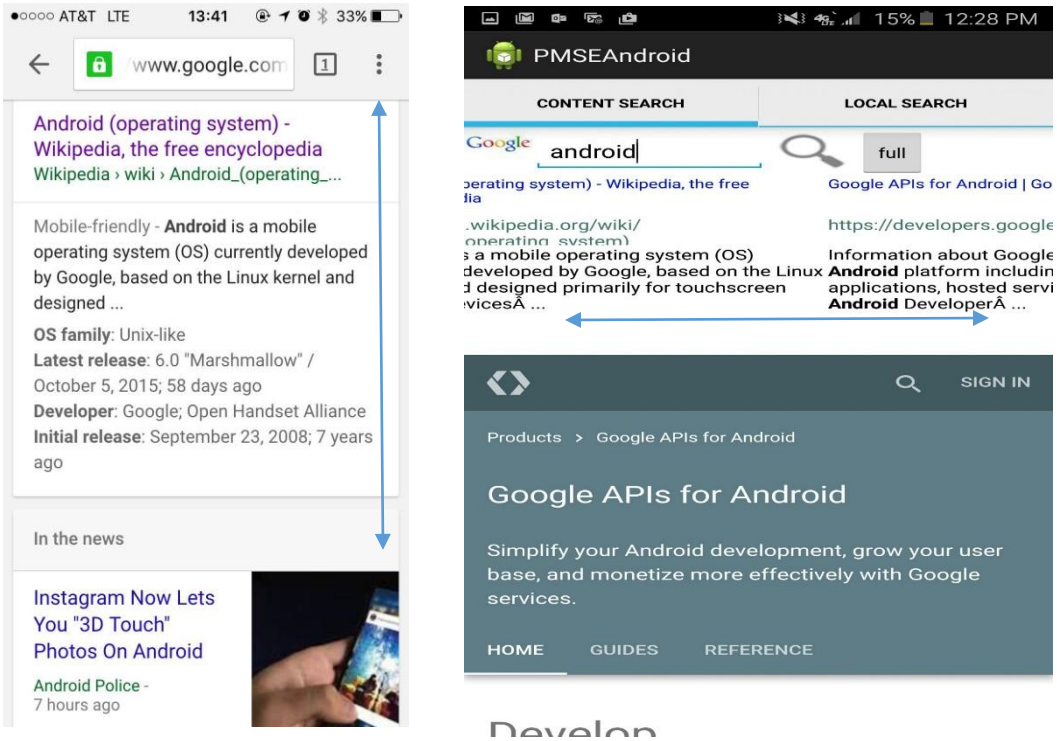

Figure 2 Comparison of search results

As shown in Figure 2, the existing system displays results vertically, requiring users to scroll up or down in order to surf all results. In the proposed system, search results are displayed horizontally, allowing users to scroll left or right to surf results. In the proposed system another beneficial feature that makes the proposed system distinct from traditional browsers is that users have a glance of relevant information of each result in the bottom half of the screen, which is especially helpful when a user is unsure of an exact search. A challenge associated with this feature, however, is the application experience of small-screen device users; because the screen is divided into halves, users may have difficulty reading results on small-screen devices. Therefore, a full-screen mode was developed for this application. After selecting the desired result, the user can turn on full-screen mode to have better experience.

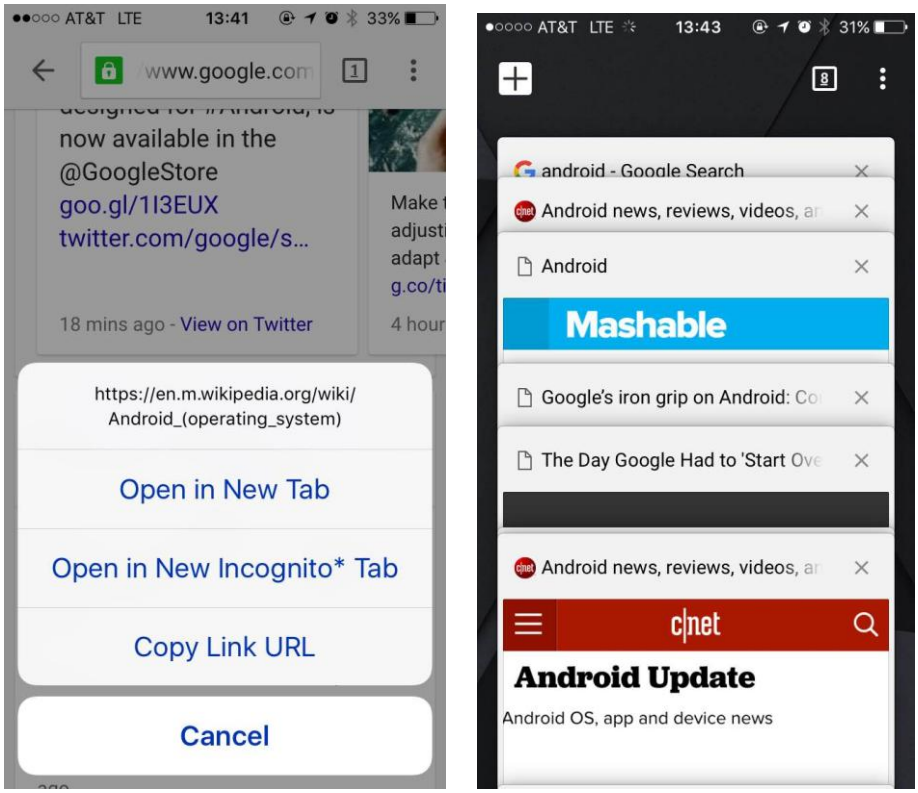

Figure 3 Switching between results

Figure 3 demonstrates the difficulty of navigating between results and webpages in the current system. A user must either open a particular link in a new tab or click on link in the current tab in order to navigate or glance at a specific link. If a user opens in the same tab and the

result is unsatisfactory, navigation becomes tedious because the user must go back and select a new result. If the user opens the other way in new tab to see the result which resulting in open of many tabs and end up in performance issues. Therefore, the Android *Quick Browser* was built to improve user experience and increase application performance by offering a glance of many results in a single screen.

Chapter 3 - Requirement Analysis

Users often experience difficulty when searching for a specific webpage and are often required to return to the search results page in order to navigate to other webpages of relevant information. These tasks can be tedious, especially on mobile phones, which do not offer as much flexibility as a computer for navigating websites. Therefore, the concept of an Android application that would provide a solution to this common problem and would increase knowledge of Android development was conceived.

One of the hardest aspects of application development was determination of the exact requirements or use cases of the intended application. Thus, the author surveyed peers in regards to desired application features that would solve the problem of frequent navigations among search results and webpages. Input received recommended a feature in which users could glance the webpage of a search result in the same activity or on the screen containing the search results. Consequently, the first version of the developed application allowed a user to view a result on the same screen, meaning that the website loaded on the same screen as the displayed search results. The second part of the screen dynamically changed to selected results from the search results.

The first version of the application was tested by the initial peers in order to obtain additional input or suggestions regarding the existing version. A follow-up suggestion was implementation of a local search option in which the user may bookmark websites for future use. Consequently, the local search option in this application was developed, allowing a user to save a relevant website locally in the database. Another suggestion was to implement a full-screen preview of the target website, especially for users with small screens who are unable to view search results and the target website on the same screen. A feature that would enable users to view the target website in a majority of the screen space was recommended. Therefore, in the final version of the application, a button was included that displayed the target website in almost three-fourths of the main screen. Requirements for developing this application were also considered, as listed in the following section.

3.1 Requirement Specification

For running the application:

- ✓ Android Smart Phone 2.2 or higher
- ✓ 4.0MB space for execution

Chapter 4 - System Architecture and Design

4.1 Architecture

Figure 4 shows an architecture diagram of the system in which the principal parts are represented by blocks and connected lines represent relationships of the blocks. This diagram is typically used to provide higher level abstract of the system with less emphasis on implementation and more emphasis on understanding the overall structure of the system.

