User Interface Android Application for A Context-Centric Model for Organizing A Personal Knowledge Network

Nagendra Vemulapalli

Problem Report submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Approved by
Dr. Vinod Kulathumani, Ph.D., Chair
Dr. Y.V. Reddy, Ph.D.
Dr. Elaine Eschen, Ph.D.

Lane Department of Computer Science and Electrical Engineering Morgantown, West Virginia 2013

Keywords: Vijjana, Knowledge Network, User Interface, Knowledge Objects, Middleware Server, Context
Productivity of a knowledge-Worker is directly related to the ease with which knowledge buried in a variety of documents can be accessed. When the semantic relationships between the documents are not axiomatic, the search process by search engine and the common method of organizing documents as a pecking order of folders which are tagged with meaningful names is highly inefficient. A simple model for organizing documents as a semantic network is described in this paper. The semantic network uses tags which are automatically generated by content analysis augmented by user markup when warranted [1]. In this model, the documents are semantically related to one or more contexts. A user can describe the work-process as a shallow hierarchy of contexts. All the relevant documents could be displayed on the request of the user through a client which is an Android application. This report describes the design of an Android application which acts as a client to provide access to users over the knowledge network described above.
ACKNOWLEDGEMENTS

I would like to thank Dr. Vinod Kulathumani for his constant support and invaluable guidance in many aspects of my research.

I would like to thank Dr. Y.V. Reddy for his encouragement and enthusiasm which motivated me to work with the Vijjana team and share my innovative thoughts to implement them for the development of the Vijjana architecture.

I would like to extend my thanks to Dr. Elaine Eschen for being on the committee and supporting the project with their suggestions and encouragement.

I would like to thank the Vijjana team and all my friends, roommates for their immense support in various stages of my research project.

Finally, I am deeply thankful to my beloved parents and family for their encouragement and support in all the things I do.
# TABLE OF CONTENTS

ABSTRACT.................................................................................................................................................. ii

ACKNOWLEDGEMENTS.................................................................................................................................. iii

TABLE OF CONTENTS .................................................................................................................................. iv

LIST OF FIGURES ........................................................................................................................................ vi

CHAPTER 1: INTRODUCTION ........................................................................................................................ 1
  1.1 Background ................................................................................................................................... 1
  1.2 Problem Statement ........................................................................................................................ 2

CHAPTER 2: LITERATURE OVERVIEW .............................................................................................................. 4
  2.1 Information Filtering Agent ........................................................................................................... 4
  2.2 Search Together ............................................................................................................................ 4
  2.3 WebTagger ..................................................................................................................................... 5
  2.4 Deepa Mehta Desktop ....................................................................................................................... 5
  2.5 Social Bookmarking ........................................................................................................................ 6

CHAPTER 3: ANDROID OPERATING SYSTEM .................................................................................................. 7
  3.1 Architecture ......................................................................................................................................... 7
    3.1.1 The kernel layer ........................................................................................................................... 8
    3.1.2 Native libraries layer .................................................................................................................... 8
    3.1.3 Applications framework layer ...................................................................................................... 10
    3.1.4 Application layer ........................................................................................................................ 10
  3.2 Android Application development .................................................................................................. 11
    3.2.1 Application building blocks ....................................................................................................... 11
      3.2.1.1 Activity Manager .................................................................................................................... 12
      3.2.1.2 Contents Provider .................................................................................................................... 14
      3.2.1.3 View System ........................................................................................................................... 14
      3.2.1.4 Resource Manager ................................................................................................................... 15
      3.2.1.5 Location Services .................................................................................................................... 15
  3.3 Android Software Development ....................................................................................................... 16
    3.3.1 Software Development Kit (SDK) ................................................................................................. 16
  3.4 Security .............................................................................................................................................. 16

