Mobile Application for Transliteration of Roman Script to Gurmukhi Script

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Computer Science Engineering

Submitted By:
MANINDER SINGH
(801032013)

Under the supervision of:

Mr. PARTEEK KUMAR
Assistant Professor

Ms. RUPINDERDEEP KAUR
Lecturer

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2012

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Mobile Application for Transliteration of *Roman* Script to *Gurmukhi* Script"**, in partial fulfilment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Parteek Kumar and Ms. Rupinderdeep Kaur* refers other researcher's works which are duly listed in the reference section. The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Maninder Singh)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Mr. Parteek Kumar)
Assistant Professor
Computer Science & Engg. Deptt.
Thapar University,
Patiala.

(Ms. Rupinderdeep Kaur)
Lecturer
Computer Science & Engg. Deptt.
Thapar University,
Patiala.

Countersigned by:

(Dr. MANINDER SINGH)
Assoc. Professor & Head
Computer Science & Engg. Deptt.,
Thapar University,
Patiala.

(Dr. S. K. MOHAPATRA)
Dean (Academic Affairs)
Thapar University,
Patiala.

Acknowledgement

No volume of words is enough to express my gratitude towards my guide, ***Mr. Parteek Kumar,*** *Assistant professor and* ***Ms. Rupinderdeep Kaur,*** *Lecturer, Computer Science and Engineering Department, Thapar University, who has been very concerned and has aided for all the material essential for the preparation of this thesis report. He has helped me to explore this vast topic in an organized manner and provided me with all the ideas on how to work towards a research-oriented venture.*

I am also thankful to ***Dr. Maninder Singh****, Head of Department, CSED and* ***Mr. Karun Verma****, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.*

I would also like to thank the staff members and my colleagues who were always there in the need of the hour and provided with all the help and facilities, which I required, for the completion of my thesis.

Most importantly, I would like to thank my parents, my wife and the Almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Maninder Singh
801032013

Abstract

Language transliteration is one of the important areas in Natural Language Processing. Accurate transliteration of named entities plays an important role in the performance of machine translation and cross-language information retrieval processes. As Punjab has biggest NRI population and they frequently visit to their motherland. The new generation of these NRI people does not have understanding to the Punjabi Language. They know the words but grammatically they don't know how to write Punjabi. With the help of this mobile application, NRI Punjabi can send a SMS to the transliteration service and receive the SMS as output in the desired Language (Punjabi). There are many challenges in transliteration of *Roman* script to *Gurmukhi* script, because there is large character gap in both the scripts. As in *Roman* script, there are 21 consonant and 5 vowels only, but in Gurmukhi script there are 41 consonants and 19 vowels. So it is a bit difficult to map these characters. There are different approaches for transliteration like rule based approach, statistical approach, *etc*. Rule based approach is used for the transliteration purpose, in this system and is discussed in this thesis. J2ME Environment is used to develop this mobile application and is connected to a Web Application, which in turn transliterate the user input. For Transliteration Purpose, more than 24 rules are being developed, which are used by web application while transliterating. Experiments conducted on application have shown that the performance is sufficiently good. Most commonly used words are taken from daily life and processed by this application. Output generated by processing these words is shown in chapter 5. Accuracy of this system comes out to be 80% for these daily use sentences. The wrong word creation was mainly because of ੱ (Adhak), ਂ (Bindi) and ੰ (Tippi). The accuracy of system can be further improved with the help of dictionary.

Table of Contents

List of Algorithms

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
CLDC	Connected Limited Device Configuration
CSM	Character Sequence Modeling
EPU	English Pronunciation Unit
J2ME	Java 2 Micro Edition
JVM	Java Virtual Machine
KVM	Kilobyte Virtual Machine
MIDP	Mobile Information Device Profile
PMT	Punjabi Machine Transliteration
SMS	Short Message Service
SMT	Statistical Machine Translation
VMD	Visual Mobile Designer
WEKA	Waikato Environment for Knowledge Analysis
WSA	Web Service Architecture

Chapter 1

Introduction

The communication among human beings is done by Language. This world is collection of diverse cultures and societies. We use different types of Languages to communicate among each other. The world has become a Village with new technologies like Computer, Internet and Telecommunication. There is a need to bridge the Language barriers that exist all around the world. We can use the technology for crossing these Language barriers. The mobiles can be used to translate or transliterate the source Language to desired target Language with only pressing of few keys.

The Language transliteration is very subjective. The software cannot be directly used on the limited resource system like mobile phones. But these services can be utilized with the help of Web Services, Short Message Service (SMS) and MIDlets [18] on mobile systems. SMS is a text messaging service component of phone, web, or mobile communication systems, using standardized communications protocols that allow the exchange of short text messages between fixed line or mobile phone devices. A MIDlet is an application that uses the Mobile Information Device Profile (MIDP) of the Connected Limited Device Configuration (CLDC) for the Java ME environment. Typical applications include games running on mobile devices and cell phones which have small graphical displays, simple numeric keypad interfaces and limited network access over Hyper Text Transfer Protocol (HTTP). Java is the platform on which the MIDlets are developed for specific domains. The java application uses Mobile Information Device Profile (MIDP) [13] to develop an application for mobile phone. In this thesis a simple MIDlet has been created using Java 2 Micro Edition (J2ME), *i.e.*, an application on MIDP for mobile phones. This application converts the English word written in *Roman* text into the Punjabi word written in *Gurmukhi* text using rule based transliteration techniques.

To develop a mobile based application, J2ME programming environment can be used. This application uses the MIDP profile by J2ME for Mobile phones and sun wireless toolkit of Netbean Integrated Development Environment. The user will enter the English text as input in emulator screen and get the transliterated Punjabi text as

output. Further, the web application is also developed that can be used to connect the mobile with the web service, which actually transliterate the input string. Further, it can be used to connect the existing website that implements machine translation or transliteration.

1.1 Transliteration

Transliteration is as important as translation in area of Natural Language Processing. Transliteration system converts the input source string into target Language word, which is based on phonetic of source word. Transliteration is of two types, forward and backward. When a word is transliterated from its origin Language to a foreign Language, then it is called forward transliteration, while transliterating a word from foreign Language to its origin Language, is called backward transliteration. In this thesis, backward transliteration is being discussed, which is English to Punjabi transliteration system. In this system, user gives input string on its mobile application like '*oh ghar ja ria hai*' and it will give output in Punjabi as ਓਹ ਘਰ ਜਾ ਰਿਆ ਹੈ *uh ghar jā riā hai* '*he is going home*'. The most spoken and understandable Language of the north India and NRI Diaspora is *Gurmukhi* Language while they do not know exactly how to write the words or sentence in *Gurmukhi* Language. Till now, the best option for such users was to use the Punjabi phonetic keyboard in which, most of the Roman keys were mapped phonetically to nearest sounding *Gurmukhi* characters. But these keyboards had their shortcomings, as not all the keys could be mapped. The Language is also not standardized for the informal use. It varies from every village to village. This adds to problems for Language transliteration software development. Another problem is that all the sounds of Punjabi are not properly represented in English. A user can use same English character to represent more than one *Gurmukhi* character, *e.g*., '*t*' could represent both ਟ *ta* and ਤ *ṭa* in the *Gurmukhi* script. The challenge for the software is to automatically map the *Roman* character to an appropriate *Gurmukhi* character. To resolve the problem the transliteration is one of the viable solutions.

1.2 Approaches for Transliteration

There are three basic approaches to transliterate a word from source Language to target Language. First is Grapheme based approach, second is Phoneme based

approach and third is hybrid approach. These different approaches are explained below.

1.2.1 Grapheme Based Modeling

In this type of model, English string is not converted into its phonemic representation before its alignment with the transliterated string. Rather, it is broken down into arbitrary substrings, each of which is output independently (and possibly incorrectly). This method fails to generate correct transliteration where English spelling is not reflected in pronunciation.

1.2.2 Phoneme Based Modeling

The systems based on phoneme alignment are more numerous. Results generated by phoneme based systems are more accurate than that of grapheme based models. That is why, mostly systems are based on phoneme based modelling.

1.2.3 Hybrid Modeling

In this type of system, both phoneme and grapheme modelling are used to transliterate a string. That is why, this model is most popular.

Transliteration can be further divided on the basis of approach to transliterate, which may be one of the following:

1.2.4 Rule Based Transliteration

It includes Transfer-based machine transliteration, Interlingual machine transliteration, and Dictionary-based Machine Translation. The Transfer based and Interlingual method uses the intermediate representations. This intermediate representation captures the meaning of the original text in order to generate correct transliteration. This Dictionary based translation, as the name suggests, uses the dictionary conversion of the sentence.

1.2.5 Statistical Transliteration

Some statistical method is used on bilingual corpus to generate the required results. For example, MOSES, CSM, WEKA *etc*.

1.3 Comparative Analysis of English and Punjabi Language

English is a West Germanic Language that arose in the Anglo-Saxon kingdoms of England [15]. It is one of six official Languages of the United Nations. India is one of the countries where English is spoken as a second Language. There are 21 consonant letters in English. These are B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, and Z. The rest of the letters of the alphabet are called vowels. The vowels are: A, E, I, O, U. As defined in the Constitution of India, English is one of the two official Languages of communication (Hindi being the other) for India's federal government and is one of the 22 scheduled Languages specified in the Eighth Schedule to the Constitution. A working knowledge of English has become a requirement in a number of fields, occupations and professions such as medicine and computing; as a consequence over a billion people speak English to at least a basic level.

Punjabi (ਪੰਜਾਬੀ in *Gurmukhi* script) is an Indo-Aryan Language spoken by inhabitants of the historical Punjab region (in Pakistan and India). Punjabi is the 10th most spoken Language in India and 3rd most spoken Language in South Asia [15]. The number of consonants is 41 in *Gurmukhi* script as given in Table 1.1. There are 19 vowels in total (10 Independent vowels and 9 dependent vowels) in *Gurmukhi* Script as shown in Table 1.2. Two symbols are used for nasalization ◌ਂ (Bindi) and ◌ੰ (Tippi) and one symbol ◌ੱ (Adhak) that doubles the consonant before which it appears. In India, Punjabi is one of the 22 Languages with official status in India. It is the first official Language of Punjab (India) and Union Territory State Chandigarh and the 2nd official Language of Haryana, Himachal Pradesh and Delhi. It is used in government, education, commerce, art, and mass media and in every day communication [15].

Table 1.1: Consonants in Gurmukhi Script

ੳ	ਅ	ੲ	ਸ	ਹ	ਕ	ਖ
ਗ	ਘ	ਙ	ਚ	ਛ	ਜ	ਝ
ਞ	ਟ	ਠ	ਡ	ਢ	ਣ	ਤ
ਥ	ਦ	ਧ	ਨ	ਪ	ਫ	ਬ

ਭ	ਮ	ਯ	ਰ	ਲ	ਵ	ੜ
ਸ਼	ਖ਼	ਗ਼	ਜ਼	ਫ਼	ਲ਼	

Table 1.2: Vowels in Punjabi Language

Independent vowels	Dependent Vowels
ਅ	◌ੱ
ਆ	◌ਾ
ਇ	◌ਿ
ਈ	◌ੀ
ਉ	◌ੁ
ਊ	◌ੂ
ਏ	◌ੇ
ਐ	◌ੈ
ਓ	◌ੋ
ਔ	◌ੌ
ਅੰ	

1.4 Challenges in English to Punjabi Transliteration

As there is huge character gap in English and Punjabi, so it is quite difficult to transliterate English to Punjabi accurately. There are many challenges in English to Punjabi transliteration, some of which are given below [9].

1.4.1 Character Gap

This problem arises because for 21 consonants in English there are 41 consonants in Punjabi and for 5 vowels in English there are 19 vowels in Punjabi. Due to this gap transliteration become difficult, *e.g.*, for Punjabi character ਞ *ñ* there is no corresponding character in English keyboard.

1.4.2 One-to-Multi Mapping Problem

As there are less number of consonant in English, that is why some consonants are used to produce sound of more than one consonant in Punjabi, *e.g.*, '*d*' in English Language can be transliterated into ਡ *da* and ਦ *ḍa*. Due to this problem if we transliterate '*thand*' than two transliterations are possible ਠੰਦ *ṭhand* and ਠੰਡ *ṭhaṇḍ 'cold'*, but only second one has meaning, so first should be discarded by system. Multi mapping problem associated with characters is shown in table 1.3.

Table 1.3: One to Multiple Mapping

English character	Punjabi character
'c'	'ਚ' and 'ਕ'
'r'	'ਰ' and 'ੜ'
'n'	'ਨ' and 'ਣ'
'd'	'ਡ' and 'ਦ'
't'	'ਟ' and 'ਤ'
'th'	'ਠ' and 'ਥ'

1.4.3 Schwa Deletion

Schwa is defined as the mid-central vowel that occurs in unstressed syllables. The problem with schwa is that sometimes it is retained and sometime it is deleted without any exception. Schwa deletion is an important issue in grapheme-to-phoneme conversion for Indo-Aryan Languages. For example, word '*kamra*' is transliterated into ਕਮਰਾ *kamrā* '*room*' whereby '*a*' in between characters '*k*' and '*m*' is deleted and

‘*a*’ after character ‘*r*’ is retained [11]. To find out when to retain schwa sound is a trivial task.