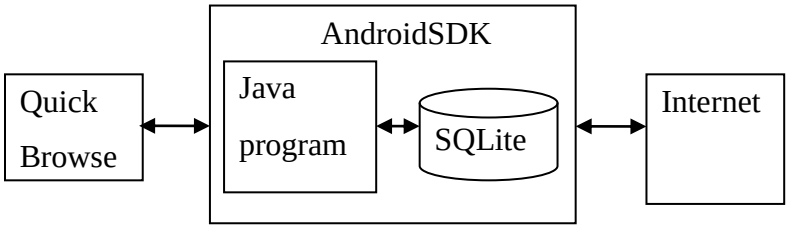

Figure 4 Architecture

An Android *Quick Browser* user interacts with the user interface, which typically has a search text box in which a user must enter a keyword for the search. The request is processed, results are extracted and stored in SQLite Database, and a java program places it on the user interface.

4.2 Data Flow Diagrams

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. It typically illustrates how data is processed by a system in terms of inputs and outputs. Also, it focuses on flow of information, where data comes from, where it goes, and how it gets stored. [1] Figure 5 illustrates a DFD of the system. The rectangles represent inputs and outputs, the circle represents the function, and the arrows represent the flow. A user can use input to search in the search engine, causing the search engine to interact with the URL source,

obtain results, and process those results to the user. For a saved URL, the search engine interacts with the database to provide results to the user.

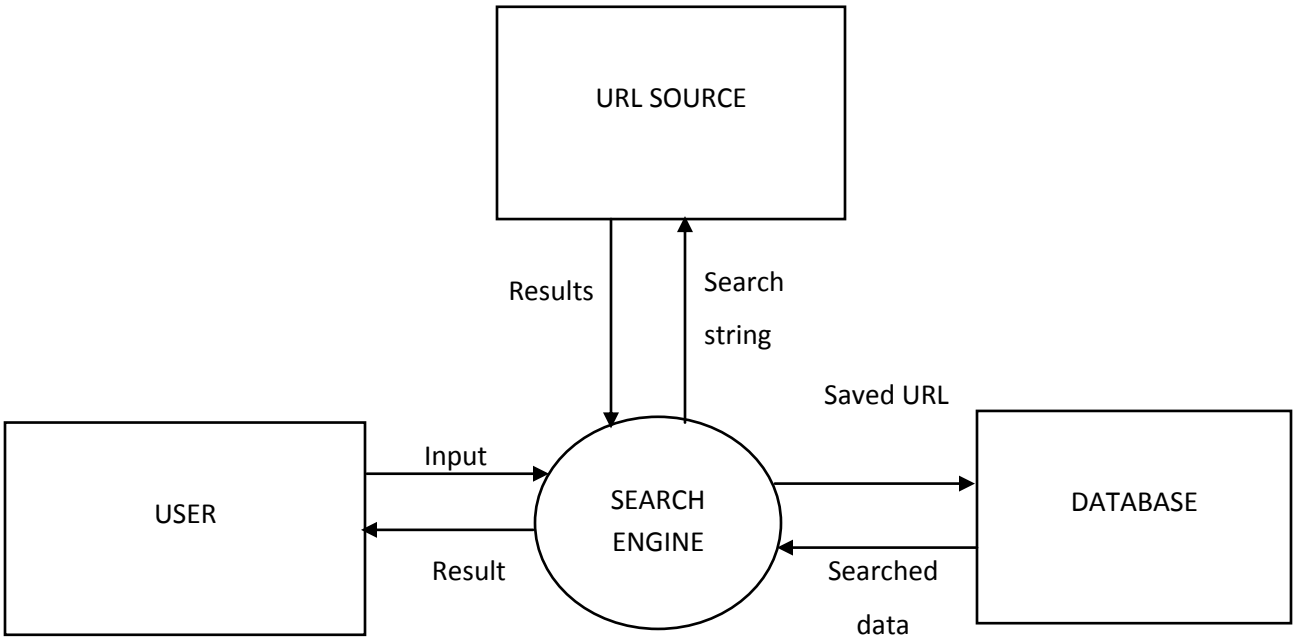

Figure 5 Data flow diagram

4.3 Unified Modeling Language

The unified modeling language (UML), a general-purpose, developmental, modeling language in the field of software engineering, is intended to provide a standard way to visualize the design of a system. Two UML diagrams are commonly used: structural and behavioral. Structural diagrams emphasize the things that must be present in the modeled system. Examples of structural diagrams include class diagrams, component diagrams, and deployment diagrams. Behavioral diagrams emphasize what must happen in the modeled system. Examples of behavioral diagrams include activity diagrams, sequence diagrams, and use case diagrams. [2] Implementation of structural and behavioral diagrams is described in the following sections.
\

4.3.1 Use Case Diagram

A use case diagram represents user interaction with the system, demonstrating the relationship between user and use cases. Figure 6 shows a use case diagram of the system in which User1, User2, and DataBaseAdmin are the users and Search Engine, Global Search, Content, Saved to Local Search, Access, and Display are the use cases. User1 can interact with Search Engine or Global Search to search for a result and then interact with the content after results are displayed and save to a local search that interacts with that use case. Similarly, User2 can interact with Access and Display, and DataBaseAdmin can interact with Content and Saved to Local Search. [3]

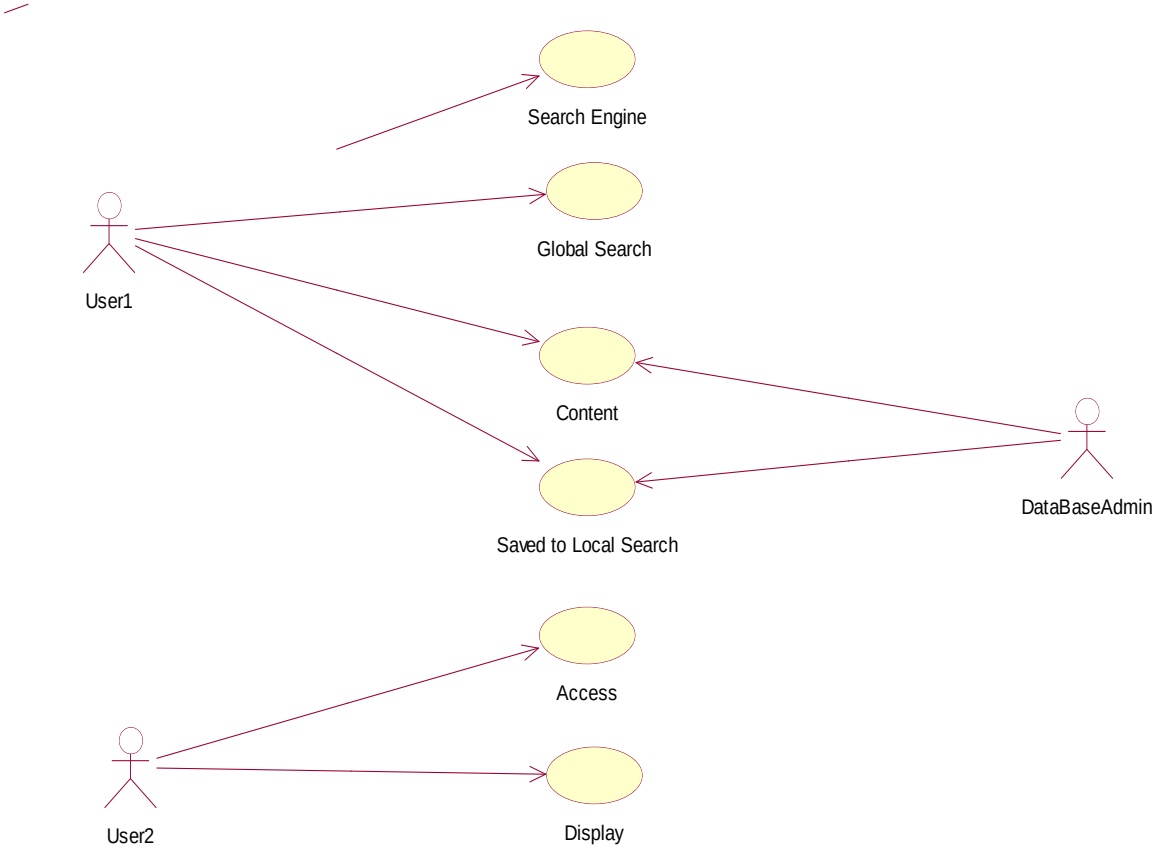

Figure 6 Use case diagram

4.3.2 Class Diagram

A class diagram is a static structure diagram that describes the structure of a system by showing classes, their attributes, operations, and relationships among objects. [4] Figure 7 shows a class diagram of the system with classes MainActivity, GoogleSearch, MyDBHelper, LocalSearch, RetrieveJsonData. Their attributes and functions are bound in the each class. The MainActivty class has a Google Search and a Local Search. When something is searched in GoogleSearch, data is retrieved from RetrieveJsonData. LocalSearch uses MyDBHelper to get the data.