CHAPTER 4: USER INTERFACE ....................................................................................................................... 18
  4.1 Introduction ....................................................................................................................................... 18
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2 Architecture</td>
<td>18</td>
</tr>
<tr>
<td>4.3 Main Activity</td>
<td>19</td>
</tr>
<tr>
<td>4.4 Primary Context Activity</td>
<td>20</td>
</tr>
<tr>
<td>4.5 Secondary Context Activity</td>
<td>21</td>
</tr>
<tr>
<td>4.6 Tertiary Context Activity</td>
<td>22</td>
</tr>
<tr>
<td>4.7 Search Activity</td>
<td>23</td>
</tr>
<tr>
<td>4.8 Upload Activity</td>
<td>24</td>
</tr>
<tr>
<td>CHAPTER 5: MIDDLEWARE SERVER</td>
<td>25</td>
</tr>
<tr>
<td>5.1: Overview</td>
<td>25</td>
</tr>
<tr>
<td>5.2 Architecture</td>
<td>26</td>
</tr>
<tr>
<td>5.3 Component Descriptions</td>
<td>27</td>
</tr>
<tr>
<td>5.4 Document Ingestion Process</td>
<td>28</td>
</tr>
<tr>
<td>5.5 Document Search</td>
<td>29</td>
</tr>
<tr>
<td>5.6 Contexts and Documents</td>
<td>30</td>
</tr>
<tr>
<td>CHAPTER 6: FUTUREWORK AND CONCLUSIONS</td>
<td>31</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>33</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1: Android Architecture ....................................................................................................... 7
Figure 2: Application Building Blocks ............................................................................................ 11
Figure 3: Activity Life Cycle ........................................................................................................ 12
Figure 4: States ............................................................................................................................. 14
Figure 5: Resource Manager ......................................................................................................... 15
Figure 6: User Interface Architecture ........................................................................................... 18
Figure 7: Main Activity ................................................................................................................ 19
Figure 8: Primary Context Activity ................................................................................................ 20
Figure 9: Secondary Context Activity .......................................................................................... 21
Figure 10: Tertiary Context Activity ............................................................................................ 22
Figure 11: Search Activity ............................................................................................................ 23
Figure 12: Upload Activity ........................................................................................................... 24
Figure 13: Technologies ............................................................................................................... 25
Figure 14: Middleware Server Architecture ................................................................................. 26
Figure 15: Document Ingestion Process ........................................................................................ 28
Figure 16: Document Search Process ........................................................................................... 29
Figure 17: Context Node Diagram ............................................................................................... 30
CHAPTER 1: INTRODUCTION

1.1 Background

The web has now become the main source for finding the information with the enormous growth of information and knowledge. Due to this explosive growth in data and usage on Web the user gets buried under the information/data which is irrelevant or useless. Hence this forces the user to spend lot of time in finding information that is truly useful among the huge amount of search results. So in order to overcome this problem, there is a need for development of personal network which stores the user's personal data or information. Vijjana is one such kind of personal knowledge network.

Knowledge advantage Machine which is dubbed as Vijjana is a personal assistant for sharing, organizing and acquisition of knowledge [2]. All relevant objects such as documents are presented to user by this knowledge advantage machine based on the place that the user is participating. Any user can create and organize his own personal knowledge network. For example a professor could have many documents organized in different folders like “Course Work”, “Thesis” etc. In this case the professor can build his personal knowledge network with those documents organized in folders and can search for those documents when needed.

The framework Vijjana is a context-centric model for organizing a personal knowledge network. The basic premise of this work is that a user can describe the work-process as a shallow hierarchy of contexts. A semantic relationship exists between all the documents and contexts. The user context could be determined using external means such as personal calendar or location. For example, a professor can have different contexts like “teaching”, “meeting”, “research” etc. When the professor is in a meeting, it could be said that the professor is in the context “meeting”. Then the professor can select the context meeting and extract all the
documents related to that meeting. The term context in this case can be defined as being surrounded by knowledge objects (such as documents) relevant to the place that the user is participating. This approach will save a lot of user’s time when compared to a conventional search. In a conventional search it takes more time for the user to search for the knowledge objects whereas this contextual search makes the user surrounded by knowledge objects.