1.4.4 Multiple Representation of Same Word

In English, there are certain words which can be represented in multiple ways in Punjabi Language. Which one of them is accurate depends upon user’s perception, *e.g.*, word ‘*Infosys*’ can be transliterated to ਇਨਫੋਸਿਸ *inphōsis* and ਈਨਫੋਸਿਸ *īnphōsis*.

1.4.5 Multi-to-One Map Problem

In some cases to pronounce a character in Punjabi, combination of English characters is required because English characters are less in number than Punjabi. For example, ‘*ch*’ is used to represent ਚ *c*. This problem makes transliteration a bit difficult. There are some multi-to-one map characters shown in Table 1.4.

Table 1.4: Multi-to-One Map Characters

Multiple character in English	Mapped Punjabi character
‘ch’	‘ਚ’
‘dh’	‘ਢ’ and ‘ਧ’
‘th’	‘ਠ’ and ‘ਥ’
‘kh’	‘ਖ’
‘rh’	‘ੜ’
‘ai’	‘ੈ’
‘sh’	‘ਸ਼’
‘chh’	‘ਛ’
‘gh’	‘ਘ’

1.4.6 Double Occurrence

There are certain alphabets whose double occurrence represents a different character in Punjabi, *e.g.*, ‘*ee*’ represent ੀ character in Punjabi, but if there is single occurrence

of ‘*e*’ then it represents ੇ character in Punjabi. Thus, these types of characters create a problem in transliteration, *e.g.*, ‘*gurmeet*’ is transliterated to ਗੁਰਮੀਤ *gurmīt* and ‘*saver*’ to ਸਵੇਰ *savēr* ‘*morning*’. Some double occurrence words are ‘*ee*’, ‘*oo*’ and ‘*aa*’.

1.4.7 Wrong Word Creation Due to Rules

Some rules which are required for some words also give wrong solution for other words. For example, there is rule that if ‘*n*’ is followed by ‘*a*’ then replace ‘*an*’ for ੰ (tippi) in Punjabi. This rule gives right solution for word ‘*kandh*’ as ਕੰਧ *kandh* ‘*wall*’, ‘*rang*’ as ਰੰਗ *raṅg* ‘colour’ which is true, but at the same time if we write ‘*mansa*’ it will give ਮੰਸਾ *maṃsā* instead of ਮਾਨਸਾ *mānsā*. Thus rule is leading us to a wrong answer, so dictionary is required for such cases.

Hence transliteration system converts source script to target script based on phonetics of target Language. There are different approaches to perform transliteration, *e.g.*, rule based approach, statistical approach, *etc*. There are some problems also in *Roman* script to *Gurmukhi* script transliteration, because there is huge character gap between Roman script and *Gurmukhi* Script. These problems are schwa deletion, one-to-multi map, wrong word creation due to rules, *etc*.

Chapter 2

Review of Literature

Transliteration is a subset of the science of hermeneutics. It is a form of translation, and is the practice of converting a text from one script to another. There are some words in Language which cannot be translated, *e.g.*, names of people, places, *etc*. So, to convert these names to desired Language script, term transliteration came into existence. There are many systems, which are developed till now for the purpose of transliteration.

2.1 Indian Language Transliteration Systems

Malik (2006) has developed a rule based Punjabi Machine Transliteration (PMT) [12]. PMT preserved the phonetic of transliterated word and meaning of transliterated word. The limitation of this system is that, the system works only at input data which has been edited for missing Vowels or Diacritical Marks which particularly has limited use. It converts a *Shahmukhi* (Punjabi variant of *Perso-Arabic* Script) word into a *Gurmukhi* word irrespective of the type constraints of the word. It not only preserves the phonetics of the transliterated word but in contrast to usual transliteration, also preserves the meaning. He has discussed and compared two scripts. Based on this comparison and analysis, character mappings between *Shahmukhi* and *Gurmukhi* are drawn and transliteration rules are discussed. Finally, architecture and process of the PMT system is discussed. When it is applied to Punjabi Unicode encoded text, especially designed for testing, the results were compiled and analyzed. PMT system will provide basis for Cross-Scriptural Information Retrieval (CSIR) and Cross-Scriptural Application Development (CSAD). PMT is a special type of Machine Transliteration in which a word is transliterated across two different writing systems used for the same Language. It is independent of the type constraint of the word. This system has reported to be more than 98% accurate for classical literature and more than 99% accurate for modern literature.

Vijaya *et. al*. (2009) have developed a '*English to Tamil Transliteration using WEKA system*' [20]. WEKA (Waikato Environment for Knowledge Analysis) is a popular

suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. WEKA is free software available under the GNU General Public License. They used j48 classifier of WEKA for purpose of classification. They have presented a transliteration model where the transliteration problem is modelled using classification technique and implemented using WEKA j48 decision tree classifier. The transliteration model is experimented on English to Tamil transliteration and its performance is evaluated. As corpus is very important for any transliteration system so they generated a bilingual corpus consisting of 30,000 Indian person names and 30,000 Indian place names. This corpus is used in training the transliteration model. The source Language considered there is English and the target Language is Tamil. This system scales well, even where there are varying numbers of training examples and considerable numbers of attributes in large databases. This transliteration system was tested with a data set of 1000 English names. The transliteration system based on decision tree classifier produced exact Tamil transliteration for English words with an accuracy of 84.82%. The misclassifications by the model can be attributed to the delicate characteristics of the Tamil Language while transliterating vowels and the consonants '*r*', '*l*', '*n*', '*d*', '*t*'. Several possible transliterations are made based on the second best class label for the vowels and the above mentioned consonants. Considering only the top 2 results, 87.19% of accuracy is obtained.

Ali and Ijaz (2009) have developed mapping rules based '*English to Urdu transliteration system*' [1]. It consist of three steps, first step is used to generate Urdu text from English transcription with the help of mapping rules. Urdu syllabification is applied on English transcription in second step. Syllabification stands for breaking up a word into syllables. In the third step some Urduization rules are implemented to improve the system's overall accuracy. In order to develop English to Urdu transliteration system, first the rule-based approach employing transliteration from English orthography to Urdu orthography was explored but soon it was realized that it would not work well as there is no one-to-one mapping between English orthography and its corresponding sound. So, pronunciation based transliteration was chosen as it produced better results. An '*Arpabet*' based English pronunciation lexicon is used for acquiring pronunciation of English words. English text is converted to Urdu using English pronunciation and mapping rules. The English pronunciation lexicon is based

on American accent, hence the transliteration into Urdu also depicts American accent. Frequently used English words are transliterated manually and some rules are applied for Urduization of the transliterated text in order to make it appropriate and as close as possible to the local accent, *i.e.*, the accent that is used in Pakistan while speaking English. Out-Of-Vocabulary problem is resolved using statistical techniques by first aligning English orthography to pronunciation sequences. Overall system's accuracy is 96% which is quite promising. The System can be improved by training transliteration model on Urdu accent instead of American accent.

Chinnakotla *et. al*. (2010) have developed a '*Transliteration System for Resource Scarce Languages*' [2]. To perform transliteration, they used Character Sequence Modeling (CSM) to identify word origin. They showed that by using just the monolingual resources in conjunction with manually created rule base, one can achieve reasonable transliteration performance when compared to that of baseline statistical systems trained using parallel corpora. They achieved this performance by properly harnessing the power of Character Sequence Modeling (CSM), typically called the Language Model. Their system uses CSM on the source side for word origin identification, a manually generated non-probabilistic character mapping rule base for generating transliteration candidates, and then again uses the CSM on the target side for ranking the generated candidates. They perform a step-by-step fine-tuning of various CSM parameters. As done earlier, they build a constrained rule-base for mapping English character sequences to Hindi character sequences. Similar from Hindi to English, the constrained rule-base contains both simple and compound rules where a rule is defined as simple or compound depending on whether the source of a rule is a single English letter (*k*) or an English letter sequence (*ck*). In English, vowel sequences (Example: '*oe*', '*eo*', '*ie*', '*ee*', '*ai*') and silent letters (*e.g.*, '*lm*' (holmes), '*ps*' (pseudo)) are handled through compound rules. While transliterating from English to Hindi, due to the Schwa phenomenon, English vowels may not be mapped to any character on the Hindi side. For example, in '*ganesh*', '*but*', '*ton*', the letters '*a*', '*u*', '*o*' do not map to any Hindi character. To handle this, they include NULL mapping on the target side for each English vowel.

Deep and Goyal (2011) have developed a '*Punjabi to English Transliteration System*' [3]. They have made the Punjabi to English Machine transliteration system which is forward transliteration system. This system works in three phases to transliterate a

Punjabi word to English. These phases are tokenization, rules and direct mapping and comparison with dictionary. In tokenization phase system gets Punjabi word as a input and break it into the individual character. On the tokenized word, rules are applied. They have developed the 41 rules for Punjabi to English Transliteration. Rules in their system include constraints which specify the context in which they are applicable like Start of a Word, ending of a Word, After Vowel, and After Consonant *etc*. If none of rule is applicable on the word then direct mapping is applied. Reason to develop rules is that if they used direct character mapping, then accuracy of system would be very low. To improve that accuracy, they have developed different rules. Combination of different mapping options for each character in inputting Punjabi words results in different output words. They have created a dictionary that contains the spellings of names that are commonly used in real life. Thus, output of the system is then compared with the dictionary words to give accurate result.

2.2 Transliteration Systems for Foreign Language

Wan and Verspoor (1998) have developed an '*Automatic English-Chinese name transliteration'* for development of multilingual resources [21]. They have made a multilingual Natural Language Processing system, which aim for coverage of both Language using *Roman* alphabet and Language using other alphabets, the development of lexical resources must include mechanism for handling words which do not have standard translations. Words falling in this category are words which do not have any obvious semantic content, *e.g.*, most Indo-European personal and place names, and which can therefore not simply be mapped to translation equivalents. They use the term transliteration to refer generally to problem of identification of a specific textual form in output Language, which corresponds to specific textual form in an input Language. For words with semantic content, this process is essentially equivalent to translation of individual words. For words with little or no semantic content, such as personal and place names, dictionary lookup may suffice where standard translation exists, but in general it can't be assumed that names will be included in bilingual dictionary. In multilingual system, designed only for Languages sharing the *Roman* alphabet, such names pose no problem as they can simply be included unaltered in output texts in any of the Languages. They cannot, however, be included in Chinese text, as roman characters cannot standardly be realized in Han

character set. English-Chinese name transliteration occurs on the basis of pronunciation. That is, the written English word is mapped to written Chinese character via spoken form associated with the word. The idealized process consists of: mapping an English word to a phonemic representation and mapping each phoneme composing the word to a corresponding Chinese character. In practice this process is not entirely straightforward they outline several issues complicating the automation of this process.

Oh and Choi (2002) have developed '*Hybrid Approach base English to Korean Transliteration System*' [14]. They have used both phonetic information and orthography. This system first performs alignment and then transliteration. Firstly, English pronunciation unit and its phoneme have been aligned phonetically. Secondly, English words are transliterated to Korean through several steps. They have used an English pronunciation dictionary, with the help of this dictionary, they try to find complex word. For this purpose, they divide the word in two (word + word) using pronunciation dictionary. If match for the words is found in pronunciation dictionary then system can assign pronunciation to given word otherwise system will estimate the pronunciation. When match is not found in dictionary than system checks the origin of English word means whether it is a pure English word or a Greek origin word, because the way to pronounce Greek origin English word into Korean is different from original English words. First, an English Pronunciation Unit2 (EPU) and its corresponding phoneme are aligned. EPU-to-Phoneme alignment is to find out the most phonetically probable correspondence between an English pronunciation unit and phoneme. EPU to phoneme aligned results acquired from the alignment algorithm offer training data for estimating pronunciation of English words, which are not registered in a pronunciation dictionary, for example, 'zinkenite'. Second, English words are transliterated into Korean words through several steps. Finally, Korean transliterated words are generated using conversion rules. Using English, This system has reported 90.82% word accuracy and 56% character accuracy.

Zelenko and Aone (2006) have proposed some '*Discriminative methods for Transliteration*' [22]. They present two discriminative methods for name transliteration. The methods correspond to local and global modeling approaches in modeling structured output spaces. Both methods do not require alignment of names in different Languages. Their features are computed directly from the names

themselves. They perform an experimental evaluation of the methods for name transliteration from three Languages (Arabic, Korean, and Russian) into English, and compare the methods experimentally to a state-of-the-art joint probabilistic modeling approach. They find that the discriminative methods outperform probabilistic modeling, with the global discriminative modeling approach achieving the best performance in all Languages. They address the problem of transliteration in the general setting: it involves trying to recover original English names from their transcription in a foreign Language, as well as finding an acceptable spelling of a foreign name in English. They have applied name transliteration in the context of cross-lingual information extraction. Name extractors are currently available in multiple Languages. Their goal was to make the extracted names understandable to monolingual English speakers by transliterating the names into English. The extraction context of the transliteration application imposes additional complexity constraints on the task. In particular, they aim for the transliteration speed to be comparable to that of extraction speed.