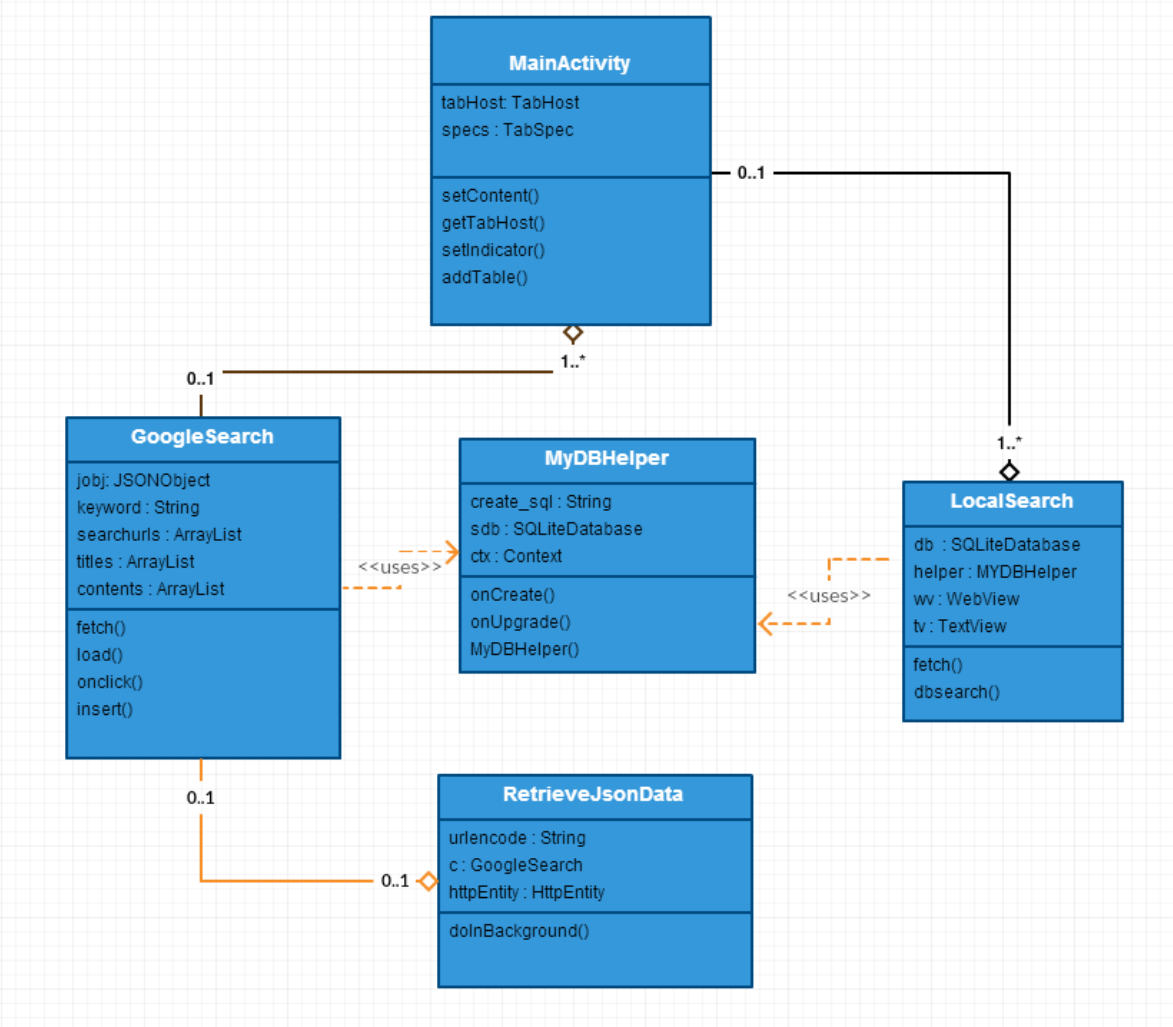

Figure 7 Class diagram

4.3.3 Sequence Diagram

A sequence diagram shows the method and order of processes' operations with one another. Figure 8 shows a sequence diagram of the system. [5] As shown in the figure, the flow of sequence diagram for this application starts with the user's request to Content Search, followed by results sent back as links to the user from Content Search. The user then selects the link, and that link can be saved on Local Search. The user also can send a request in order to directly interact with Local Search to access saved links.

.

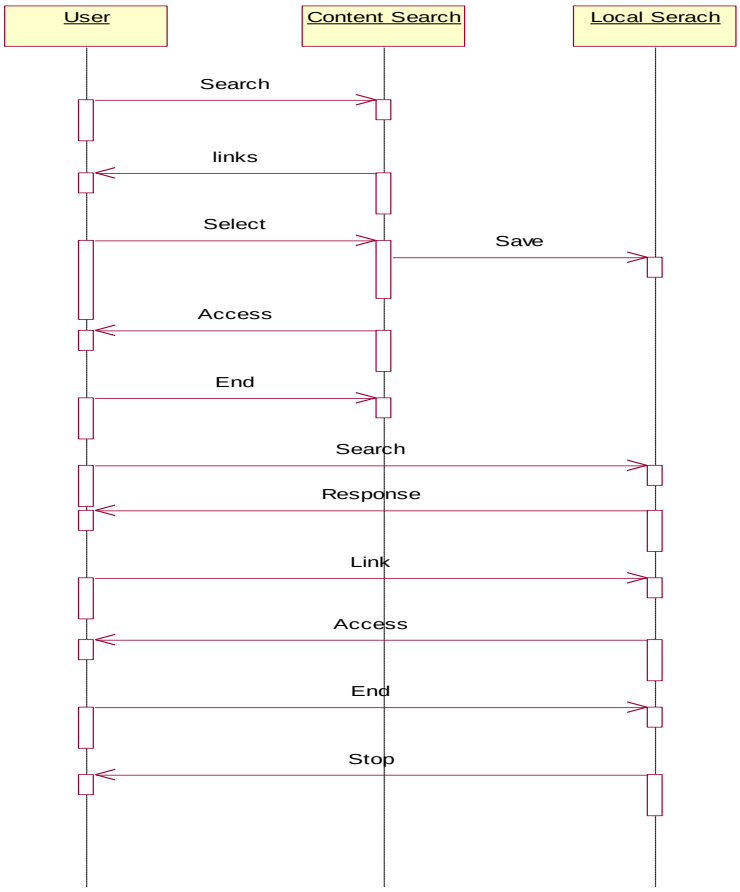

Figure 8 Sequence diagram

4.3.4 State Chart Diagrams

A state chart diagram describes various states of a system and state machine, as shown in Figure 9. The figure illustrates four states between start and final state in the application. The Start state is followed by the Search Engine state in which a keyword is entered in order to obtain results; these results comprise the Links state in which a user can save a link in the Local Search state. The user can also access Content and review on the search engine or end the search.