In order to access the information from a personal knowledge network, a mobile application is developed. This is a user interface mobile application for Android operating system which provides access to various agents of Vijjana architecture in different layers. This application retrieves the contexts from all levels and allows the user to search for documents in the selected contexts. This application also allows the user to upload the files into the server.

1.2 Problem Statement

The main goal of designing Vijjana is to build a knowledge domain which is collaborative and useful to every single user. The users will be able to store the knowledge objects called jans in the server and retrieve them when ever needed. Such mechanism needs a client which is used to access the data from knowledge domain. One way of achieving this process is through Android mobile application.

This application is developed with services for uploading the knowledge objects into the server and also to retrieve them from the server. The knowledge objects in the served are organized manually with in the contexts. Once the user gets a knowledge object that he wants to store in knowledge domain, this user interface application will help in this process. To accomplish this task a secure connection is created to the server using HTTP Post request.
The search service is provided with different levels of contexts. The user can search for knowledge objects by keywords in the appropriate contexts. Once the knowledge objects are retrieved, the visualization service can display the objects in appropriate formats.
CHAPTER 2: LITERATURE OVERVIEW

2.1 Information Filtering Agent

One of the enhanced ways of filtering URLs that are given by the search engines which helps users to select from a set of useful links was first brought up by Kwang Mong [3]. The filtering agent is designed in a very user friendly way that all the out of context URLs given by the search engine are removed according to the phrases or keywords the user inputs. These filtering agents are helpful in determining the relevance of web pages through approaches like relativity of keywords in the web page, checking for the evidence phrases in the document, etc. Even though the filtering agent is successful in eliminating the irrelevant links based on context, this technique lacks one of the most important specifications of modern day search results, the domain centric ontology.

2.2 Search Together

A new solution for collaborative searching was brought up by Meredith et al [4], in which an interface is created for a group of users performing a web search either synchronously or asynchronously. The main advantages of such interface are group searching and persistence. It creates an interactive environment among the users by providing integrated support in evaluating the search results used by several users. The main purpose behind the design of “Search Together” is to share the searched results, division of labor and persistence, but it lacks organization of web gained knowledge. Specific user ratings are given to the links searched through “Search Together” by several users depending on the relevance and eccentricity of the context. This prototype is very useful in the context of collaborative searching and could also be expanded in terms of taxonomy and ontology.
2.3 WebTagger

A new bookmarking service was brought up by Richard. M. Keller et al [5] for sharing and organizing URLs over the web. The WebTagger is an interface designed to give the individuals a shared memory known as repository to organize their URLs in different categories of the system. This helps the users to access their bookmarks anywhere in the world using this system. WebTagger gives the user an advantage to create different categorical folders to save the bookmarks unlike browsers where there is a predefined structure to save the bookmarks. The created folders can be organized either manually or automatically. A pre-defined strategy of ranking and augmented indexing is used to organize the folders automatically. When retrieving the bookmarks from the repository, the user is given an option of browsing all the folders to obtain his desired URL or search depending on indexed categories such as rating mechanisms and relevant feedbacks. This strategy is very helpful to the users who want to store one’s links over the internet in a public shared memory and access anywhere in the world over the internet.

2.4 Deepa Mehta Desktop

Jorg Richter and his team, in the year 2000, developed an open source semantic desktop application [6]. This application uses relational maps termed as topic maps to integrate all the workspace applications into a single knowledge network. This application is useful for the user to work with a collection of applications rather than working on each and every application individually. All the applications are connected in an interface full of topic maps and each application is given an ID to be represented in the map. In this, the user has all the luxury to switch between various applications.
2.5 Social Bookmarking