Hong *et. al*. (2009) have developed '*A Hybrid Approach to English-Korean Name Transliteration system*' [6]. The base system is built on 'MOSES' with enabled factored translation features. They expand the base system by combining with various transliteration methods including a '*Web-based n-best re-ranking*', a dictionary-based method, and a rule-based method. Their standard run and best non-standard run achieve 45.1 and 78.5, respectively, in top-1 accuracy. Experimental results show that expanding training data size significantly contributes to the performance. Also, they discovered that the Web-based re-ranking method can be successfully applied to the English-Korean transliteration.

In order to build their base system, they use MOSES, a well-known phrase-based system designed for Statistical Machine Translation (SMT). MOSES is a free software statistical machine translation engine that allows automatically training translation models for any Language pair given a collection of source and target text pairs. MOSES offers a convenient framework which can be directly applied to machine transliteration experiments. In this framework, the transliteration can be performed in a very similar process of SMT task except the following changes. First, the unit of translation is changed from words to characters. Second, a phrase in transliteration refers to any contiguous block of character sequence which can be

directly matched from a source word to a target word. Also, they do not have to worry about any distortion parameters because decoding can be performed in a totally monotonic way. The process of the general transliteration approach begins by matching the unit of a source word to the unit of a target word. The unit can be based on graphemes or phonemes, depending on Language pairs or approaches. In English-Korean transliteration, both grapheme-to-grapheme and grapheme-to-phoneme approaches are possible. In their method, they selected grapheme-to-grapheme approach as a base system, and they apply grapheme-to-phoneme functions in pronouncing dictionary-based approach.

2.3 Tools for Mobile Application Development

To make the mobile application, J2ME can be used as a platform for development. The term J2ME stands for Java 2 Micro Edition, *i.e.*, Java for that device, which has limited memory power and other resource. J2ME is one of the few technologies that are currently used by the developers for developing mobile applications.

2.3.1 Java 2 Micro Edition (J2ME)

Sun Microsystems defines J2ME [19] as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems". By adding a virtual machine between applications and operating systems, J2ME programs are inherently platform independent and thus increase productivity for the programs running on mobile devices. The virtual machine of J2ME [10], Kilobyte Virtual Machine (KVM), is much smaller than traditional Java Virtual Machine (JVM). It is designed to run under the constraints of the limited resources available on micro devices. Currently, J2ME has been one of the most widely-adopted developing platforms for mobile devices. The conceptual view of J2ME is shown in figure 2.1. Java 2 Micro-Edition Connected Limited Device Configuration (J2ME CLDC) is the platform of choice when it comes to running mobile applications on resource constrained devices (cell phones, set-top boxes, *etc*.). The Mobile Information Device Profile (MIDP) with the Connected Limited Device Configuration (CLDC) is the Java runtime environment for today‘s mobile information devices.

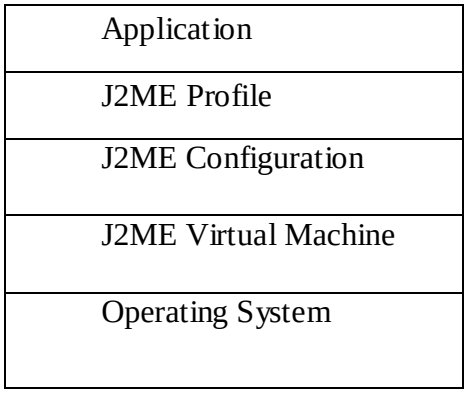

Figure 2.1: Hierarchy of J2ME

2.3.2 Mobile Information Device Profile (MIDP) Layered Architecture

The Mobile Information Device Profile (MIDP) [17] is designed for mobile phones and entry-level Personal Digital Assistant (PDA's). It offers the core application functionality required by mobile applications, including the user interface, network connectivity, local data storage and application management. Combined with Connected Limited Device Configuration (CLDC), MIDP provides a Java runtime environment that is suitable to the capabilities of handheld devices. This profile layer resides on the CLDC layer. The CLDC layer resides on native operating system. The Conceptual Architecture of MIDP is shown in figure 2.2. The application can also be developed by Original Equipment Manufacturer (OEM). The programmer can use J2ME to develop desired application of his need.

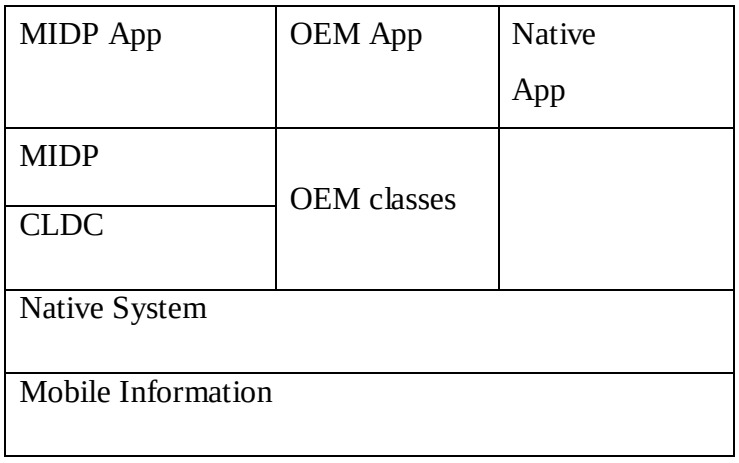

Figure 2.2: MIDP Architecture

2.3.3 Programming Architecture of MIDP 2.0

The MIDP is designed to operate on top of the CLDC. At a high level, the MIDP [17] specification assumes that the MID is limited in its processing power, memory, connectivity, and display size.

The MIDP 2.0 [7] specification is based on the MIDP 1.0 specification and provides backward compatibility with MIDP 1.0 so that MIDlets written for MIDP 1.0 can execute in MIDP 2.0 environments. The MIDP 2.0 specification was designed assuming only CLDC 1.0 features; it will also work on top of CLDC 1.1, and presumably any newer versions. It is anticipated that most MIDP 2.0 implementations in future will be based on CLDC 1.1. The combination of CLDC and MIDP [8] provides a complete environment for creating applications on cell phones and simple two-way pagers. The core of a MID Profile is a MIDlet application. The application extends the MIDlet class to allow the application management software to control the MIDlet, retrieve properties from the application descriptor, and notify and request state changes.

The application management software can manage the activities of multiple MIDlets within a runtime environment. In addition, the MIDlet can initiate some state changes by itself, and notify the application management software of those changes. The programming architecture of MIDP 2.0 is shown in figure 2.3.

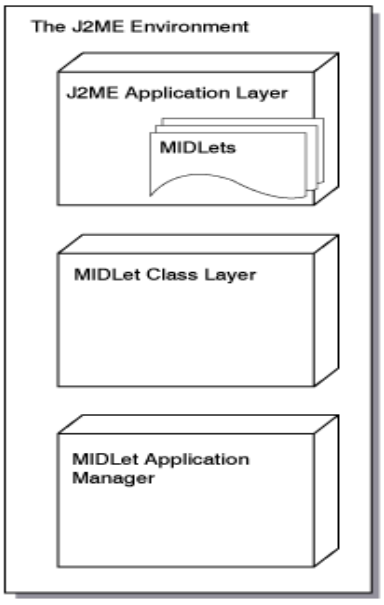

Figure 2.3: Programming Architecture of MIDP 2.0 [5]

- **MIDP APIs for the User Interface**

  These APIs are designed so that interaction with the user is based around a succession of screens, each of which presents a reasonable amount of data to the user. Commands are presented to the user on a per-screen basis. The APIs allow the application to determine what screen to display next, what computation to perform, and what request to make of a network service.

- **MIDP APIs for Handling the Database**

  These APIs organize and manipulate the devices database, which comprises information that remains persistent across multiple invocations of the MIDlet. The underlying CLDC API is used to handle strings, objects, and integers. A subset of the Java 2 API is also provided to handle I/O and network communication.

2.3.4 J2ME Web Services API (WSA)

Web Services are loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols. The technological evolution of Post PC era allows using mobile devices for accessing Web Services. Therefore, the mobile device manufacturers, the service platforms providers and of course the developers have a great opportunity for building high quality applications and services for the massive market that uses Web Services. However, not all Web Services may be available for accessing from mobile devices, due to high processing and memory requirements they may demand. As Web Services are eXtensible Markup Language (XML) based, Simple Object Access Protocol (SOAP) messages used to be verbose, which implies burdensome parsing tasks. On the other hand, not all Web Services defined data types are supported for mobile devices. Therefore, it is required to define access architectures, so that mobile devices may overcome these constraints. This is developed within the Java Community Process as Java Specification Request. This web service architecture has three elements shown in figure 2.4.

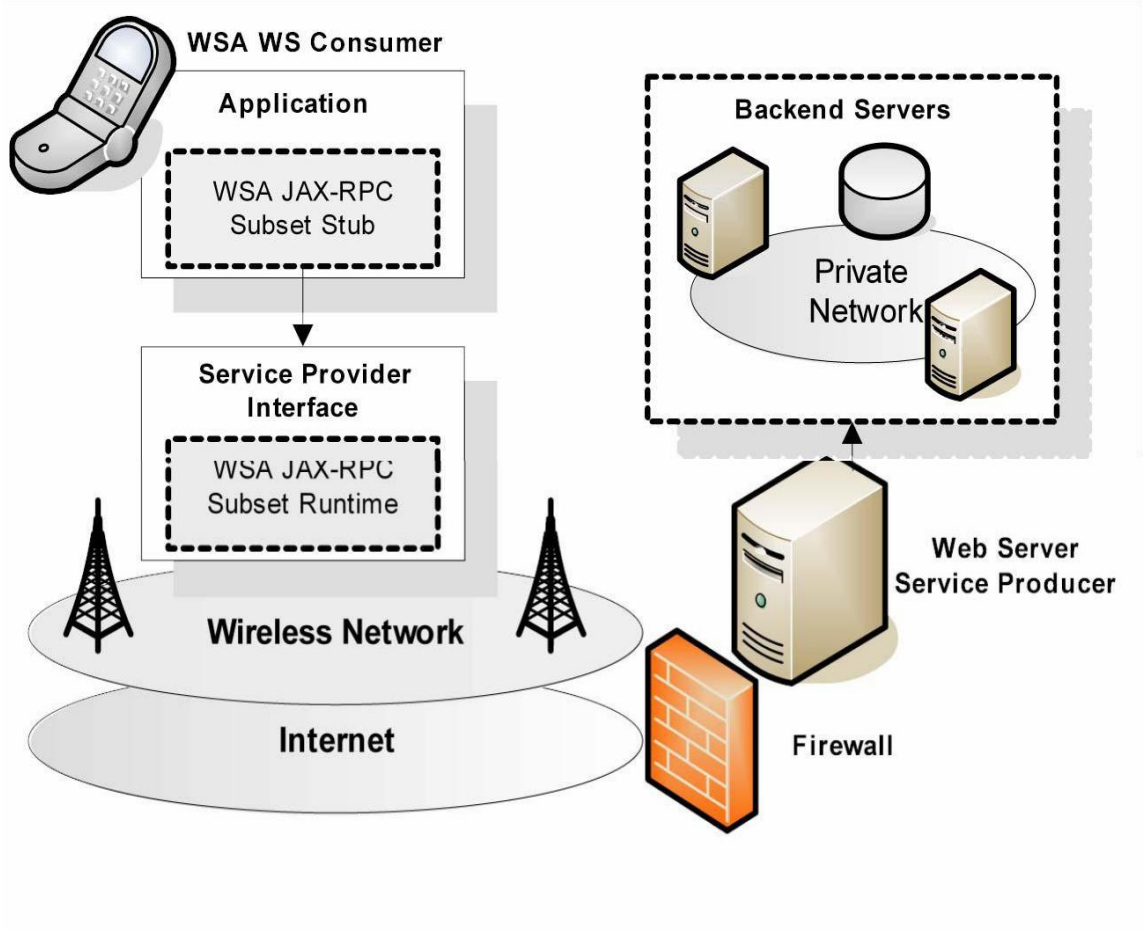

Figure 2.4: WSA working [16]

2.3.5 Life Cycle of MIDlet

The application developed in this thesis is based on Mobile information device profile and these are called MIDlet. The MIDlet of Java 2 Micro Edition is relevant to the Applet of Java 2 Standard edition. Every MIDlet must have three methods as shown in figure 2.5 that constitute the life cycle of MIDlet:

- startApp: startApp method is called when the MIDlet is started. After startApp, the MIDlet is in Active state and the AM (Application management Software) allows it to hold resources. If a runtime exception occurs during startApp, the MIDlet will be destroyed immediately.