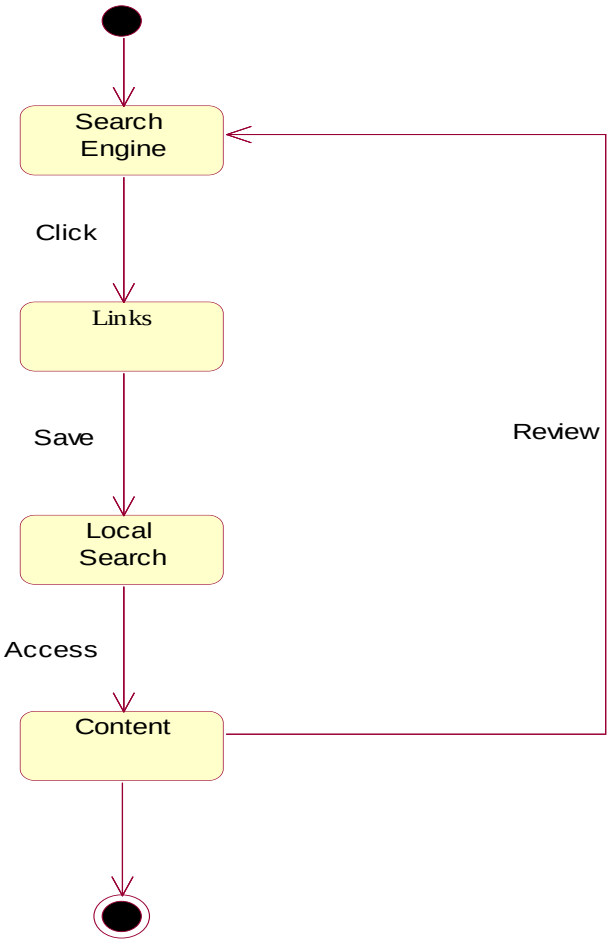

Figure 9 State chart diagram

4.3.5 Deployment Diagrams

A deployment diagram shows existing hardware components, distinguishes what software components run on each node, and illustrates how the pieces are connected. In the deployment diagram shown in Figure 10, [6] the Search Engine is the center hub that connects to all possible results, such as Global Search, Local Search, Content, Title, URL, and Home Page

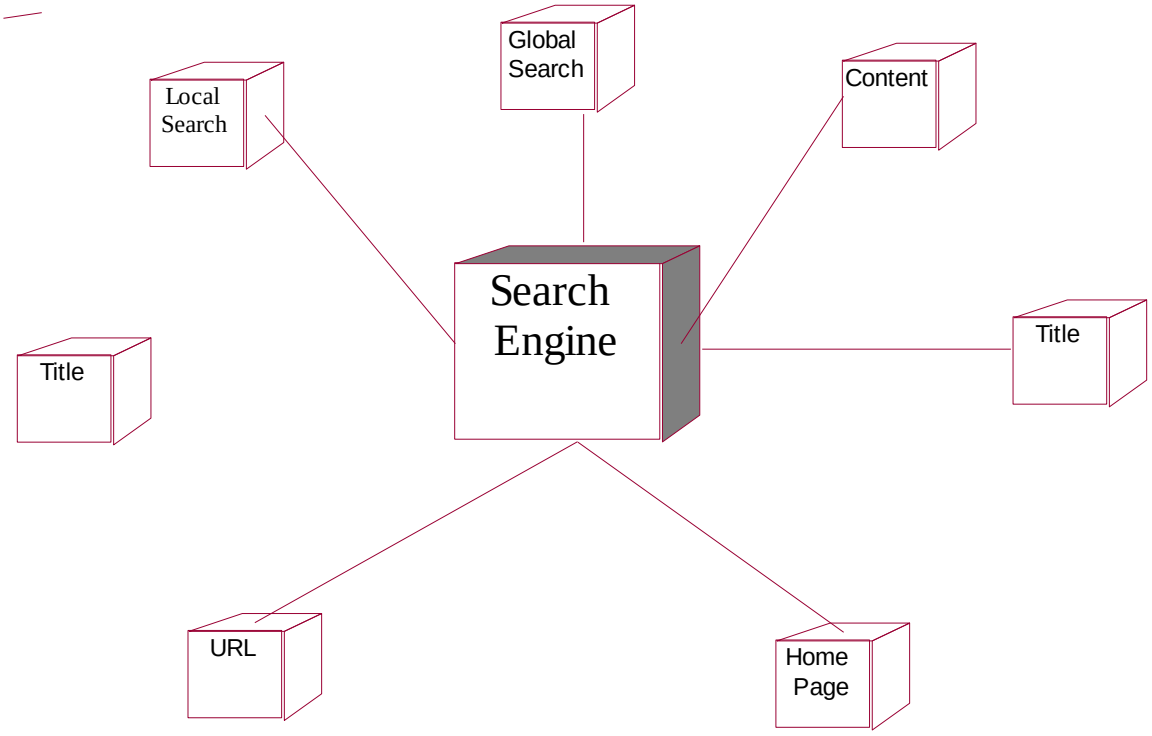

Figure 10 Deployment Diagram

Chapter 5 - Android Framework Components

5.1 AndroidManifest.xml

Every Android application contains an AndroidManifest.xml file in its root directory. This file contains a list of all activities, intents, and permissions included in the application. It also names the Java Package for the application which serves as a unique identifier for the application. All the highlighted items below are the elements in the manifest file, starting and ending with element manifest. [7]

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.aravind.pmseandroid" // unique identifier of the application
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.tf.quickbrowser.MainActivity"
            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="com.tf.quickbrowser.GoogleActivity"
            android:configChanges="orientation"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateHidden" >
        </activity>
        <activity
            android:name="com.tf.quickbrowser.LocationActivity"
            android:label="@string/title_activity_location" >
        </activity>
```

```
    </application>
</manifest>
```

Where

<*uses-sdk*> expresses an application's compatibility with one or more versions of the Android platform using an API

<*uses-permission*> requests permission for the application in order to achieve correct operation

<*activity*> declares an activity that implements part of the application's visual user interface

<*intent-filter*> specifies the types of intents that an activity, service, or broadcast receiver can respond to

<*action*> adds an action to an intent-filter

<*category*> adds a category to an intent-filter

<*application*> is the declaration of application.

5.2 Activities

An activity, represented as a single screen in the application, helps in viewing data, creating data, and editing data. A project can have one or more activities. Activities in this application are listed below.

```
public class MainActivity extends TabActivity {

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState)
    }
```

The MainActivity, the one of the activity in this project, contained the onCreate method within it. Execution begins from the MainActivity class, followed by LocationActivity and GoogleActivity classes to display Content and Local Search options.

```
public class LocationActivity extends Activity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
```

```
    }

        public void dbsearch(View v) {
     // Code Inside
     }

        public void fetch() {
      // Code Inside
     }


        public boolean onCreateOptionsMenu(Menu menu) {
       // Code Inside
      }

        public boolean onOptionsItemSelected(MenuItem item)
                {
       //Code inside