The term Social Bookmarking refers to various activities like share, organize, manage and search bookmarks, links or other resources over the web. Social Bookmarking is one of the terms that is getting popular by the day in today’s technology but only after several attempts have been made to bring a revolution in the World Wide Web. delicious.com [7], dig.com, twine.com [8], diigo.com are some of the websites that are successful in storing user links and have made their mark in social bookmarking over the web. But these websites failed to provide the user specific knowledge of the domain he is searching for. Hence to provide a domain centric semantic web network, Vijjana is designed to organize the web links stored by the user in a very well defined domain specific manner.
CHAPTER 3: ANDROID OPERATING SYSTEM

Modified Linux version is the base of the Android operating system. Linux operating system runs on personal computers. In order to make it run on mobile devices, the drivers and libraries are modified. The components Android Runtime and Linux kernel are the key components in the operation of Android operating system. Linux operating system provides some important tools like Dalvik Virtual machine. Dalvik Virtual machine is its own java virtual machine. The tools like emulator, debugger and compiler are also provided by this Linux operating system.

3.1 Architecture

The architecture of Android consists of mainly four layers. They are kernel, libraries, applications framework and applications. As discussed earlier, kernel is Linux. Graphics, media capabilities, data storage are provided by the libraries that come with Android. The Android runtime contains the Dalvik virtual machine which is embedded in the libraries layer. All the applications will use the application framework which is an API to access the lowest level of architecture [9].

Figure 1: Android Architecture
### 3.1.1 The kernel layer

The kernel is Linux. As personal computers use the Linux operating system which has a proven record, it was chosen as a kernel for Android operating system. The rewriting of drivers is not required in Linux operating system. The things like virtual memory, power management, drivers and networking are provided by this Linux operating system. Android source code is shipped along with the kernel without any changes in the functions of kernel.

The kernel has many features like:

- It is the patch of kernel enhancements to support android operating system.
- There is no glibc support
- It does not include the full set of standard linux utilities
- It is not Linux but it is built based on the Linux kernel
- There is no native windowing system
- It is standard linux 2.6.24 kernel

### 3.1.2 Native libraries layer

The capabilities for ore features are provided by the native libraries layer. SGL is shipped with the Android. SGL is the primary 2D graphics renderer. To provide the 3D graphics in the Android operating system, OpenGL ES is the tool used. The data storage in the operating system is taken care by SQLite which is in the package of Android. The webpages are also rendered for small screen sizes by the WebKit web rendering engine. This engine is also shipped along with the Android package. The Dalvik virtual machine is also the part of this layer. The bytecode
interpreter is the Dalvik virtual machine. For executing this interpreter on the mobile device, it is highly optimized. Some of the features of Dalvik virtual machine are:

- It is highly CPU-optimized byte code interpreter
- Java .class/.jar files are converted to .dex at build time
- It provides application portability and runtime consistency
- It supports multiple VM processes for device
- It runs optimized file format (.dex) and Dalvik bytecode
- It is designed for embedded environment
- It uses runtime memory very efficiently

To run on the small processors, the java binaries are converted into bytecodes. Java is the main platform where the core libraries are written as it provides many core classes. Some of the core libraries in this layer are

- Surface manager: It is typically about getting access to the display system for 2D/3D layers supported by multiple applications.
- SQLite: It is used by all applications to store the data.
- System C library: The standard C system library (libc) for embedded Linux-based devices
- Others: For supporting audio and video formats in the mobile devices, Media libraries are used. LibWebCore to support the browser with an embedded web view; SGL and 3D libraries for graphics and FreeType for font rendering.
3.1.3 Applications framework layer

Java is the main platform where this layer is written. The applications in the Android operating system use the APIs. All these major APIs are provided by this Application framework layer. The things like accessing the phone data and data sharing are provided by this application framework layer. The authors that build the applications are different. Although the authors are different, same framework is used by all applications which is an import point about Android operating system. This is the main difference when Android operating system is compared with other mobile operating systems.