- pauseApp: Usage of pauseApp is not supported in S60 and Series 40 platforms. It is still required by the MIDP specification and therefore must be included in the MIDlet although it cannot be called at any point.

- destroyApp: destroyApp signals the MIDlet to terminate and places it in Destroyed state. During termination, all the MIDlet's resources are released and objects deleted. The MIDlet has five seconds to handle the destroyApp call, after which the AMS closes the application itself.

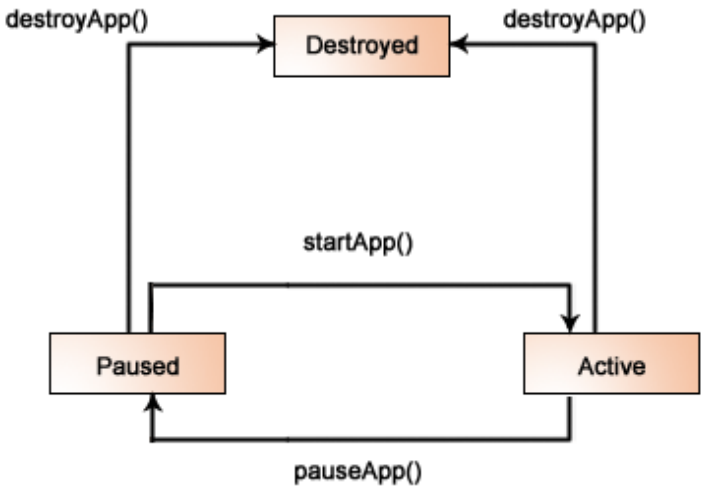

Figure 2.5 MIDlet Lifecycle [17]

By default MIDlet is in the paused states. When the application is executed by default startApp method will call and when close the application the destroyApp method will be called. But when the constructor is not null type, then it will be executed firstly.

Till now, there are different transliteration system that have been developed using different approaches for Indian Languages as well as Foreign Languages. These systems are basically made for desktops. But power of these can be more efficiently utilized by connecting them to mobile phones. Java 2 Micro Edition is a programming platform for developing mobile applications, which can be used to access present transliteration system on mobile phones. J2ME's MIDlet concept can be used to develop these applications.

Chapter 3

Problem Statement

The problem of Language pertains in every region of the world. The globalization has come up with the need for different Languages in Computer system. The people who speak Punjabi traveled around the world and they faced the problem of understanding the universal Language English and the Punjabi NRI's do not understand the Punjabi Language. Most of the countries in the world are using Computer and Internet in their own Languages. But Indian users are deprived to use Computer and Internet resources in their own Languages even in this era of technology, despite dominance of Indian engineers and scientists in the field of information technology throughout the world. This shuts out most of the Indian population from the worldwide web and its huge potential. Thus, there is need to support our Languages on technical front to uplift and improve socio-economic condition of our country.

Punjabi Language is world's 10th most widely spoken Language [15]. It is used in both parts of Punjab, in India and in Pakistan. Punjabi is the Language used by hundreds of millions of people in India, and is also the Language used by Punjabis around the world. Punjab has biggest NRI population and they frequently visit to their motherland. The new generation of these NRI people does not have understanding to the Punjabi Language. They know the words but grammatically they don't know how to write Punjabi. With the help of this application, NRI Punjabi can send a SMS to the transliteration service and receive the SMS as output in the desired Language (Punjabi). The Objective is to make Mobiles end users Interface English SMS to transliterate into Punjabi Language. Mobile Transliteration is helpful to utilize the potentiality of computing system for transliterating one Language to another with minimal human intervention.

3.1 Objectives

In this thesis, the following objectives have been proposed for the development of transliteration mobile application from *Roman* script to *Gurmukhi* script.

1. To understand table for single letter mapping of *Roman* Script to *Gurmukhi* Script.

2. To understand existing rules to transliterate source Language to target Language and Create new rules as per our requirement.
3. To design a database, that will be used to map single unit of English letter to Punjabi letter.
4. To create the transliteration application, that will transliterate English to Punjabi.
5. To design a web interface for mobile application and design mobile application.

3.2 Methodology

To achieve the above discussed objectives, a step-by-step methodology has been followed, which is described below.

1. Comprehensive study of English and Punjabi Languages.
2. Study grammar of English and Punjabi Language and study previously developed systems and rules used in those systems.
3. Study of existing Transliteration Systems and database used by those systems.
4. Web application for Punjabi Transliteration will be developed using JSP at front end and MySql at back end.
5. To develop J2ME mobile application, study of J2ME is required because is a bit different from Java.

Chapter 4

Design and Implementation

As the *Gurmukhi* Script is not available on mobile systems. Our effort is to provide the local Language transliteration system by implementing the transliteration at mobile hand set. This can be done with the help of java programming Language on mobile phone. Web Service is another mean of connecting mobile with the existing Computer based transliteration services. The existing machine Language transliteration is focused on Computer systems. These software's are resource consuming. We target the mobile phones to connect the existing systems and providing desired Language at any desired location to the user. For this purpose, one can use J2ME environment to develop the transliteration application. In J2ME, MIDP is used to create the mobile application.

4.1 Architecture for Purposed Transliteration Mobile Application

Mobile phone application, discussed in this thesis, is based upon client server computing. In this system, mobile application is a client application and transliteration system is a web application. Thus, user put a string in mobile application interface whose transliteration he want to do, then just press ok, this string will be passed to transliteration system means web application, as shown in figure 4.1.

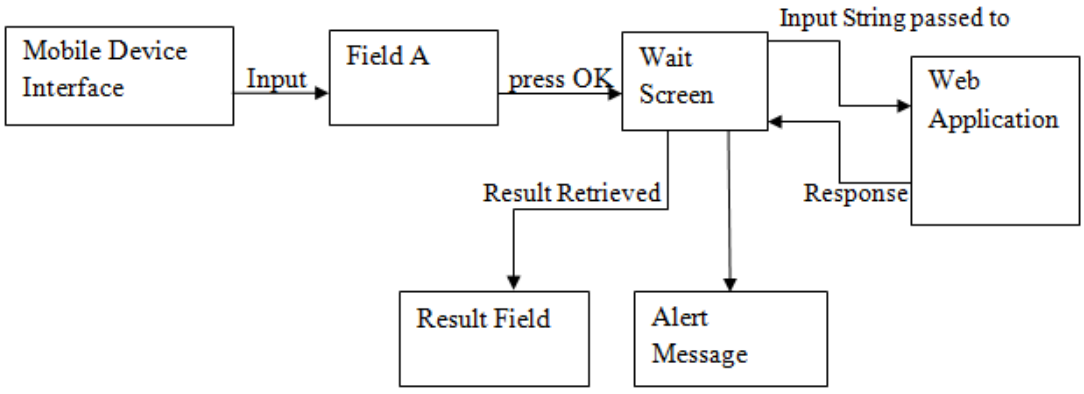

Figure 4.1: Architecture for Mobile Phone Application

Then web application will transliterate the input string with the help of data base and rules, output result is sent back to mobile application. This application is developed in J2ME environment of Netbeans. As shown in figure 4.1, wait screen is sending the

request to web application for transliteration and web application is sending back response. If by any mean, mobile is unable to connect to the web application, then wait screen will show alert massage which is "*unable to connect web app*". Otherwise it will send the result back to result field on mobile phone application. These snapshots are taken from mobile simulator developed in J2ME. This simulator feature is by default provided by J2ME so that developer can check its application on Computer first rather than getting the fault after installation on phone. This application is checked from different Computer and it is running fine. For this purpose, just install client application on remote Computer and then connect to web application by just changing the server address, which was firstly local host for same machine but from remote Computer it would be the IP address of server hosting Computer, in servlet file of mobile application. After doing this minor change, application will start running fine from remote Computer too. Now to run this application from remote Computer, which is not in same network than user have to host our server file from a Computer whose IP is public and access will be possible. Because a private network never allow a connection from outside his network. Thus, with this application, user can utilize the web based applications for transliteration more efficiently. Otherwise, it was wasting our resources, because user can't take the desktop everywhere but mobile phone can be. Here figure 4.2 shows that user input string in the mobile application, on which if user click on menu button then user will get button '*ok*'. Just press the '*ok*' button, mobile phone will show us wait screen so that user knows processing of string is going on at backend. At backend, wait screen transfers the string to web server which in turn transliterates the string and send the result back to mobile phone application, *e.g.*, in figure 4.2 input is '*mai sakool gia*' and after pressing ok button user get output as shown in figure 4.3, which is ਮੈ ਸਕੂਲ ਗਿਆ *mai sakūl giā* '*I went to school*' .

Figure 4.2: Input Interface for Mobile Application

Figure 4.3: Output Interface for Mobile Application

4.2 Working of Purposed Web Application

To transliterate the input string that came from mobile application, web application performs transliteration in two phases. First is tokenization and second is rules and direct mapping. To show the working of this mobile application, take example of input string passed in figure 4.2, which is '*mai sakool gia*'. Transliteration of this string will be done with following steps.

4.2.1 Tokenization

First of all, string passed by user is tokenized by the system, *i.e.*, it will generate three tokens corresponding to string 'mai sakool gia'. Where in this case, first token is '*mai*', second token is '*sakool*' and third token is '*gia*'. Each token will be processed one by one by the system and result of each token is stored in buffer. When all the tokens are processed, than transliterated results stored in buffer are sent back to the mobile device.

- Preprocessing of tokens is also done in tokenization phase. In preprocessing, each token is broken down to character array. As first token is '*mai*'. So, it will be further broken down to character array which contain each character as '*m+a+i*'.

4.2.2 Rules and Direct Mapping

In third step each character of token is processed and corresponding Punjabi words are generated with the help of direct mapping and rules. As character of first token is '*m+a+i*'. System will check database entry corresponding to first character '*m*' output comes out as ਮ 'ma'. Next character is '*a*' and according to schwa rules it is deleted by the system, but there is another condition also according to which it will first check whether next character is '*i*' or '*u*'. As in this case character next to '*a*' is '*i*' so, it will be processed as combination of both character, *i.e.*, system will check corresponding Punjabi word to '*ai*'. Thus, output will be ੈ and thus transliteration corresponding to first token is ਮੈ *mai* '*I*', this string is saved into buffer. After this, next token '*sakool*' will be processed and when all the tokens are transliterated and saved to buffer, then this buffer is returned back to mobile device. Here table 4.1 and table 4.2 shows how system matches the Punjabi Character for given input English character.

Table 4.1: English to Punjabi Consonants Mapping

English Letter	Punjabi Letter	English Letter	Punjabi Letter	English Letter	Punjabi Letter
K	ਕ	dh	ਢ	R	ਰ
Kh	ਖ	N	ਣ	L	ਲ
G	ਗ	T	ਤ	v,w	ਵ
Gh	ਘ	Th	ਥ	rh,r	ੜ
Ng	ਙ	D	ਦ	S	ਸ
Ch	ਚ	dh	ਧ	H	ਹ
Chh	ਛ	N	ਨ	Sh	ਸ਼
J	ਜ	P	ਪ	Khh	ਖ਼
Jh	ਝ	f,ph	ਫ	Ghh	ਗ਼
Yan	ਞ	B	ਬ	Z	ਜ਼
T	ਟ	Bh	ਭ	F	ਫ਼
Th	ਠ	M	ਮ	Lla	ਲ਼
D	ਡ	Y	ਯ	D	

4.3 Rules to Transliterate *Roman* Script to *Gurmukhi* Script

Now, while mapping these words, as same English words are mapped to different Punjabi word depending upon their position in the sentence, *e.g.*, English character '*a*' mapped to ਅ *a* when it is at first position, but it is mapped to ਾ when not at first position. So, there are some rules for characters at first position, middle position and last position. These rules are given in table 4.2.

