***Mobile Computing Lab***

***ETIT-452***

**Faculty Name: Student Name:**

**Enrollment No:**

**Semester:8th**

**Batch: 8-I-**



**Department of Information Technology**

Maharaja Agrasen Institute of Technology,PSP area,

Sector-22,Rohini, New Delhi-110085

**Mobile Computing Lab**

**PRACTICAL RECORD**

Paper Code : ETIT - 452

Name of the student :

University Roll No. :

Branch : Information Technology

Section/Group : 8-I-

**PRACTICAL DETAILS**

Experiments according to the lab syllabus prescribed by GGSIPU

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No. | Experiment Name | Date of Performance | Date of Checking | Remarks | Marks |
| 1. | Introduction of WML and J2ME. |  |  |  |  |
| 2. | Write a program to design a Calculator by using WML/J2ME. |  |  |  |  |
| 3. | Write a program to design a Calendar by using WML/J2ME. |  |  |  |  |
| 4. | To write a program to design a timer by using WML/J2ME. |  |  |  |  |
| 5. | Write a program to design and implement a simple game using WML. |  |  |  |  |
| 6. | Write a program to animate an image using WML. |  |  |  |  |
| 7. | Write a program to design and implement a Phonebook using WML. |  |  |  |  |
| 8. | To Study the Authentication and encryption technique used in GSM. |  |  |  |  |

***Experiment 1***

***Aim :-***Introduction ofWML and J2ME.

**WML**

* WML stands for **W**ireless **M**arkup **L**anguage
* WML is an application of XML, which is defined in a document-type definition.
* WML is based on HDML and is modified so that it can be compared with HTML.
* WML takes care of the small screen and the low bandwidth of transmission.
* WML is the markup language defined in the WAP specification.
* WAP sites are written in WML, while web sites are written in HTML.
* WML is very similar to HTML. Both of them use tags and are written in plain text format.
* WML files have the extension ".wml".
* WML supports client-side scripting. The scripting language supported is called WMLScript.

**WML Versions**

WAP Forum has released a latest version WAP 2.0. The markup language defined in WAP 2.0 is XHTML Mobile Profile (MP). The WML MP is a subset of the XHTML. A style sheet called WCSS (WAP CSS) has been introduced alongwith XHTML MP. The WCSS is a subset of the CSS2.

Most of the new mobile phone models released are WAP 2.0-enabled. Because WAP 2.0 is backward compatible to WAP 1.x, WAP 2.0-enabled mobile devices can display both XHTML MP and WML documents.

WML 1.x is an earlier technology. However, that does not mean it is of no use, since a lot of wireless devices that only supports WML 1.x are still being used. Latest version of WML is 2.0 and it is created for backward compatibility purposes. So WAP site developers need not to worry about WML 2.0.

**WML Program Structure**

A main difference between HTML and WML is that the basic unit of navigation in HTML is a page, while that in WML is a card. A WML file can contain multiple cards and they form a deck.

When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. So if the user goes to another card of the same deck, the mobile browser does not have to send any requests to the server since the file that contains the deck is already stored in the wireless device.You can put links, text, images, input fields, option boxes and many other elements in a card.

Following is the basic structure of a WML program:

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"  "http://www.wapforum.org/DTD/wml12.dtd">  <wml>  <card id="one" title="First Card">  <p>  This is the first card in the deck  </p>  </card>  <card id="two" title="Second Card">  <p>  Ths is the second card in the deck  </p>  </card>  </wml> |

The first line of this text says that this is an XML document and the version is 1.0. The second line selects the document type and gives the URL of the document type definition (DTD).

One WML deck (i.e. page ) can have one or more cards as shown above. We will see complete detail on WML document structure in subsequent chapter.

Unlike HTML 4.01 Transitional, text cannot be enclosed directly in the <card>...</card> tag pair. So you need to put a content inside <p>...</p> as shown above.

Develop WAP applications you will need the following:-

* **A WAP enabled Web Server:** You can enable your Apache or Microsoft IIS to serve all the WAP client request.
* **A WAP Gateway Simulator** This is required to interact to your WAP server.
* **A WAP Phone Simulator:** This is required to test your WAP Pages and to show all the WAP pages.

You can write your WAP pages using following languages:-

* Wireless Markup Language(WML) to develop WAP application.
* WML Script to enhance the functionality of WAP application.

**Java 2 Platform, Micro Edition (J2ME)**

Traditional computing devices use fairly standard hardwareconfigurations such as display, keyboard,large amount of memory and permanent storage. However new breed of computing devices lacs hardware configuration. J2ME is special y designed fordeveloping applications for small computing devices such as cell phones, PDA etc.

**J2ME Configurations:**

Configuration defines the JVM for a particular small computing device.