The services and systems provided by several components in this layer are:

- **Content providers**: This is used by applications for accessing the data or sharing the data between applications.
- **Activity Manager**: The lifecycle of the applications is managed by this layer. It provides navigation back stack to navigate from one activity to other.
- **Resource Manager**: For storage of strings, bitmaps, layout file descriptions and other external parts of the application.
- **View System**: This generates a set of buttons and lists used in UI.
- **Notification manager**: For customizing functions and alerts that are displayed.

3.1.4 Application layer

The software of Android is written in Java. It is interpreted by the Dalvik virtual machine. This layer has the core features like contacts and phone application. The software written by Android team is contained in this layer. The third-party software which the user installs in the device is
also in this layer. There is an advantage of allowing developers to access this layer. The user interface can be checked easily by allowing the developers to access this layer. The events developed by the Android team can also be handled by the third party applications. So any developer can write their own application. Due to this the Android is more robust than the other operating systems. The handling of events by applications can also be specified by the user.

3.2 Android Application development

Android SDK is the platform used by developers for creating the applications. Java programming is used for writing the applications. After writing the applications, they are run on Dalvik virtual machine which is designed for embedded use and run on top of a Linux kernel [10].

3.2.1 Application building blocks

![Application Building Blocks](image)

Android application is bundle of components, where each component is loosely coupled with other. These components run in the same system process. We can also create multiple threads...
within that process or it can be run as separate child thread as the application demands. Most Android mostly makes the processes transparent to the code in the Android apps.

### 3.2.1.1 Activity Manager

Activity is a single focused thing. Activities can give direct interaction to the user. They can run in the foreground. For example they can run current window/tab as background services or they can be embedded in other services.

![Activity Life Cycle](image_url)

*Figure 3: Activity Life Cycle*
Activity in android in an application can be in any of the four states as shown in the figure above.

- **Active/Running state**: This state is gained when the activity is in front and has focus in it. It is visible and active to the user.

- **Paused State**: In this state, the activity is partially visible to the user but not active and lost focus. This happens when some another Activity is on top of this one which doesn’t cover the entire screen or having some transparency so that the underlying Activity is partially visible. A paused activity is completely alive and maintains its state but it can be killed by system under low memory when memory can be freed by no other ways.

- **Stopped State**: The activity is said to be in this state when the Activity is no longer visible in the screen. This happens when another activity is on top of it and completely obscures its view. In the stopped state, the activity is alive and preserves its state, but it is more likely to be killed by the system to free resources whenever necessary.

- **Destroyed/Dead State**: An Activity is said to be dead or destroyed state when it no longer exists in the memory. This happens when either the Activity hasn’t been started yet or once it was started and killed by the system in Paused or Stopped state to free resources.

onStop(), onStart(), onResume(), onPause(), onCreate(), onRestart(), onDestroy() methods define the transitions in the state of the activity when the certain action is performed on it.
3.2.1.2 Contents Provider

The data across applications is handled globally by the Content provider. Android operating system should handle contacts or multi-media data. So it comes with built in content providers which handles contacts or multimedia data. Developers which develop the applications for Android operating system can build their own content providers or they can embed their code with the existing content providers. For example, browser can access the data online through the browser interface.

3.2.1.3 View System

All the classes that handle graphical user interface are joined by this View system. All the elements in the view system are arranged in single tree manner. These elements could be included in XML layout files or can be called from a java code. Android operating system uses the xml files extensively which is a noticeable point about Android development. The abstraction between the layout elements and backend java code is provided by XML files. In the same way
of HTML based web designing, the UI related elements are also designed in Android operating system.

### 3.2.1.4 Resource Manager

All non-code things like icons, text or graphics are handled by this Resource manager. A directory named “res” holds these resources. All the design work and icons will reside under the layout directories as shown in figure.

![Figure 5: Resource Manager](image.png)

### 3.2.1.5 Location Services

The location services like GPS and coarse-grained location providers like cell phone triangulation are supported by the Location services. The central component of location framework is the Location Manager system. The API for accessing the location information of
device is provided by this system. There are many important classes from android.location besides location manager class. Some applications like Location Provider, Gps Satellite and Geocoder use the methods from android.location class.