Table 4.2: Rules for Character Mapping from English to Punjabi

Serial no.	Condition	Action	Example
1.	If '*a*' at first position	Replace it with 'ਅ' rather than ' ਾ'	'*Asman*' will be transliterated to ਅਸਮਾਨ '*asamān*' '*sky*'
2.	If '*e*' at first position	Replace it with 'ਏ' rather than 'ੇ'	'*Ekta*' will be transliterated to ਏਕਤਾ *ēkatā* But '*der*' to ਦੇਰ *dēr* '*late*'
3.	If '*i*' at first position	Replace it with 'ਇ' rather than 'ਿ'	'*is*' will be transliterated to ਇਸ *is* 'this' but '*hit*' to ਹਿਤ *hit*
4.	If '*o*' at first position	Replace it with 'ਓ' rather than 'ੋ'	'*Omkar*' will be transliterated to ਓਮਕਾਰ *ōmkār* but '*chor*' to ਚੋਰ *cōr* '*thief*'
5.	If '*u*' at first position	Replace it with 'ਉ' rather than 'ੁ'	'*Urdu*' will be transliterated to ਉਰਦੁ *urdu* but '*sukh*' to ਸੁਖ *such*
6.	If '*s*' is followed by '*h*'	Transliterate it as '*sh*' not according to '*s*' and '*h*'	'*Shimla*' as ਸ਼ਿਮਲਾ *shimlā* but '*sahara*' as ਸਹਾਰਾ *sahārā*
7.	If '*k*' is followed by '*h*'	Transliterate it as '*kh*' not according to '*k*' and '*h*'	'*khatra*' as ਖਤਰਾ *khatrā* but *kahani* as ਕਹਾਨੀ *kahānī* '*story*'
8.	If '*g*' is followed by '*h*'	Transliterate it as '*gh*' not according to '*g*' and '*h*'	'*Ghar*' as ਘਰ ghar '*home*' but '*gamla*' as ਗਮਲਾ *gamlā* '*pot*'

9.	If '*c*' is followed by '*h*'	Transliterate it as '*ch*' not according to '*c*' and '*h*'	'*Chaacha*' as ਚਾਚਾ *cācā* '*uncle*'
10.	If '*c*' is followed by '*hh*'	Transliterate it as '*chh*' not according to '*c*' ,'*ch*'and '*h*'	'*chhatree*' will be transliterated as ਛਤਰੀ *chatrī* '*umbrella*'
11.	If '*j*' is followed by '*h*'	Transliterate it as '*jh*' not according to '*j*' and '*h*'	'*Jharkhand*' as ਝਾਰਖੰਡ *jhārkhaṇḍ* but '*jahan*' as ਜਹਾਨ *jahān* '*world*'
12.	If '*t*' is followed by '*h*'	Transliterate it as '*th*' not according to '*t*' and '*h*'	'*thand*' will be transliterated as ਠੰਡ *ṭhaṇḍ* '*cold*'
13.	If '*d*' is followed by '*h*'	Transliterate it as '*dh*' not according to '*d*' and '*h*'	'*thandh*' will be transliterated as ਠੰਢ *ṭhaṇḍh* '*cold*'
14.	If '*p*' is followed by '*h*'	Transliterate it as '*ph*' not according to '*p*' and '*h*'	'*phulvarhee*' will be transliterated as ਫੁਲਵਾੜੀ *phulvāṛī*
15.	If '*b*' is followed by '*h*'	Transliterate it as '*bh*' not according to '*b*' and '*h*'	'*bholu*' will be transliterated as ਭੋਲੁ *bhōlu*
16.	If '*r*' is followed by '*h*'	Transliterate it as '*rh*' not according to '*r*' and '*h*'	'*brarh*' will be transliterated as ਬਰਾੜ *barāṛ*
17.	If '*e*' is followed by another '*e*'	Transliterated for '*ee*' and 'ੀ' will be generated instead of 'ੇ'	'*navneet*' will be transliterated as ਨਵਨੀਤ *navnīt* but '*harrnel*' to ਹਰਮੇਲ *harmēl*

18.	If '*o*' is followed by another '*o*'	Transliterated for '*oo*' and '◌ੂ' will be generated instead of '◌ੋ'	'*Sckool*' will be transliterated as ਸਕੂਲ *sakūl* '*school*'
19.	If '*a*' is followed by '*a*'	Transliterated to 'ਆ' rather than '◌ਾ'	'*Hariaalee*' will be transliterated to ਹਰਿਆਲੀ *hariālī*
20.	If '*a*' is followed by '*n*'	Transliterated for '◌ੰ'	'*kandh*' will be transliteratede to ਕੰਧ *kandh* '*wall*'
21.	If '*a*' is followed by '*i*'	Transliterated to '◌ੈ'	'*Mai*' will be transliterated to ਮੈ *mai* '*I*'
22.	If '*a*' is followed by '*u*'	Transliterated to '◌ੌ'	''laura' will be transliterated to ਲੌਰਾ laurā
23.	If '*i*' is followed by '*n*'	Transliterated for '*in*' and '◌ਿ◌ੰ' are generated	'*Maninder*' will be transliterated to ਮਨਿੰਦਰ *manindar*
24.	If '*t*' is followed by '*t*'	Transliterated to 'ਟ' rather than 'ਤ'	'*bhatti*' transliterated to ਭੱਟੀ *bhaṭṭī*

4.4 Development of purposed Mobile Application

The java mobility pack in Netbean has been used to design and implement this mobile application. Creating an application, a mobile application in Netbean 7.1 involves very simple and small steps that are listed below [23].

- Choose File > New Project (Ctrl-Shift-N). Under Categories, select Java ME. Under Projects, select Mobile Application and click Next.

- Enter MobileApplication in the Project Name field. Change the Project Location to a directory on your system. As shown in figure 4.4, the project is named as MobileApplication_Concat_Version1.1.
- Uncheck the Create Hello MIDlet checkbox (it is checked by default). Click Next.
- Leave the Java(TM) Platform Micro Edition SDK 3.0 as the selected Emulator Platform. Click Next, then Finish.

The IDE creates the NetBeansProjects/MobileApplication project folder. The project folder contains all of sources and project metadata, such as the project Ant script as shown in figure 4.4.

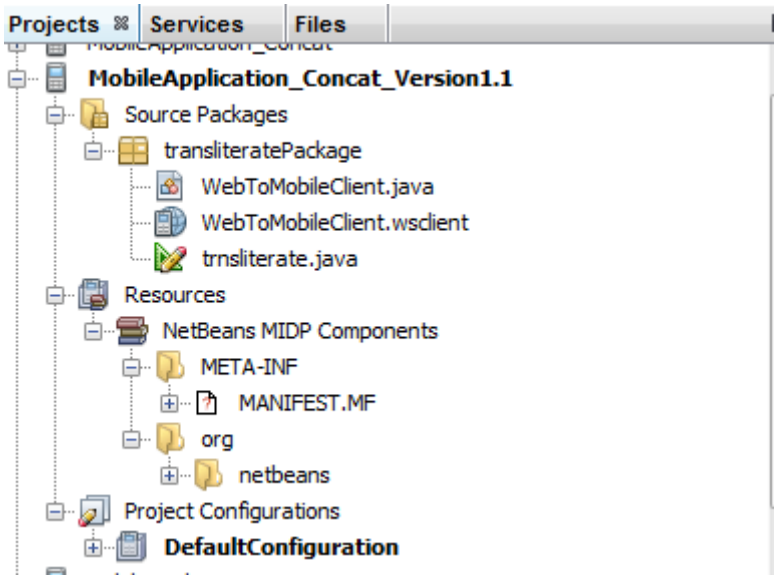

Figure 4.4: Snapshot of Mobile Application containing Ant Script

4.5 Application Development Process

To develop the application Visual Mobile Designer (VMD) of Netbeans has been used. The VMD develops GUIs components rapidly. With the Visual Mobile Designer (VMD) Drag and drop components like wait screens, login screens, file browsers, an SMS composer, and splash screens are included. The Analyzer tool helps us to decrease file size by identifying unused components for removal and it also checks for MIDP 1.0 compliance. The VMD also makes GUI localization easier.

4.6 Adding Package and a Visual MIDlet to Project

After making the project, it is required to add MIDlets to the application. As described in the procedure given below:

- Choose the MobileApplication project in the Projects Window, then choose File > New File (Ctrl-N). Under Categories, select MIDP. Under File Types, select Visual MIDlet. Click Next.
- Enter Transliterate into the MIDlet Name and MIDP Class Name fields, type TransliteratePackage into the Package text field. Click Finish. Figure 4.5 shows how to add visual MIDlet.The application displays in the Flow Design window of the Visual Mobile Designer, as shown in figure 4.6.

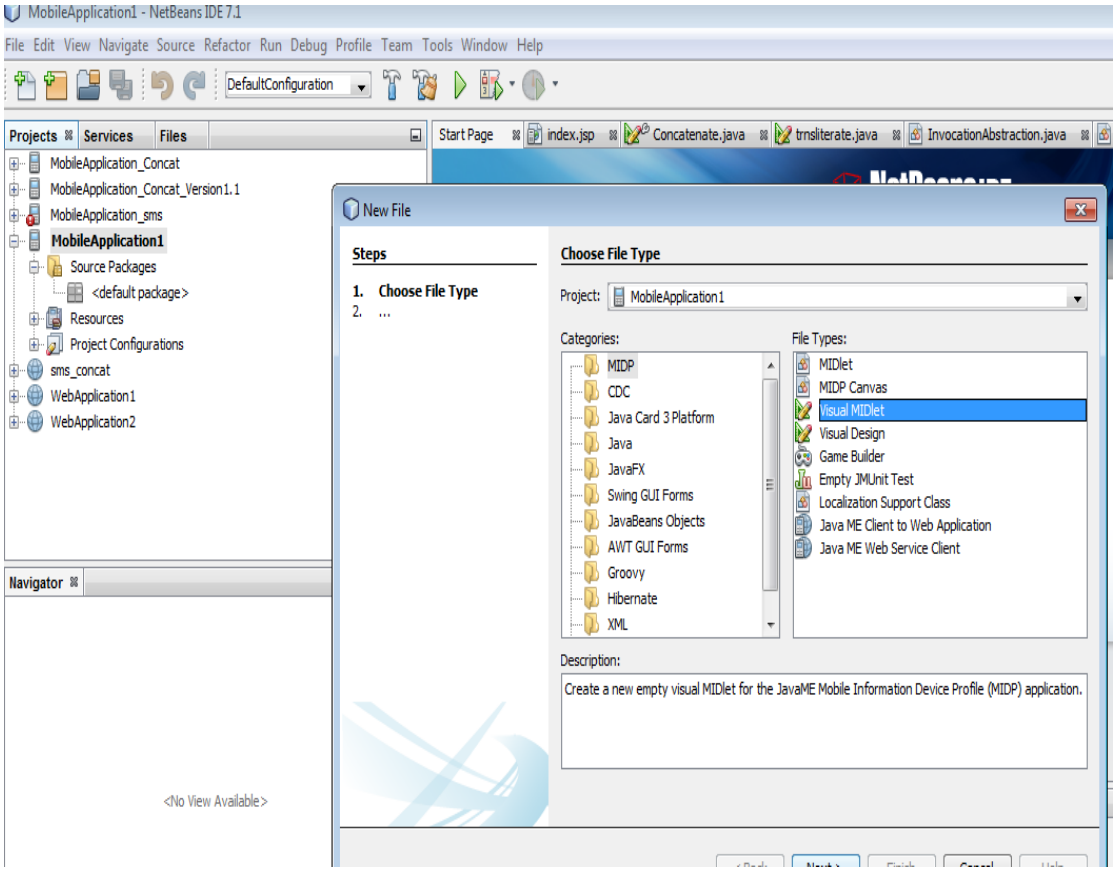

Figure 4.5: Snapshot of adding Visual MIDlet to Mobile Application

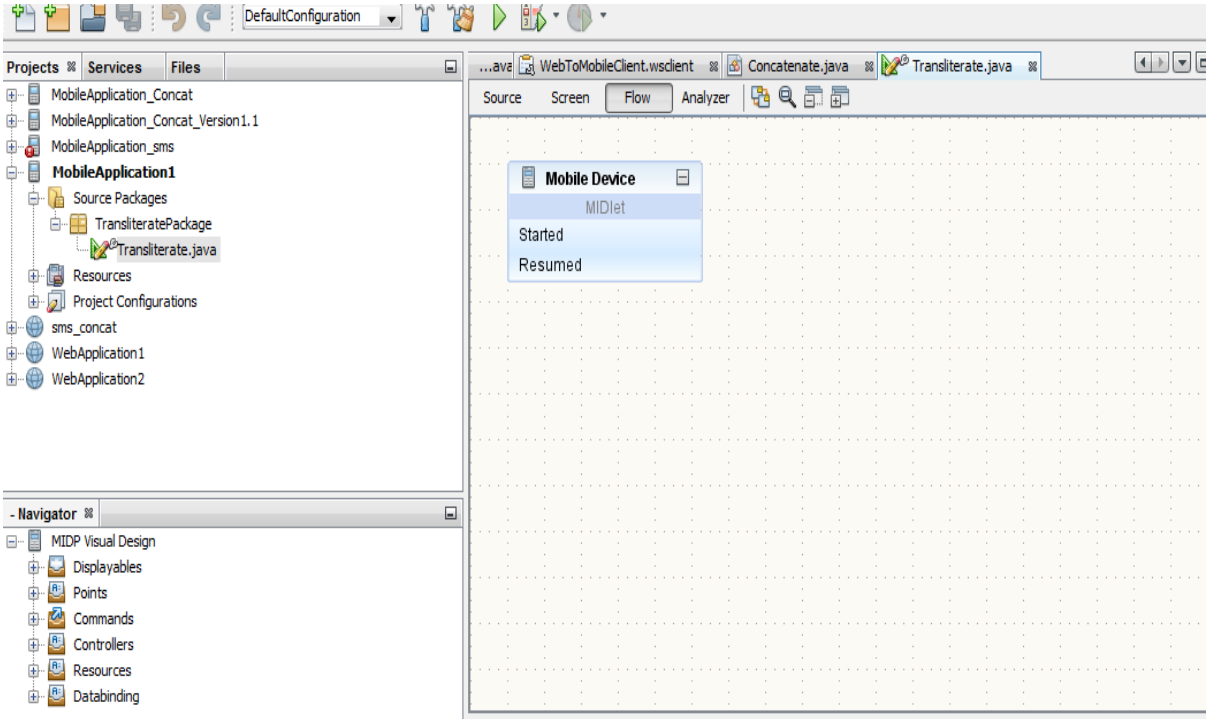

Figure 4.6: Application Display in Flow Design Window

4.7 Adding Components to Project

In the Flow View, drag and drop the following components from the Displayable section of the Palette:

- Wait Screen
- Form (x2)
- Alert

These three are the basic components to develop our application. Figure 4.7 shows how to add components to application. As many components can be added as per requirement. For this transliteration application, components needed are wait screen, two forms and one alert screen. After adding these components, add ext boxes and buttons as per requirement for the application.