**Types:-**

1. CLDC (Connected limited Device configuration) :CLDC is used for the devices with the limited resources. CLDC devices usestripped version of JVM called KVM. CLDC devices are mobile phones, PDA etc.
2. CDC ( Connected device configuration) : CDC devices use complete

JVM. CDC devices are set-top box, Home appliances such as Air conditioner etc.

**J2ME Profiles**

Profile consists of classes that enable developers to implements features found in a related group of small computing devices.

Many profile are available. Here we use MIDP ( Mobile Information Device Profile)MIDP is used with CLDC configuration that provides classes for local storage, a userinterface and networking capabilities. Other profiles are Game profile, Foundation profile, RMIprofile and many more.

**Java Virtual Machine layer :** This layer is an implementation of a Java Virtual Machine that is customized for a particular device's host operating system and supports a particular J2ME configuration.

**Configuration layer :** The configuration layer defines the minimum set of Java Virtual Machine features and Java class libraries available on a particular category of devices. In a way, a configurationdefines the commonality of the Java platform features and libraries that developers can assume to be available on all devicesbelonging to a particular category. This layer is less visible to users, but is very important to profile implementers.

**Profile layer :** The profile layer defines the minimum set of application programming interfaces (APIs) available on a particular family of devices.

Profiles are implemented upon a particular configuration. Applications are writtenfor a particular profile and are thus portable to any device that supports that profile.

**MIDP layer :** The Mobile Information Device Profile (MIDP) is a set of Java APIs that addresses issues such as user interface, persistence storage and networking.

***Experiment – 2***

***Aim :-*** Write a program to design aCalculator by using WML/J2ME.

**Code**

**Calc.wml**

<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML

1.1//EN" "http://www.wapforum.org/DTD/wml\_1.1.xml">

<wml>

<!-- Call init routine when deck is first entered,

or when card is called -->

<card id="init">

<onevent type="onenterforward">

<go href="calc.wmls#init()"/>

</onevent>

<p>

Initializing...

</p>

</card>

<!-- Display current values and operator -->

<card id="display">

<p>

<select>

<option onpick="#num1">$(num1).$(dec1)</option>

<option onpick="calc.wmls#changeop()">$(operator)</option>

<option onpick="#num2">$(num2).$(dec2)</option>

<option onpick="calc.wmls#domath()">Compute</option>

</select>

</p>

</card>

<!-- Accept input for value 1 -->

<card id="num1">

<do type="accept">

<go href="#display"/>

</do>

<!-- Offer easy way to clear values

(options key) -->

<do type="options" label="clear">

<refresh>

<setvar name="num1" value=""/>

<setvar name="dec1" value=""/>

</refresh>

</do>

<p>

Enter the value:

<input name="num1" maxlength="8" format="\*N"/><br/>.<br/>

<input name="dec1" maxlength="3" format="\*N"/>

</p>

</card>

<!-- Accept input for value 2 -->

<card id="num2">

<do type="accept">

<go href="#display"/>

</do>

<!-- Offer easy way to clear values

(options key) -->

<do type="options" label="clear">

<refresh>

<setvar name="num2" value=""/>

<setvar name="dec2" value=""/>

</refresh>

</do>

<p>

Enter the value:

<input name="num2" maxlength="8" format="\*N"/><br/>

.<br/>

<input name="dec2" maxlength="3" format="\*N"/>

</p>

</card>

<!-- Display result of calculation -->

<card id="result">

<do type="accept">

<go href="#init"/>

</do>

<p>

The result of <br/>

$(num1) $(operator) $(num2) is:<br/>

$(result)

</p>

</card>

</wml>

**Calc.wmls**

// Initialize variables

extern function init()

{

WMLBrowser.setVar("num1","");

WMLBrowser.setVar("dec1","");

WMLBrowser.setVar("num2","");

WMLBrowser.setVar("dec2","");

WMLBrowser.setVar("operator","+");

WMLBrowser.setVar("result",0);

WMLBrowser.go("calc.wml#display");

}

// Rotate operator through + - \* /

extern function changeop()

{

var operator = WMLBrowser.getVar("operator");

if (operator == "+")

{

WMLBrowser.setVar("operator","-");

}

if (operator == "-")

{

WMLBrowser.setVar("operator","x");

}

if (operator == "x")

{

WMLBrowser.setVar("operator","/");

}

if (operator == "/")

{

WMLBrowser.setVar("operator","+");

}

WMLBrowser.go("calc.wml#display");

}

// Do the operation specified

extern function domath()

{

var operator = WMLBrowser.getVar("operator");

var num1 = WMLBrowser.getVar("num1");

var dec1 = WMLBrowser.getVar("dec1");

var num2 = WMLBrowser.getVar("num2");

var dec2 = WMLBrowser.getVar("dec2");

var result = 0;