                }
}
```

Location activity in this project extended the Activity class and included the oncreate, dbsearch, fetch, onCreateOptionsMenu, and onOptionsItemSelected methods. This activity helps in working of Content Search feature. The fetch method is used fetch the data from the database, empty string search is handled here. The onCreateOptionsMenu inflates the menu and adds items to the action bar if it is present. The onOptionsItemSelected method handles action bar clicks, and the action method automatically handles clicks on the home button if s parent activity is specified in the manifest file.

5.3 Intents

An intent can described as it is an abstract description of an operation to be performed. These are generally used for intercommunication between activities within an application or between applications to activate broadcasts or to communicate with a background service. The function startActivity(intent) begins the intent. Every intent contains some action and data. As shown below, the intent from the manifest file with action of the main entry point and launcher says that the entry point should be listed in the application launcher.

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

Chapter 6 - Implementation

Once the user installs the application, the Android Quick Browser is ready to use on any Android device. Unlike traditional mobile browsers, it is very efficient and user friendly on any device. This application also has a great user interface that is divided into two halves: top half for search results and bottom half for content display of the selected link to provide an overview before entering the site, thereby simplifying the user's selection for appropriate data. One more challenge here is to provide readability for the user, as the screen is divided into 2 halves, it might not be good experience for small screen phone users. For that there is button for any search result to expand it to full screen. Another feature which is added here is to allow user to save the links which he/she likes and providing access to revisit it for future use.

6.1 Modules

The following are the different modules involved in this project.

6.1.1 Global Search Module

In the global search module, a user can search for any website using a keyword, and then results are displayed horizontally. Each result has one link with the actual URL and another link with the title of that page. The URL link is to view the corresponding content or webpage and the title of the page, or keywords link, is to save a particular link for future reference.

6.1.2 Local Search Module

In the local search module, a user can access URLs that are saved during the global search by searching the same string or keyword. The local search module also supports multiple link openings.

6.1.3 Link Deposit Module

In the link deposit module the user conducts a global search and results are displayed horizontally. If a user wants to save a link for future reference, the user may save that link by clicking the upper link so that it is stored in the link deposit module. It acts as temporary storage for viewing multiple links during a global search, which is done internally. When a user wants to extract the saved link from the link repository, the local search module is utilized.

.

6.1.4 Web Integration Module

The web integration module connects all outputs of all activities and components essential for carrying out the web project as a whole. The web integrator generates the client that the project will comply with the agreed specifications. The web integrator is also responsible for coordinating all necessary activities. The web integrator, the client's primary contact in the project, must show sufficient capability in project management.

6.2 Screen Shots

Figure 11 shows the home page of the quick search engine containing two searches: Content Search and Local Search in which a user can search data by entering the string in Content Search. The bottom half of the figure shows the default screen when the application is loaded. Later on replaced by the selected result.