3.3 Android Software Development

The process of developing new applications is known as the software development on Android. The applications in Android are developed using the java API which uses Android Software Development Kit [11].

3.3.1 Software Development Kit (SDK)

Android SDK has a set of development tools which includes development and debugging tools, a set of libraries, a device emulator, documentation, sample projects, tutorials… Especially the developers use IDE (Integrated development environment) in Eclipse with the ADT Plugin (Android Development Tools) to create, build and debug Android applications. It needs Java Development kit and Apache Ant software to make use of command line tools. And the programmers can use command line tools to create, build and debug Android applications. The applications developed are stored in .apk format and this package contains .dex. These .dex files are compiled byte codes (dalvik executables).

3.4 Security

When compared with other mobile operating systems, Android mobile platform in long run is going to be more secure. One of the reasons is McAfee antivirus, a member of Linux foundation joins OHA (Open Handset Alliance) [12]. Such a Linux secure coding should be into Android development process without any issues. Despite, the disadvantages of open platform are:
• The open source code might be vulnerable to the hackers. There might be attackers who harm the Operating system as it is an open source. Despite the varied functionalities for developing the apps are provided, they still can be exploited and harmed by the attacker. There might be Trojans that are hidden in images. The malware may be passed from one peer to peer. They are more prone to spy and identity thefts. These hazardous malware or viruses or Trojans can be active for many days like in the computers.

• For the above listed attacks, there is the solution provided to the Android mobile platform by the SMobile systems. This security shield integrated application provides shield to the attacks such as: virus, spams… and acts as firewall. This package is ready to run on the android mobile platform.

• The current serious threat is that viruses can be available as the mobile apps and they might function as phone book, dial phone numbers, send text or MMS or connect to internet. For example: attackers can use the GPS app to track a person without his knowledge. This problem can be avoided by the above listed security shield. This can notify and block these security alerts. But still the mobile is not completely protected since it will be connected to internet and at any time, new attack might be faced by the mobile OS.
CHAPTER 4: USER INTERFACE

4.1 Introduction

The User interface Android application is user friendly. It forms an interface between the user and server. The interface gives an easy method of access over the knowledge objects and contexts in the server. It contains different buttons for contexts which are directly linked to server. It is mainly designed for two services, search and upload. The upload service uploads the knowledge objects into Vijjana server. The search service will search and retrieve the knowledge objects from Vijjana server. There is also one more service named context service. It retrieves the contexts from the server and displays in the form of buttons where user can select the context for searching the knowledge objects.

4.2 Architecture

Any user interface is designed intentionally to perform minimal tasks at the client side, whereas the major task of performance and the design goals of the interface are made to be executed on the server side. Similarly this interface is also designed to retrieve the knowledge objects from the server and also to store them in the server.

![User Interface Architecture](image)

Figure 6: User Interface Architecture
As in the architecture all the services are directly linked to the server. When a service is started, a request is sent to the server and the appropriate data is returned from the server.

### 4.3 Main Activity

The main activity is the activity that launches during start up. It comes up with a welcome surface. The Main Activity interface is shown in the figure below. It launches for 5 seconds and then starts the next activity.

![Main Activity](image)

*Figure 7: Main Activity*
4.4 Primary Context Activity

The Primary Context activity starts when the user selects the option to search for files. This surface displays the set of primary contexts that are in the server. When the activity is started, a request is sent to server for level zero contexts. The server gets back with the set of contexts which are displayed in the form of buttons on the surface. When the user selects the contexts the secondary context activity will start. The surface of the Primary Context activity is shown in the figure below.

![Primary Context Activity](image)

Figure 8: Primary Context Activity
4.5 Secondary Context Activity

The Secondary Context activity is started when the user chooses a primary context. This activity also works in the same way as the primary context. It requests the server for level one contexts when the activity is started. The server gets back with a set of secondary contexts which are displayed as buttons on the server. When the user chooses a secondary context, tertiary context will start with a set of tertiary contexts. The surface of the Secondary Context activity is shown in the figure below.