// Zero values if nothing was entered

if (String.isEmpty(num1)) { num1 = 0; }

if (String.isEmpty(dec1)) { dec1 = 0; }

if (String.isEmpty(num2)) { num2 = 0; }

if (String.isEmpty(dec2)) { dec2 = 0; }

// Assemble full numbers from whole values

// and decimals

num1 = num1 + "." + dec1;

num2 = num2 + "." + dec2;

// Do calculation, depending on what operator

// was selected

if (operator == "+")

{

result = Lang.parseFloat(num1) + Lang.parseFloat(num2);

}

if (operator == "-")

{

result = num1 - num2;

}

if (operator == "x")

{

result = num1 \* num2;

}

if (operator == "/")

{

result = num1/ num2;

}

WMLBrowser.setVar("result", result);

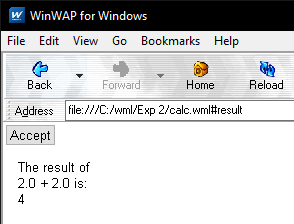
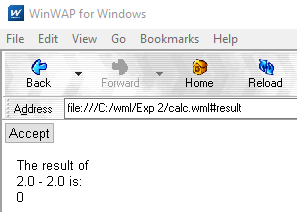
WMLBrowser.setVar("num2",num2);

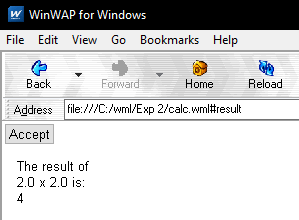
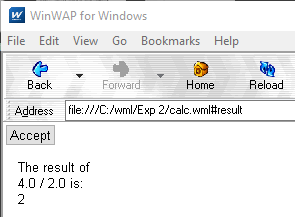
WMLBrowser.setVar("num1",num1);

WMLBrowser.go("calc.wml#result");