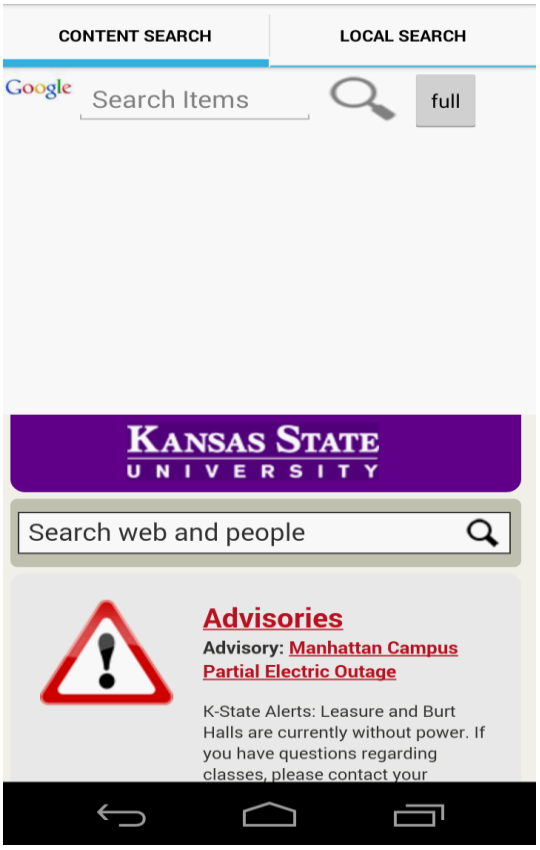

Figure 11 Homepage of App

Figure 12 shows a screenshot after a certain keyword has been entered and searched, illustrating a search in process. Once a search is complete, results are displayed horizontally.

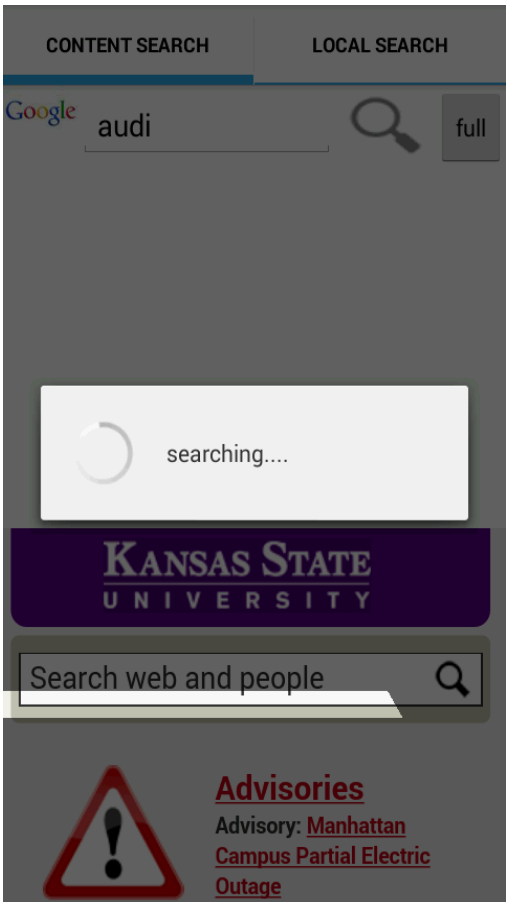

Figure 12 Search in progress

Once a user enters a keyword and searches, results are presented horizontally, as shown in Figure 13. A total of 16 results are displayed in the figure. By hovering or clicking on any specified URL, a glance of selected link content is shown. Depending on the user's requirement, the required result can be selected by dragging the results to the right or left.

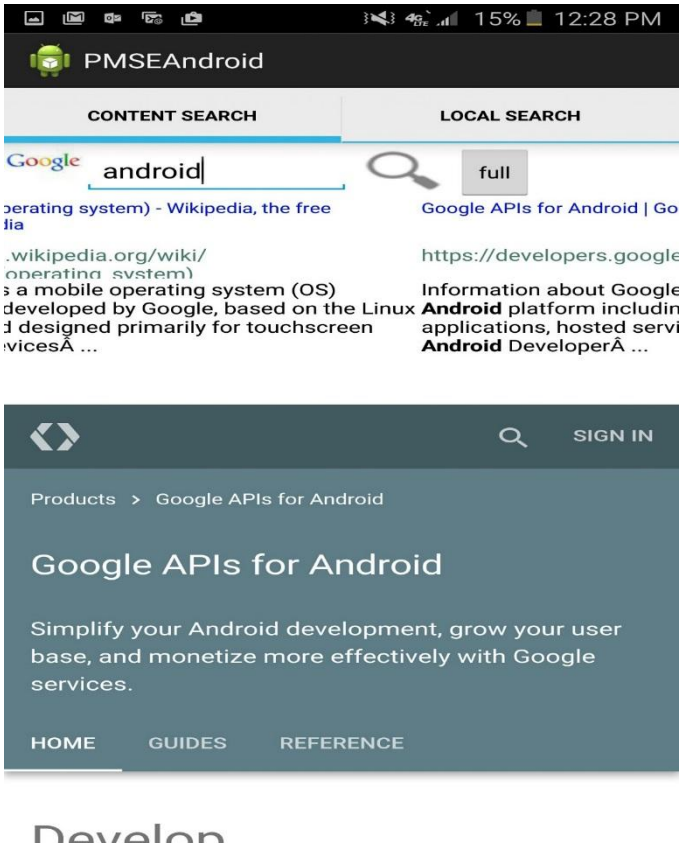

Figure 13 Search result

This particular full button was implemented in the Android Quick Browser in order to overcome the bad user experience for small-screen phones when the screen is divided into two halves. As shown in Figure 14, if a user clicks on the “full” button, the selected result is displayed in full screen.

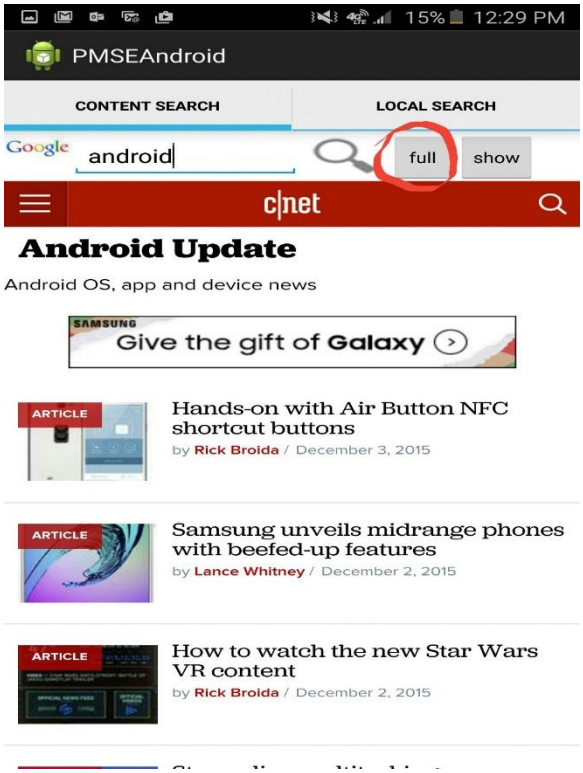

Figure 14 Full screen of selected result

When a user wants to return to results from the full-screen mode, this button becomes handy. As shown in Figure 15, the “show” button applies the two-screen mode: half of the screen with results and the other half with selected result so the user can also browse other results.

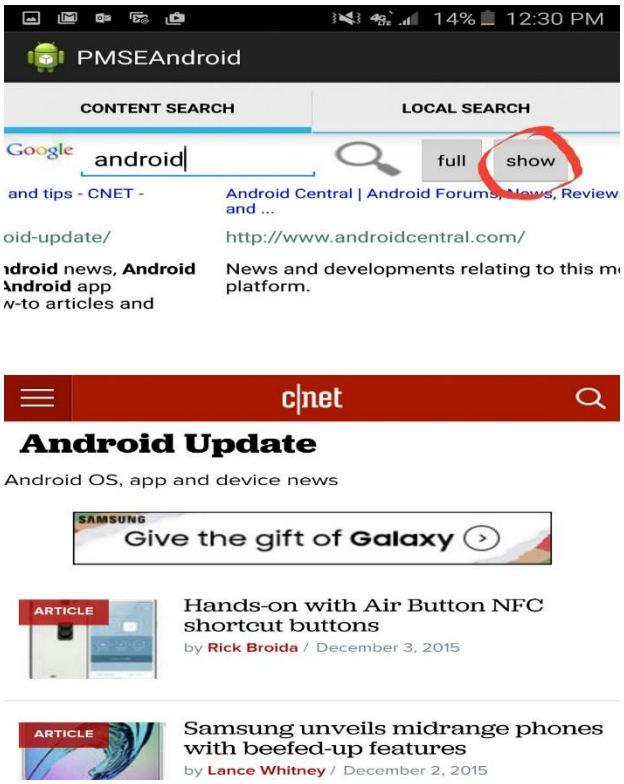

Figure 15 Show Button

Two links are contained in every search result: the actual URL and the actual name (keywords). If a user prefers a link and wants to refer to it later, for this an option to save the link is implemented, as shown in Figure 16. So, just a click away on the upper link i.e. actual name to save that.

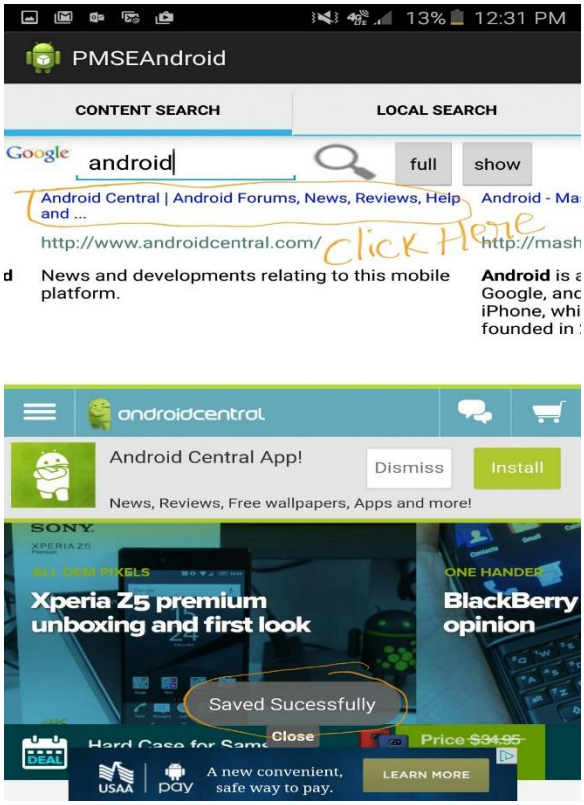

Figure 16 Save the link

The Local Search homepage, shown in Figure 17, takes the input and extracts results from the database. All previously saved results are stored in the database. A saved link can be searched revisited.

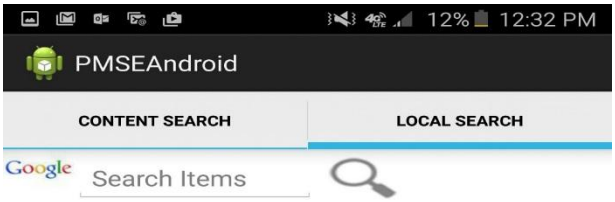

Figure 17 Local search homepage

Figure 18 shows a screenshot of a search results display in Local Search. All search results with the matching keyword are extracted from the database and displayed. The user can select or revisit any link.

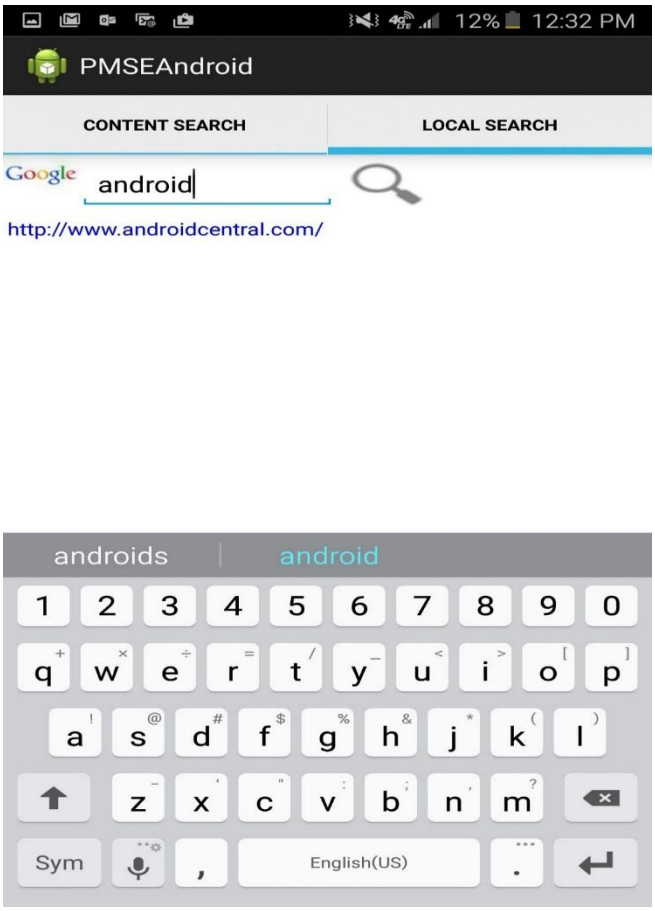

Figure 18 Local search results

Chapter 7 - Testing

Development of software systems involves a series of production activities in which opportunities for injection of human fallibilities are enormous. Errors may occur at the inception of the process, causing objectives to be erroneously or imperfectly specified, as well as in later design and development stages. Because of human inability to perform and communicate with perfection, software development is accompanied by a quality assurance activity. [9]

7.1 Unit Testing

Unit testing focuses on verification of the smallest unit of the software design module. The main goal is to ensure that every source statement and logic path has been executed correctly at least once. The output of this stage is the source code. [9]

Table 1 Unit Testing Case 1

Test case 1: Searching a string in Quick Search Engine.	**Priority (H, L):** High	
Test Objective: For searching.		
Test Description: User enters a string.		
Requirements Verified: Yes		
Test Environment: Application must be deployed in android mobile phone or emulator.		
Test Setup/Pre-Conditions:		
Actions	**Expected Results**	
The user enters a string.	Searches the related data to the entered string.	
Pass: Yes	Conditions pass: Yes	Fail: No
Problems / Issues: NIL		
Notes: Successfully Executed		

A test case is a software testing document that consists of event, action, input, output, expected result, and actual result. Large test cases may also contain prerequisite states or steps and descriptions. These steps can be stored