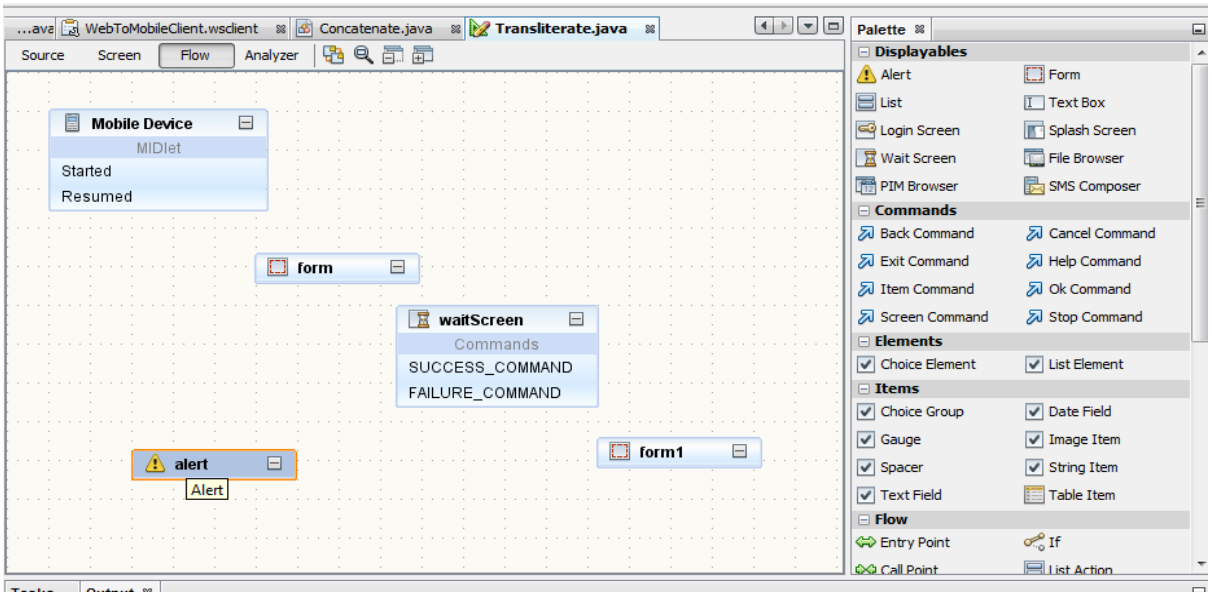

Figure 4.7: Snapshot of Adding Components to Mobile Application

Steps to add text boxes and buttons to our mobile application.

1. Click Screen to switch to the Screen view and choose form in the drop-down list to the right of the Analyzer button.
2. Right-click the form component and choose New/Add > Text Field from the popup menu.
3. Choose the textField component and enter ‘Input’ as the Label value in the Properties window (underneath the Palette).
4. Right-click the ‘Input’ component and choose Rename from the popup menu.
5. In the Rename dialog box, enter Inputfield in the New Name field and click OK.
6. Choose form1 in the drop-down list to the right of the Analyzer button.
7. Right-click the form1 component and choose New/Add > Text Field from the popup menu.
8. Choose the textField component and enter Result as the Label value in the Properties window.
9. Right-click the Result component and choose Rename from the popup menu.
10. In the Rename dialog box enter resultField in the New Name field and click OK.
11. Choose alert in the drop-down list to the right of the Analyzer button.

12. In the Properties window, change the Title property to Alert and change the String property to Error while getting results from the web application. The alert will display in case the web application does not return any result.

4.8 Connecting Components to Create an Application Flow

In the Flow View, click on the Started text on the Mobile Device and drag it to the form component. In the same manner, connect the components together, as shown in the figure 4.8. To design this workflow, just drag the line from where flow starts to the screen where it is to be moved. As shown in the figure, user want that after pressing ok, control moves to wait screen so that user get to know processing is going on at backend. If the wait screen didn't get the output from web server then control goes to alert screen which will tell the user that application is unable to connect web server. Otherwise, control will moved to form 1 which contains the result field. Thus, user will get the transliterated result. In that field, there is a back button too. If user wants to transliterate one more string then if there is no back button then user has to start the application again but with the help of this back button user can go back to input screen any time. Thus, saving user's time, this was wasted in start up of application again and again. There is an exit button too, on this form to quit the application. Thus, user can add buttons as per our convenience.

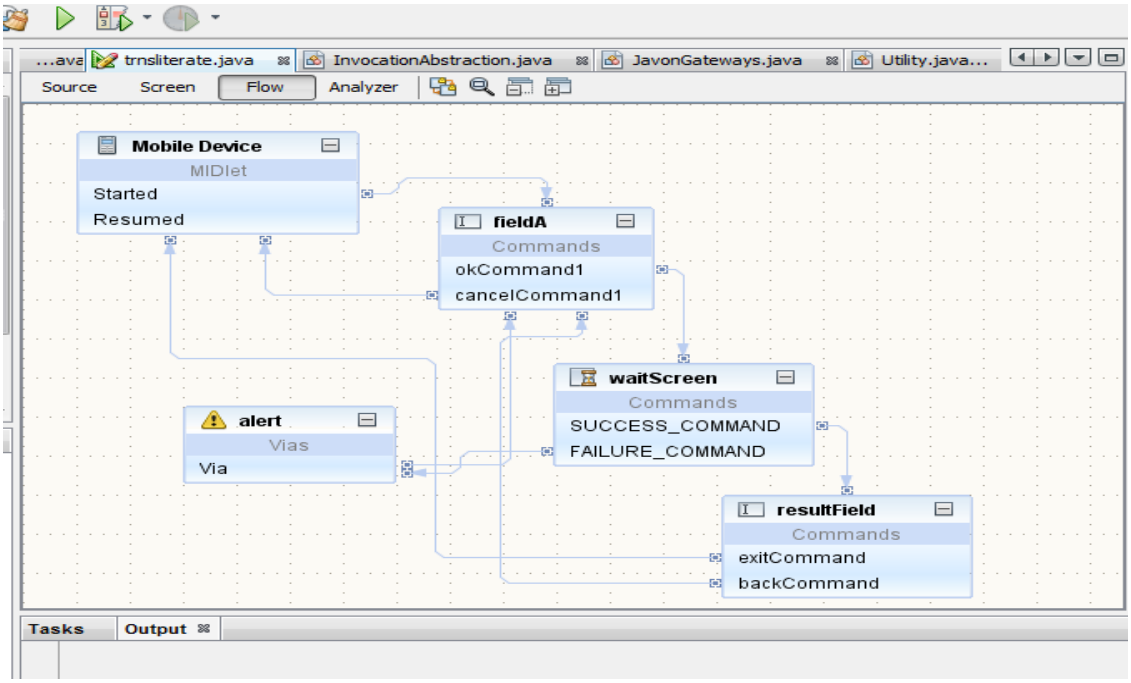

Figure 4.8: Workflow of Mobile Application

4.9 Java ME Client to Web Application Wizard

Java ME Client to Web Application wizard is being used to create a mobile client inside the mobile project with the Transliterate method selection [26]. As shown in figure 4.9, the steps for the same are described below:

1. Expand MobileApplication, right-click Source Packages and choose New > Java ME Client to Web Application.
2. In the Servlet and Client Type panel, choose the type of web application the MIDlet interacts with: enter WebToMobileServlet as the Servlet Name, ensure the Methods in Web Application option is chosen (since the mobile client is going to connect directly to the web application), and click Next.
3. In the Methods in Web Project panel, check the String Transliterate (String a) and click Next.
4. In the Client Options panel, leave the name and package for the generated client class as it is, ensure the Generate stubs and Allow floating point options are checked, and click Finish.

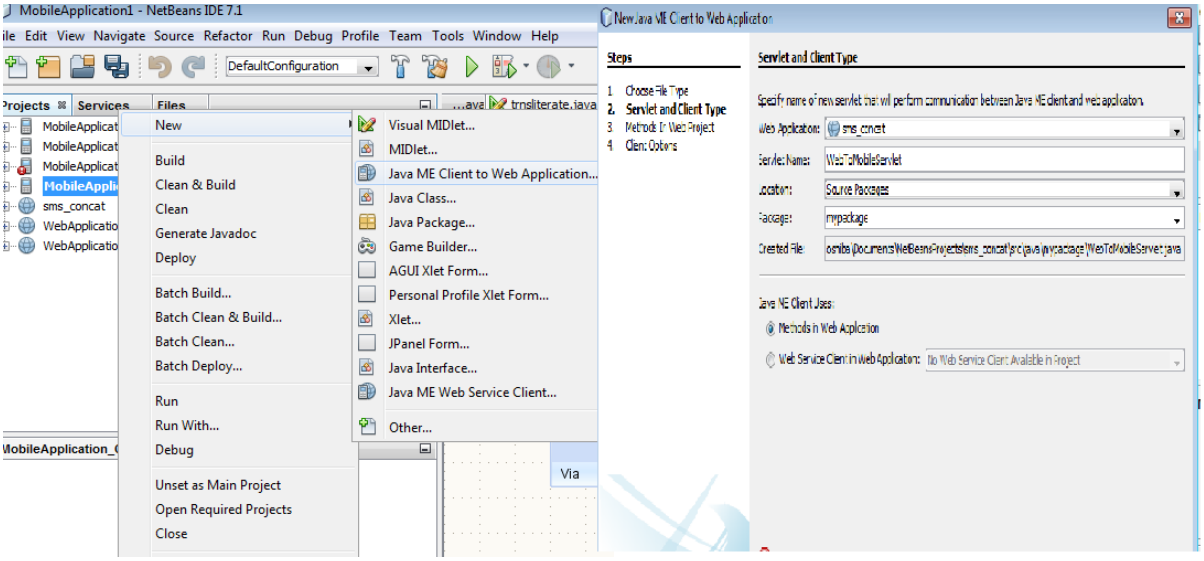

Figure 4.9: Web Application using Java ME client to Web Application

4.10 Algorithms Used in Purposed Web Application for Transliteration

For the purpose of transliteration there is a database table in this application containing four columns, as shown in table 4.3. First two columns contains word map

for characters occurring at first position while next two columns are accessed for rest of characters. There are different algorithms used in this purposed system, which is discussed below:

Algorithm 4.1: Handling Vowels, which Occurs at First Position

1. If tokenized string starts from '*a*','*e*', '*i*', '*o*' or '*u*'.
2. Then check the next character.
3. If it is same as the first character then concatenate them
4. Retrieve the output from input_eng1 and out_pun1.

Algorithm 4.2: Handling 'a' that Found in-between a String

1. If character '*a*' is found and it is not start of the string.
2. Then check the next character in the string.
3. If next character is '*i*', '*u*' or '*n*'.
4. Then concatenate them.
5. Goto Step 10.
6. Else if it is second character in the string.
7. Then just skip it for Schwa Deletion.
8. Else if the previous character is '*i*', '*e*', '*o*', '*u*'.
9. Goto step 11.
10. Retrieve output from input_eng2 and out_pun2.
11. Retrieve output from input_eng1 and out_pun1.

Algorithm 4.3: Handling 'i' that Occurred in-between a String

1. If character found is '*i*' and it is not first character of string.
2. Then check the next character.
3. If it is '*n*' then concatenate them.
4. Retrieve output from input_eng2 and out_pun2.

Algorithm 4.4: Handling Alphabets with Combination with 'h'.

1. If character found is '*c*', '*t*', '*s*', '*p*', '*r*', '*g*', '*b*', '*k*', '*d*'.
2. Then check the next character.
3. If it is '*h*' then concatenate them.
4. Retrieve output from input_eng2 and out_pun2.

Algorithm 4.5: Handling Double Occurrence Consonants

1. If character found is '*t*' or '*d*'.
2. Then check the next character.
3. If it is same as previous character.
4. Then concatenate them.
5. Retrieve output from input_eng2 and out_pun2.