![Secondary Context Activity](image)

Figure 9: Secondary Context Activity
4.6 Tertiary Context Activity

The Tertiary Context Activity is started when the user selects a secondary context. On the startup of this activity, it requests the server for level two contexts. The server will return the set of tertiary contexts which are displayed on the surface as buttons. When the user selects a tertiary context, the search activity will start where the user can enter the keywords for searching the files. The user interface of the tertiary context is shown in the figure below.

Figure 10: Tertiary Context Activity
4.7 Search Activity

The search activity will start when the user selects a tertiary context. The surface of the search activity contains a search bar which allows the user to enter the text. When the user enters a keyword and press the button for searching the files based on that keyword, a request is sent to server with that keyword. The files which are related to that keyword are returned by the server and are displayed on the surface of search activity in raw form. The surface of the search activity is shown in the figure below.

Figure 11: Search Activity
4.8 Upload Activity

This activity is started when the user selects the option to upload after the main activity. The surface of this activity contains two bars one for uploading text files and other for uploading music files. It contains buttons for browsing which allows the user to select the files that are needed to be uploaded into the server. When the user selects the file and presses the upload button, a request is sent to the server to store the file. The server will store the file in appropriate context and returns it when the user searches for it. The user interface for the upload activity is shown in the figure below.

![Figure 12: Upload Activity](image-url)
CHAPTER 5: MIDDLEWARE SERVER

5.1: Overview

The middleware server is responsible for handling all requests from the client which is the android application. It is a Java Spring MVC server implementation. It is composed with different technologies and libraries for storing the knowledge objects like contexts and documents. Spring framework technology (Spring 3.1.1RELEASE) is used in the server. Spring-WebMVC is responsible for Model, View and control paradigm for Spring. The MongoDB API for Spring Spring-Data-MongoDB is used for storing the documents in the server. For indexing the documents Apache Solr 3.6.1 technology is used. It also acts as search engine which can be used for searching the documents in the server. Apache SolrJ 3.0.1 is the Java API for interacting with Apache Solr. The Web server container is JBoss Community server 7.0.1. A database is also needed to store the documents in the server which is MongoDB 2.2.0. It is a noSQL database which stores the knowledge objects such as files in it. These are the main technologies that are used in the Vijjana Server.

Figure 13: Technologies
5.2 Architecture

An inverted component stack diagram of Vijjana server is shown below. It depicts the system architecture of Vijjana Middleware server.

![Middleware Server Architecture](image)

Apache Solr is responsible for indexing the documents. When a user uploads a document into server, Apache Solr indexes the document with an address. It doesn’t store the document but generates a unique index for the document. After generation of the unique index the document is stored in the MongoDB database. When the user searches for a document the, the unique id generated by Apache Solr for that document is searched in the MongoDB and returns the document if it finds any. The Context Service is responsible to store the contexts in all levels and retrieve them when a request from the user is sent. The document service is responsible for different types of functions that can be performed on documents. It is responsible for ingesting,
storing and retrieving the documents on the request of user. Spring Web MVC is the technology which allows the user to do the operations on server through REST client. The sample contexts are generated through the REST interface which can be used to test from the user interface android application. JBoss is the web server which holds all the functions that are performed at server.