in a word processor document, spreadsheet, database, or other common repository. A test case should also contain a place for the actual result. [12]

Test case 2: Searching a string in Quick Search Engine.	**Priority (H, L):** High
Test Objective: For Searching.	
Test Description: User enters a string	
Requirements Verified: Yes	
Test Environment: Application must be deployed in android mobile phone or emulator.	
Test Setup/Pre-Conditions:	
Actions	**Expected Results**
The user enters a string.	Searches the entered string through Quick Search Engine and should provide the related data in 16 links in parallel.
Pass: Yes Conditions pass: Yes	Fail: No
Problems / Issues: NIL	
Notes: Successfully Executed	

Table 2 Unit Testing Case 2

Test case 3: Saving the Desired link to Local Search from Global Search.	**Priority (H, L):** High
Test Objective: For searching the string in Quick Search Engine.	
Test Description: User enters a string.	
Requirements Verified: Yes	
Test Environment: Application must be deployed in android mobile phone or emulator.	
Test Setup/Pre-Conditions:	
Actions	**Expected Results**
The user selects the desired link among the 16 provide links.	Add the book into database
Pass: Yes Conditions pass: Yes Fail: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Table 3 Unit Testing Case 3

Test case 4: Searching the desired link in Local Search.	**Priority (H, L):** High
Test Objective: For searching the string in Quick Search Engine.	
Test Description: User enters a string.	
Requirements Verified: Yes	
Test Environment: Application must be deployed in android mobile phone or emulator.	

Test Setup/Pre-Conditions:	
Actions	**Expected Results**
The user enters the string which is already saved in Database for reuse.	Should provide the content, title, url, homepage for the related string.
Pass: Yes Conditions pass: Yes	Fail: No
Problems / Issues: NIL	
Notes: Successfully Executed	

Table 4 Unit Testing Case 4

7.2 Integration Testing

In integration testing, individual modules are combined and tested as a group after unit testing is complete. For this research, individual modules were combined and tested as a group, resulting in pass. [9]

7.3 Compatibility Testing

Compatibility testing is a nonfunctional test of an application in order to determine application compatibility within various environments. The Android Quick Browser was run in environments, such as AVD, and Android devices with various screen sizes in order to ensure that this application worked fine on every device and was adaptable to any size screen. [9]

7.4 User Interface (UI) Testing

User interface (UI) testing is used to ensure that users do not encounter unexpected results when interacting with user interface. A uiautomatorviewer tool provided with the Android

SDK was used for UI testing of the Android Quick Browser. This tool provides convenient visual interface to test individual UI components displayed on the device. Figures 19–22 show screenshots of UI testing results. For Content and Local Search, clickable, enabled, and focusable were true; for Search Text Field, even focused was true. [10]

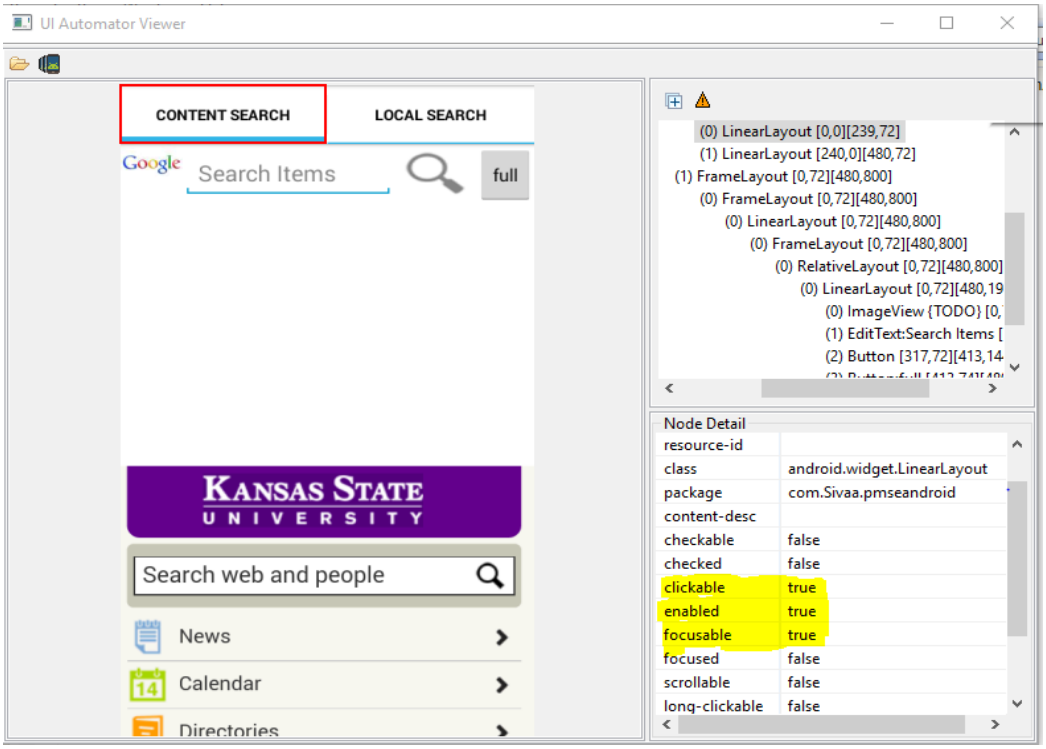

Figure 19 UI testing Content Search

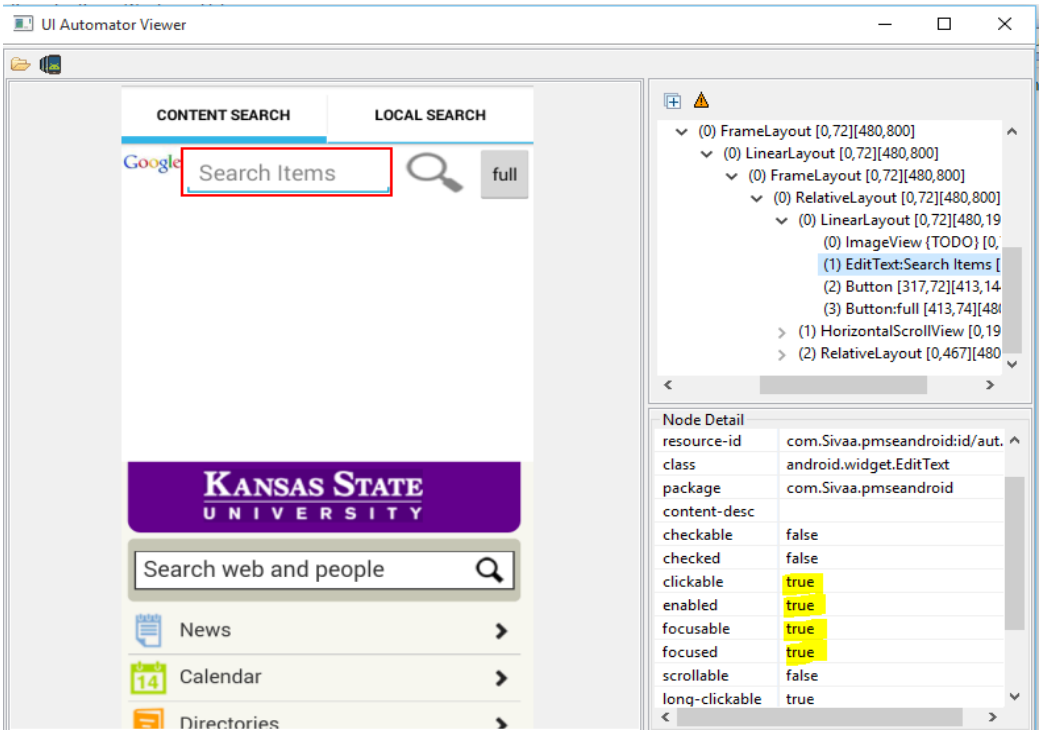

Figure 20 UI testing Search Text Field

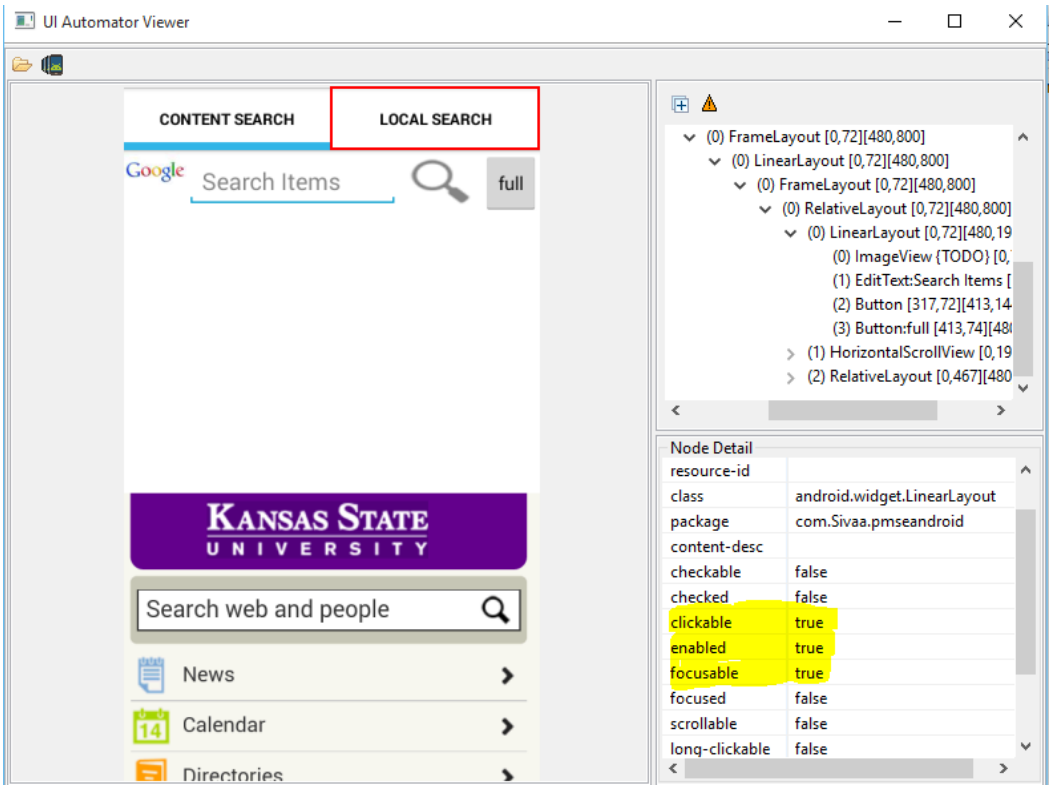

Figure 21 UI testing Local Search

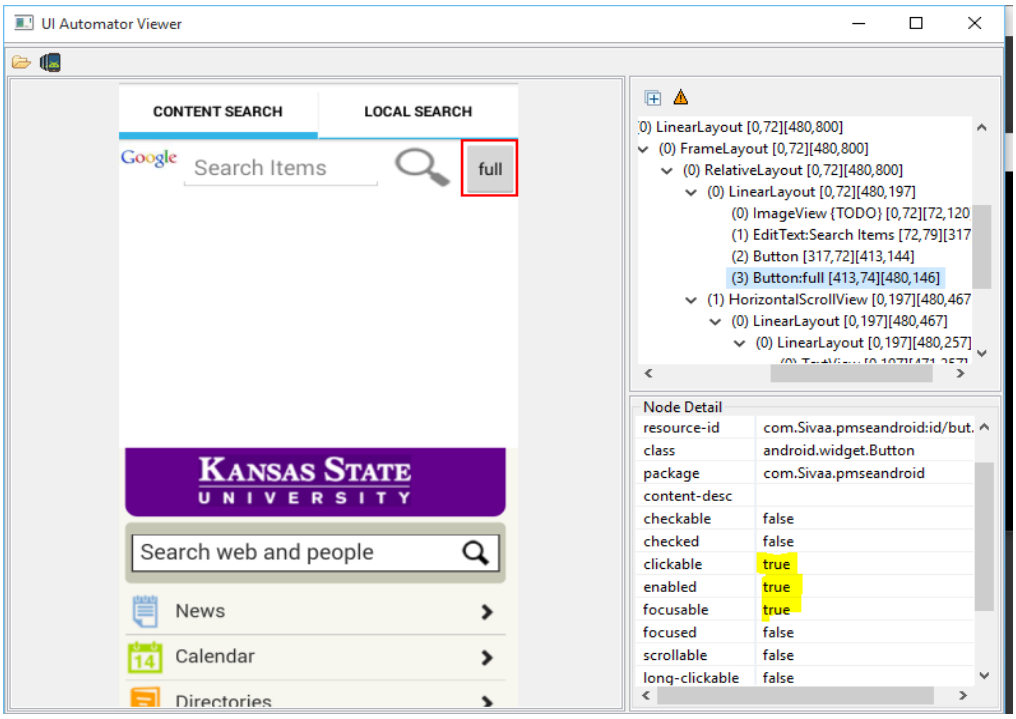

Figure 22 UI testing full button

7.5 Performance Testing

Performance testing is used to determine reliability, responsiveness, and throughput of a system. For performance testing of the Android Quick Browser, the application was run under various conditions, such as different networks (2G, 3G, 4G, and LTE) and different devices, such as emulators and various-sized devices. Every test case ran fine. In devices with high-speed networks, such as LTE, 4G, and 3G, the application ran very smoothly, but in slow-speed networks, such as 2G and lower, the payload of the application was high, resulting in some delay. The application was also run on various performance devices, demonstrating satisfactory performance in every device. When I tried to run on Eclipse emulator its taking some time to load the application, but I found the alternate emulator which loads and runs really fast i.e. genymotion. Overall, performance testing of the Android Quick Browser application passed in a majority of the devices. [11]