Algorithm 4.6: Handling '*a*' which occur just after any other vowel

1. If character found is '*a*' and it is not first character of string.
2. Then check the previous character.
3. If it is a vowel then replace it with x1.
4. Retrieve the output from input_eng2 and out_pun2.
5. Else from input_eng1 and out_pun1.

Algorithm 4.7: Handling '*c*' having combination with '*hh*'.

1. If character found is 'c'.
2. Then check next two characters.
3. If the both are '*h*'.
4. Then concatenate them.
5. Retrieve the output from input_eng2 and out_pun2.

Algorithm 4.8: Handling case of no match.

1. If no match found from the database for a particular character.
2. Then concatenate next character and again check output corresponding to it.
3. If still no match found.
4. Then give error message "no match found".

Table 4.3 represent the database used by this application for transliteration purpose.

Table 4.3: Database Used in Purposed System

Input_eng1	Out_pun1	Input_eng2	Out_pun2
s	ਸ	s	ਸ
h	ਹ	h	ਹ

Input_eng1	Out_pun1	Input_eng2	Out_pun2
k	ਕ	k	ਕ
kh	ਖ	kh	ਖ
g	ਗ	g	ਗ
gh	ਘ	gh	ਘ
m	ਮ	m	ਮ
ai	ਐ	ai	ੈ
n	ਨ	n	ਨ
oo	ੂ	oo	ੂ
l	ਲ	l	ਲ
a	ਅ	xa	ਆ
th	ਥ	th	ਥ
t	ਤ	t	ਤ
d	ਦ	d	ਦ
dh	ਧ	dh	ਧ
p	ਪ	p	ਪ
ph	ਫ	ph	ਫ
b	ਬ	b	ਬ
bh	ਭ	bh	ਭ
		a	ਾ
in	ਇੰ	in	ਿਂ
r	ਰ	r	ਰ
j	ਜ	j	ਜ
i	ਇ	i	ਿ
sh	ਸ਼	sh	ਸ਼
o	ਓ	o	ੋ
v	ਵ	v	ਵ
au	ਔ	au	ੌ
e	ਏ	e	ੇ
ee	ਈ	ee	ੀ

Input_eng1	Out_pun1	Input_eng2	Out_pun2
ch	ਚ	ch	ਚ
u	ੁ	u	ੂ
w	ਵ	w	ਵ
y	ਯ	y	ਯ
		rh	ੜ
f	ਫ	f	ਫ
		un	ੁੰ
tt	ਟ	tt	ਟ
chh	ਛ	chh	ਛ
		x1	ਅ
		an	ਂ

Chapter 5

Result and Discussion

This chapter presents the results given by the application for the different type of the cases. When user run the application through Netbeans, the emulator starts and asks the user to launch the application. The application begins with text screen on emulator shown in figure 5.1 that shows the user what it actually do.

Figure 5.1: Starting Screen of Application

5.1 Case 1: Transliteration of Single Word

In this case, when the user enters the single word of English and wants the output or transliteration of that word in Punjabi. The user enter the word in English, as shown in figure 5.2 is '*parmatma*' and after entering it, when the user press OK button, the control transfers to next screen with text box named Result along with the output as word in Punjabi, *i.e.*, ਪਰਮਾਤਮਾ *parmātmā* '*God*'. Here, in this case, user has entered a single word of English which is transliterated by the system fine. But if user want transliteration of a full string, that is discussed in next case.

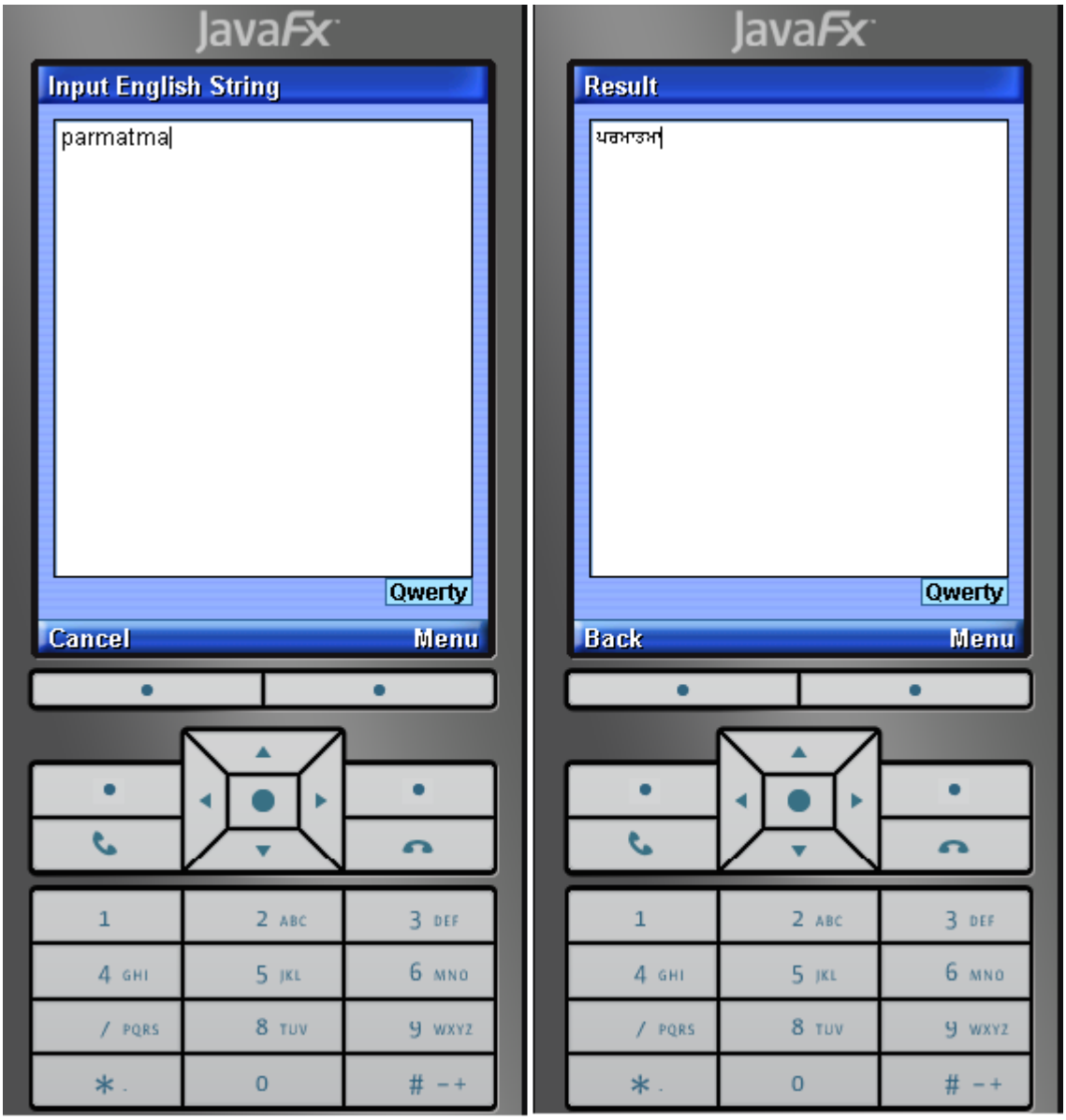

Figure 5.2: Snapshot for Transliteration of Single Word

5.2 Case 2: Transliteration of Sentence

When the user has entered the text that is more than one word, the application will do transliteration of each word one by one and combined result will be sent back to mobile application. The file also consists of commonly used Punjabi phrases and that can be used directly by the application for translation. Now, when the user presses ok, the complete transliterated text would appear in the emulator screen as shown in the figure 5.3.

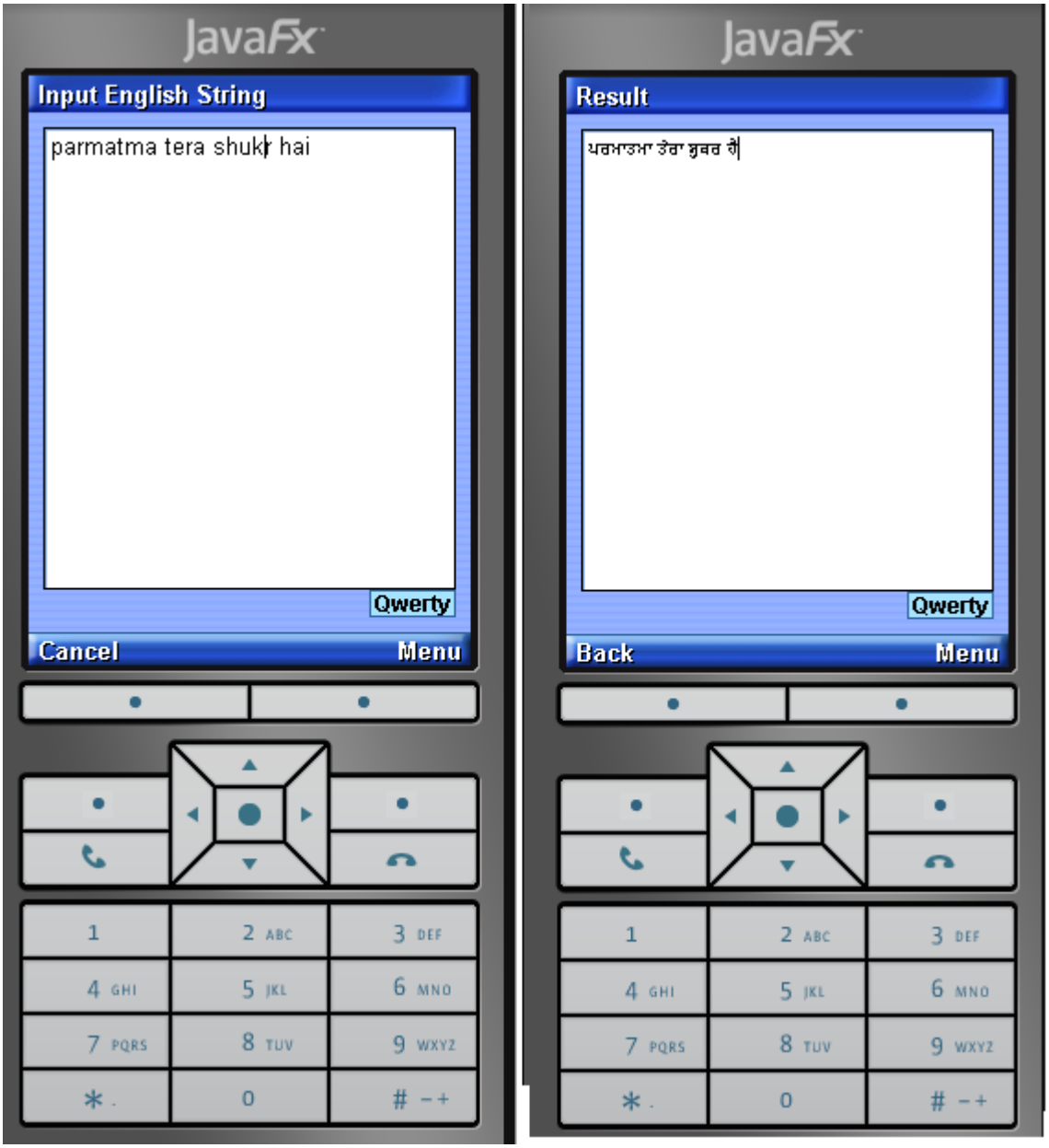

Figure 5.3: Snapshot for Transliteration of Full String

Note: if for any reason the web application is down then it will show a alert message on screen. This is the case when the user enters the text and the application does not find the text from the web application as shown in the figure 5.4. The application will show the message to try again. This is the case when the mobile application is unable to connect to web application or web server is down due to any reason. So that user would not keep on waiting for the output, we have provided an alert message to tell user that web application is down.