5.3 Component Descriptions

The middleware server provides different services like document and context. The document service is responsible for indexing, storing, searching and retrieving the document data. When the user ingests a document into the server, the document service indexes the document and stores it in database. Similarly when user searches for a document with a keyword, the document is searched based on the index and is retrieved from the database by the document service. The context service is responsible for saving, storing and retrieving the contexts upon the request of the user. When the user requests for contexts at a level, the context service is responsible for retrieving the contexts at that level. Similarly when the user creates a new context, it is saved and stored in the database which is also a function of context service. Semantic engine is responsible for processing the documents that is to store the documents in appropriate contexts. When a file is ingested into server, the semantic engine checks the context repository and stores the document in a context if it finds any context terms matching to that file. The Solr and Context are the two repositories that are present in the middleware server. The Solr repository contains all the indexes of the documents that are stored in the server. The serves searches for a file in the Solr repository based on the index. The Context repository contains all the levels of contexts that are stored in the server. When a new document is ingested, the context repository is checked by semantic engine to find any context terms that matches the document.
5.4 Document Ingestion Process

The block diagram of the document ingestion process is shown in the block diagram below.

![Diagram of Document Ingestion Process]

When a new document is ingested into middleware server, it goes through Document service. The binary data of the new document is stored on the server. The server creates a new graph node in the Graph repository. The Solr indexes the document with a unique id and stores the index in the Solr repository. The Semantic engine will check the Context repository to find any context terms matching for that document to store in appropriate context. If it finds any context matching, then these contexts will have neighbors pointing to a newly created graph node. The
graph nodes neighbors will include a reverse reference to all matched context nodes to achieve the bi-directionality. The lists of contexts are indexed with the document as a part of “category” field when it is sent to Solr.

5.5 Document Search

The flow of search in the middleware server is shown in the figure below.

![Diagram of Document Search Process]

When a search request is received from the client, the document service takes the responsibility of the function. The document service will search for the document in the Solr repository and if it finds any, it returns with the index of the file from Solr repository. The index is checked in the database and the file from the data base is retrieved and passes it to client. The semantic engine has no work to do during the search process.
5.6 Contexts and Documents

The contexts are created manually by the user through REST client and are stored as a type of node. The documents are not linked to each other. The documents are linked to context nodes. When a new document is ingested, it is linked to the context node if it finds and context term matching to it. When a document is deleted, all the references to the context and from the context are removed. If there are no references from a super node to a document node after deleting the existing references, the super node is deleted. All nodes are bi-directional which means Document A has a reference to a neighbor of Context B and vice versa when Document A and Context B are linked together. A sample context node diagram is shown in the figure below.

Figure 17: Context Node Diagram

Documents B and D are related by the contexts “CS556” and “homework”
Documents C and E are related only by the context “homework” to Documents B and D
Document A is related only by the context “CS556” to Documents B and D
CHAPTER 6: FUTUREWORK AND CONCLUSIONS

We have proposed a designed user interface android application which acts as a client for the middleware server. The user needs to install the application in the mobile to get access over the middleware server which contains the user personalized knowledge domain. The upload service in the client connects to server and ingests the documents into the middleware server. One of the other services designed here include the searching service which is incorporated in the application. This server takes the user keywords as input and retrieves the results from middleware server. The user interface designed here succeeded in uploading the files into server and retrieving the files from the server. It also succeeded in extracting the contexts in all levels from the server which is a context centric knowledge sharing network. The middleware server also succeeded in providing services for storing and retrieving the documents and contexts. The documents in the server are stored in appropriate contexts which reduces the complexity of users selecting the documents from the results.

The search service implemented here takes a single keyword at a time for searching the database. This service can be extended to take multiple keywords or a sentence as input from the user and search the database for a more clear view of the knowledge domain network. This needs to incorporate high scalable searching algorithms. Optimal algorithms to increase the speed of the search strategy can be designed to increase the performance of the search service implemented here.

In this server, the contexts are created manually by the user. This can be extended such that the contexts could be extracted automatically from the documents ingested using statistical algorithms. The server can be automated such that when a new file is ingested, it should
automatically search for contexts and place the file in appropriate contexts. If the server doesn’t find any context, it should create one based on the keywords in the file and place the new file in newly created context.

The user interface can be extended which adds more graphics to different surfaces. The documents retrieved in this user interface are visualized in raw format. This can be extended such that the documents retrieved from the server could be visualized in different graphical formats along with the relations between the files.
REFERENCES