Chapter 8 - Summary

I started this project as a naïve user who has very less knowledge on Android and Web services. The main motivation of this project is to learn Android that is how I started this project. After I decided to do some project in android, *Quick Browser* is the another motivation which is the solution of some problems in the existing browsers. Developing this application helped me gain a goof amount of knowledge in the android development, also in web services as my project used lot of web services.

Programming Language	Lines of Code	Description
Java	2657	Project contained 5 java files which has 2657 lines of code, including queries on SQLite Database and JSON web services
XML	1008	Project hads 4 xml files with 1008 lines of code.

Table 5 Line of Count

There is lot of scope for enhancements on this application to make it potential. Some future enhancements include adding a GPS feature to initially display relevant information, tracking search history and corresponding display of results, implementation of maps, separate searches for videos and photos, and search by voice. These features could be incorporated into the Android Quick Browser to make it more competitive with the current existing browser.

Chapter 9 - Conclusion

This project presented a unique search engine for effective searching through mobile interface. The engine adopted a combination of a content-based method for retrieval that utilized low-level visual characteristics of multimedia material and a hybrid method in order to offer a complete result set to the user. In addition, easy access and system portability increased system performance in a better manner.

Future work should include extension of the hybrid search engine and integration of additional cultural content. Investigation also is currently underway for the addition of a semantic recommendation engine that could automatically make query suggestions to the user.

Chapter 10 - Future Improvement

The Android Quick Browser application can be enhanced in the following ways:

- A revolutionary search engine could use the GPS from the Android device to determine which results should initially be presented.
- A searching algorithm could be created that would process more results based on the GPS and the user's full history based on the device's unique entry or the Google account to determine initial results.
- Separate searches for videos and photos would provide additional relevant information to end users.
- Search by audio could be another cool feature.

Chapter 11 - References

[1] “Data flow diagram”, Online, November 20, 2015
https://en.wikipedia.org/wiki/Data_flow_diagram

[2] “Unified Modeling Language”, Online, November 20, 2015

https://en.wikipedia.org/wiki/Unified_Modeling_Language

[3] “Use Case Diagram”, Online, November 20, 2015

https://en.wikipedia.org/wiki/Use_Case_Diagram

[4] “Class diagram”, Online, November 20, 2015

https://en.wikipedia.org/wiki/Class_diagram

[5] “Sequence diagram”, Online, November 20, 2015

https://en.wikipedia.org/wiki/Sequence_diagram

[6] “Deployment diagram”, Online, November 20, 2015

https://en.wikipedia.org/wiki/Deployment_diagram

[7] “Android Developers”, Online, November 23, 2015

http://developer.android.com/index.html

[8] “UML Diagrams”, Online, November 24, 2015

http://www.smartdraw.com/uml-diagram

[9] “Software Testing”, Online, November 25, 2015

https://en.wikipedia.org/wiki/Software_testing

[10] “GUI testing”, Online, December 08, 2015

http://whatis.techtarget.com/definition/GUI-testing-graphical-user-interface-testing

[11] “Fundamentals of Web Application Performance Testing”, Online, December 08, 2015

https://msdn.microsoft.com/en-us/library/bb924356.aspx