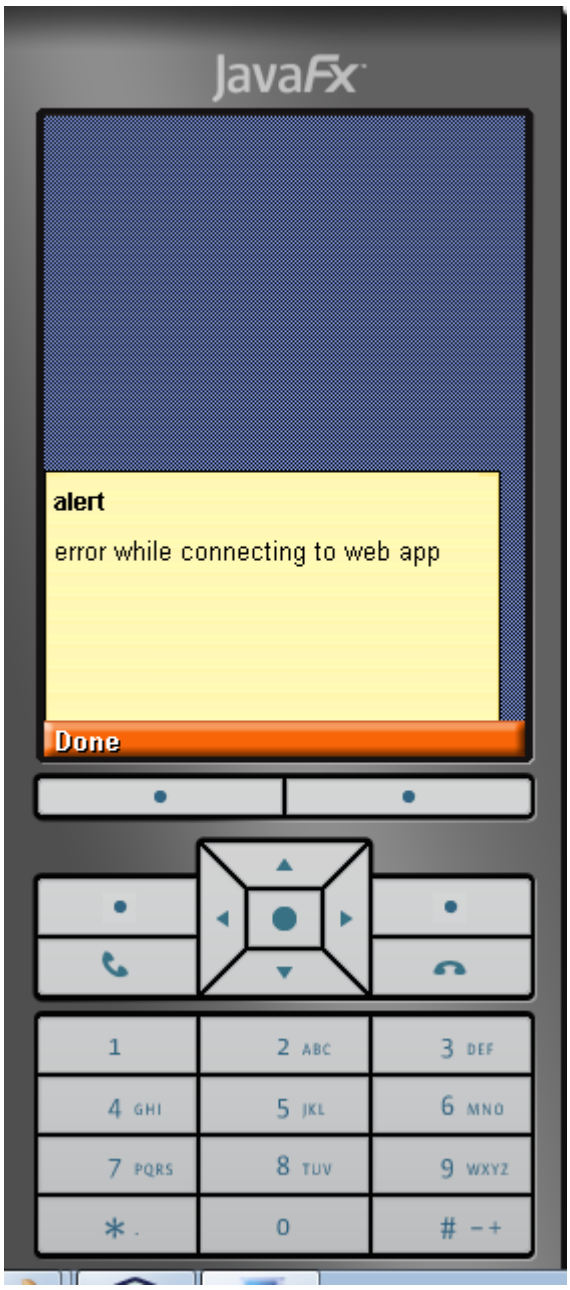

Figure 5.4: Snapshot Showing Web Application is Down

5.3 Results Obtained from Mobile Application

Some words are processed in the mobile application, which are commonly used by a person in daily life. Table 5.1 shows the output generated by mobile application.

Table 5.1: Test Results of Mobile Application

Sr. No.	Input Given in System	Output Retrieved	English Meaning
1.	Mai meetting vich ha	ਮੈ ਮੀਟਿੰਗ ਵਿਚ ਹਾ	I am in meeting
2.	Mai lett ha	ਮੈ ਲੇਟ ਹਾ	I am late
3.	Mai ik bje uthe hovaga	ਮੈ ਇਕ ਬਜੇ ਉਥੇ ਹੋਵਾਗਾ	i will be there at one
4.	Mai ghare ha kirpa karke phon karo	ਮੈ ਘਰੇ ਹਾ ਕਿਰਪਾ ਕਰਕੇ ਫੋਨ ਕਰੋ	i am at home please call
5.	Mai kam te ha kirpa karke phon karo	ਮੈ ਕਮ ਤੇ ਹਾ ਕਿਰਪਾ ਕਰਕੇ ਫੋਨ ਕਰੋ	i am at work please call
6.	Mai do bje phuch javaga	ਮੈ ਦੋ ਬਜੇ ਪਹੁਚ ਜਾਵਾਗਾ	i will arrive at two
7.	Meetting nahee ho rahee hai	ਮੀਟਿੰਗ ਨਹੀ ਹੋ ਰਹੀ ਹੈ	meeting is cancelled
8.	Mai tainoo tinn bje milda ha	ਮੈ ਤੈਨੂ ਤਿੰਨ ਬਜੇ ਮਿਲਦਾ ਹਾ	see you at three
9.	Mai tainoo subah milda ha	ਮੈ ਤੈਨੂ ਸੁਬਾਹ ਮਿਲਦਾ ਹਾ	see you in morning
10.	Maaf karna mai teree is wich madd nahee kar sakda	ਮੈ ਤੇਰੀ ਇਸ ਵਿਚ ਮਦਦ ਨਹੀ ਕਰ ਸਕਦਾ	sorry i cant help u on
11.	Jnm din mubark	ਜਨਮ ਦਿਨ ਮੁਬਾਰਕ	Happy birthday
12.	Mai chhutee te ha	ਮੈ ਛੁਟੀ ਤੇ ਹਾ	I am on leave
13..	Kirpa karke mainu mel karo	ਕਿਰਪਾ ਕਰਕੇ ਮੈਨੂ ਮੇਲ ਕਰੋ	Please mail me
14.	Mainu kaal na karo	ਮੈਨੂ ਕਾਲ ਨਾ ਕਰੋ	Don’t call me

Sr. No.	Input Given in System	Output Retrieved	English Meaning
15.	Mai viast ha	ਮੈ ਵਿਅਸਤ ਹਾ	I am busy
16.	Mainu preshan na karo	ਮੈਨੂ ਪਰੇਸ਼ਾਨ ਨਾ ਕਰੋ	Don’t disturb me
17.	Mai daftr to bahr ha	ਮੈ ਦਫਤਰ ਤੋ ਬਾਹਰ ਹਾ	I am out of office
18.	Chalo lanch karn chalde ha	ਚਲੋ ਲੰਚ ਕਰਨ ਚਲਦੇ ਹਾ	Let’s go for lunch
19.	Mai bimar ha	ਮੈ ਬਿਮਾਰ ਹਾ	I am sick
20.	Chalo film dekhn chaliye	ਚਲੋ ਫਿਲਮ ਦੇਖਨ ਚਲਿਯੇ	Let’s go for movie
21.	aj chhuttee hai	ਅਜ ਛੁਟੀ ਹੈ	Today is holiday
22.	Saalgirah mubark	ਸਾਲਗਿਰਾਹ ਮੁਬਾਰਕ	Happy anniversary
23.	Mubarka	ਮੁਬਾਰਕਾ	congratulations
24.	Paarttee kithe hai	ਪਾਰਟੀ ਕਿਥੇ ਹੈ	Where is party
25.	Chalo baahr chalde ha	ਚਲੋ ਬਾਹਰ ਚਲਦੇ ਹਾ	Lets’s go out.

5.4 Error Analysis

As shown in table given above, twenty five different sentences are processed in this mobile application. Out of these sentences unique words processed are 55, from which 11 are incorrect and 44 are correct according to grammar. Incorrect words are mostly due to dependent vowels like ‘ੱ’(Adhak), ‘ਂ’ (Bindi), ‘ੰ’ (Tippi), *etc*. Based is approximately 80 %, which can further be improved with introducing of a database, which will act as dictionary to this system.

Chapter 6

Conclusions and Future Scope

The study incites to bridge the existing transliterating, dictionary system with the new vibrant, happening and emerging technology of Mobile phone. With the advancement of the infrastructure in wireless network and improvement in the mobile phone hardware capacity, this problem can be solved with the optimum solutions.

What has always driven the market for portable electronic devices is the unquenchable consumer demand for access to virtually unlimited amounts of information, anywhere, anytime. Wireless phones are no longer just used for voice communications. The growth of mobile devices brings new challenges and opportunity for computational support for cross Language communication.

6.1 Conclusion

Most of Punjabi NRI person don't know how to write word in Punjabi. But with the help of transliteration they will be able to write in Punjabi because they know phonetically, what to write in *Roman* script and transliteration application will convert it into *Gurmukhi* script. Transliteration can be done with different approaches like rule based approach, statistical approach or hybrid approach. There are many challenges in transliterating *Roman* script to *Gurmukhi* script, because there is large character gap in both the scripts. As in *Roman* Script there are 21 consonant and 5 vowels only, but in *Gurmukhi* script there are 41 consonants and 19 vowels. So it is a bit difficult to map these characters. In rule based systems, rules are made to overcome these challenges. From these challenges, Schwa deletion is most trivial task.

Different transliteration systems for different Languages have been discussed for Computer system. But Computers are not as much portable as mobiles are. So, to utilize the potentiality of computing system for transliterating one Language to another with minimal human intervention, a transliteration application for mobile phones is created. This application is used to convert input English text in *Roman* script to output Punjabi text in *Gurmukhi* script. To develop this mobile application, java J2ME platform is used. This application simply sends a request to web application developed, which in turn does the transliteration tasks with the help of

rules and table for mapping English words to Punjabi words. The rules defined in this application, play an important role in transliteration. This application uses 24 rules and a mapping table for transliteration purpose. There occurred many problems in defining these rules as there is huge character gap in English and Punjabi so, more than one word of Punjabi can be represented by single word of English, *e.g.*, there is a rule for schwa deletion that, '*a*' proceeding first Punjabi word after transliteration are silent or should be deleted. This rules work fine with '*maninder*', '*ravi*' *etc.* But in case of '*mansa*', '*a*' is giving sound of '◌ਾ' (kanna) but according to this rule it is silent so mobile application will give incorrect output as ਮਨਸਾ *mansā* instead of ਮਾਨਸਾ *mānsā*. Finally, testing of this system is done with the 25 most commonly used sentences, with 55 unique words and system come out with accuracy of approximately 80%.

6.2 Future Scope

To use this application on a mobile phone, the user has to download it from the source, *i.e.*, from Computer System or from website. This limits the scope of utilization of the application. The transliteration of the words is also limited due to scarcity of space in mobile phone. The following advancements can be incorporated in the current solution:

1. The accuracy and quality of work on transliteration can be extended by implying the example based or statistical based methods of machine Language transliteration techniques. This will improve the current accuracy and scope of the existing application.
2. There is scope to dictionary to our transliteration system, to increase accuracy. The short message can be used for sending the request and receiving the requested target Language. The web service for the existing Punjabi translation/transliteration websites like AKHAR, KHOJ *etc.* can be created and then mobile phones can be connected to them.
3. The Punjabi Language translation/transliteration can be extended to other Languages like Hindi, Marathi, Gujrati, *etc*.

References Annexure I

[1] Ali and Ijaz, "*English to Urdu Transliteration System*", Proceedings of the Conference on Language & Technology, pp: 15-23, 2009.

[2] Chinnakotla *et. al*., "*Transliteration for Resource Scarce Language*", ACM Transactions on Asian Language Information Processing, vol. 9, no. 4, pp: 1-30, 2010.

[3] Deep and Goyal, "*Development of A Punjabi to English Transliteration System*", International Journal of Computer Science and Communication, 2011.

[4] English Language available at "*http://en.wikipedia.org/wiki/English_Language*", last accessed on June 2012.

[5] Gosh, Architecture of J2ME available at "*http://www.ibm.com/developerworks/library/wi-j2me/*" last accessed on June 2012.

[6] Hong *et. al*., "*A Hybrid Approach to English-Korean Name Transliteration*", Proceedings of the Named Entities Workshop, pp: 108–111, Suntec, Singapore, 2009.

[7] Java Community Process "*Mobile Information Device Profile JSR 118 version 2.0*" for Java 2 Micro Edition available at "*http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html*" last accessed on June 2012.

[8] Jonathan, "*Wireless Java: Developing with J2ME, Second Edition*". pp: 8-18, 2003.

[9] Kaur and Josan, "*Statistical Approach to Transliteration from English to Punjabi*", International Journal on Computer Science and Engineering, vol. 3, no. 4, pp: 1518-1527, 2011.

[10] Lawton, "*Moving Java into Mobile Phones*", Computer, vol. 35, no. 6, pp: 17–20, 2002.

[11] Lehal and Josan, "*A Punjabi to Hindi Machine Translation System*", Coling 2008: Companion volume Posters and Demonstrations, pp: 157-160, Manchester, 2008.

[12] Malik, "*Punjabi Machine Transliteration System*", In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pp: 1137-1144, 2006.

[13] Nicolas *et. al*., "*Mobile Information Device Profile white paper*" available at "*http://java.sun.com/products/midp/midp-ds.pdf*", last accessed on June 2012.

[14] Oh and Choi, "*An English-Korean Transliteration Model*" using Pronunciation and Contextual Rules, In Proceedings of the 19th International Conference on Computational Linguistics, pp: 393–399, 2002.

[15] Punjabi Language available at "*http://en.wikipedia.org/wiki/Punjabi_Language*", last accessed on June 2012.

[16] Rendón *et. al*., "*Architectures for Web Services Access from Mobile Devices*". Available at "*http://www.computer.org/portal/web/csdl/doi/10.1109/LAWEB.2005.9*", last accessed on June 2012.

[17] Sun Microsystems JCP Expert Group, "*Mobile Information Device Profile Specifications for Java™ 2 Micro Edition Version 2.0*". Available at "http://jcp.org/en/jsr/ec", last accessed on June 2012.

[18] Sun Microsystems J2ME Specification available at "*http://java.sun.com/j2me/index.jsp*", last accessed on June 2012.

[19] Sun Microsystems J2ME Specification, Available at "*http://java.sun.com/j2me/index.jsp*" last accessed on June 2012.

[20] Vijaya *et. al*., "*English to Tamil Transliteration using WEKA system*", International Journal of Recent Trends in Engineering, vol. 1, no. 1, pp: 498-500, 2009.

[21] Wan and Verspoor, "*Automatic English-Chinese name transliteration for development of multilingual resources*". In Proceedings of the 17th International Conference on Computational Linguistics, pp: 1352–1356, Stroudsburg USA, 1998.

[22] Zelenko and Aone, "*Discriminative methods for transliteration*", In Proceedings of the Conference on Empirical Methods in Natural Language Processing Association for Computational Linguistics, pp: 612-617, Stroudsburg USA, 2006.

Papers Annexure II

[1] Singh *et. al*. "*Development of Mobile Application for Transliteration of Roman Script to Gurmukhi Script*", International Journal of Computational Linguistics and Natural Language Processing, 2012.
Communicated