}

**Output**





***Experiment – 3***

***Aim :-***Write a program to design aCalendar by using WML/J2ME.

**Code**

**Calendar.wml**

<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml\_1.1.xml">

<wml>

<template>

<do type="prev"><prev/></do>

</template>

<card id ="card1" title="JAN">

<do type="unknown" label="Next"><go href="#card2"/></do>

<p>

<table columns="7" align="LCC">

<tr>

<td>su</td><td>mo</td><td>tu</td>

<td>we</td><td>th</td><td>fr</td><td>sa</td>

</tr>

<tr>

<td>1</td><td>2</td><td>3</td>

<td>4</td><td>5</td><td>6</td><td>7</td>

</tr>

<tr>

<td>8</td><td>9</td><td>10</td>

<td>11</td><td>12</td><td>13</td><td>14</td>

</tr>

<tr>

<td>15</td><td>16</td><td>17</td>

<td>18</td><td>19</td><td>20</td><td>21</td>

</tr>

<tr>

<td>22</td><td>23</td><td>24</td>

<td>25</td><td>26</td><td>27</td><td>28</td>

</tr>

<tr>

<td>29</td><td>30</td><td>31</td>

<td></td><td></td><td></td><td></td>

</tr>

</table>

</p>

</card>

<card id ="card2" title="FEB">

<do type="unknown" label="Next"><go href="#card3"/></do>

<p>

<table columns="7" align="LCC">

<tr>

<td>su</td><td>mo</td><td>tu</td><td>we</td>

<td>th</td><td>fr</td><td>sa</td>

</tr>

<tr>

<td></td><td></td><td></td><td>1</td><td>2</td><td>3</td><td>4</td>

</tr>

<tr>

<td>5</td><td>6</td><td>7</td><td>8</td>

<td>9</td><td>10</td><td>11</td>

</tr>

<tr>

<td>12</td><td>13</td><td>14</td><td>15</td>

<td>16</td><td>17</td><td>18</td>

</tr>

<tr>

<td>19</td><td>20</td><td>21</td><td>22</td>

<td>23</td><td>24</td><td>25</td>

</tr>

<tr>

<td>26</td><td>27</td><td>28</td><td></td>

<td></td><td></td><td></td>

</tr>

</table>

</p>

</card>

<card id ="card3" title="MAR">

<do type="unknown" label="Next"><go href="#card4"/></do>

<p>

<table columns="7" align="LCC">

<tr>

<td>su</td><td>mo</td><td>tu</td><td>we</td>

<td>th</td><td>fr</td><td>sa</td>

</tr>

<tr>

<td></td><td></td><td></td><td>1</td><td>2</td>

<td>3</td><td>4</td>

</tr>

<tr>

<td>5</td><td>6</td><td>7</td><td>8</td><td>9</td>

<td>10</td><td>11</td>

</tr>

<tr>

<td>12</td><td>13</td><td>14</td><td>15</td>

<td>16</td><td>17</td><td>18</td>

</tr>

<tr>

<td>19</td><td>20</td><td>21</td><td>22</td>

<td>23</td><td>24</td><td>25</td>

</tr>

<tr>

<td>26</td><td>27</td><td>28</td><td>29</td>

<td>30</td><td>31</td><td></td>

</tr>

</table>

</p>

</card>

<card id ="card4" title="APR">

<do type="unknown" label="Next"><go href="#card1"/></do>

<p>

<table columns="7" align="LCC">

<tr>

<td>su</td><td>mo</td><td>tu</td><td>we</td>

<td>th</td><td>fr</td><td>sa</td>

</tr>

<tr>

<td></td><td></td><td></td><td></td>

<td></td><td></td><td>1</td>

</tr>

<tr>

<td>2</td><td>3</td><td>4</td><td>5</td>

<td>6</td><td>7</td><td>8</td>

</tr>

<tr>

<td>9</td><td>10</td><td>11</td><td>12</td>

<td>13</td><td>14</td><td>15</td>

</tr>

<tr>

<td>16</td><td>17</td><td>18</td><td>19</td>

<td>20</td><td>21</td><td>22</td>

</tr>

<tr>

<td>23</td><td>24</td><td>25</td><td>26</td>

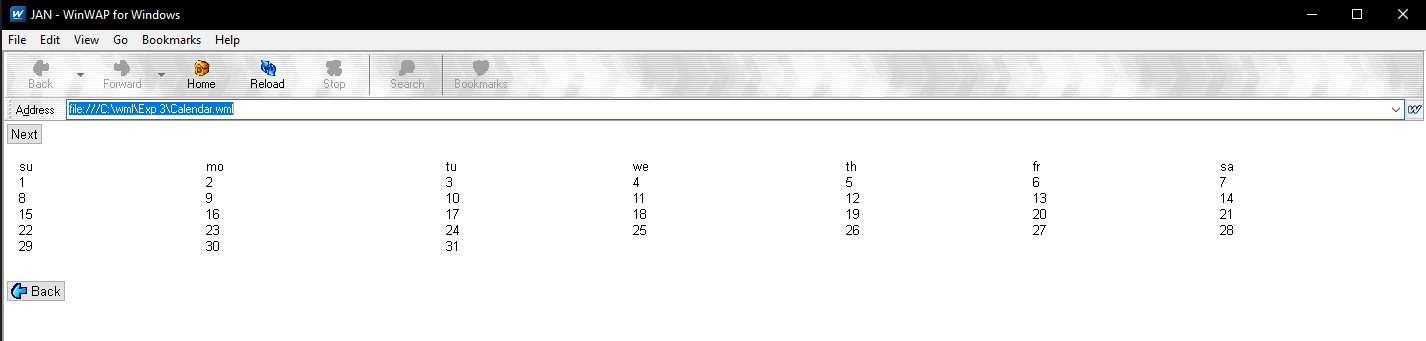
<td>27</td><td>28</td><td>29</td>

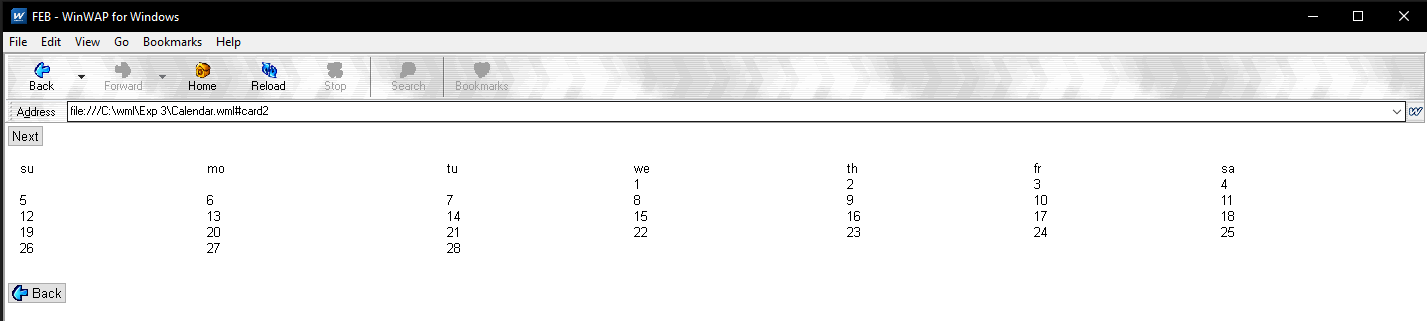
</tr></table></p>

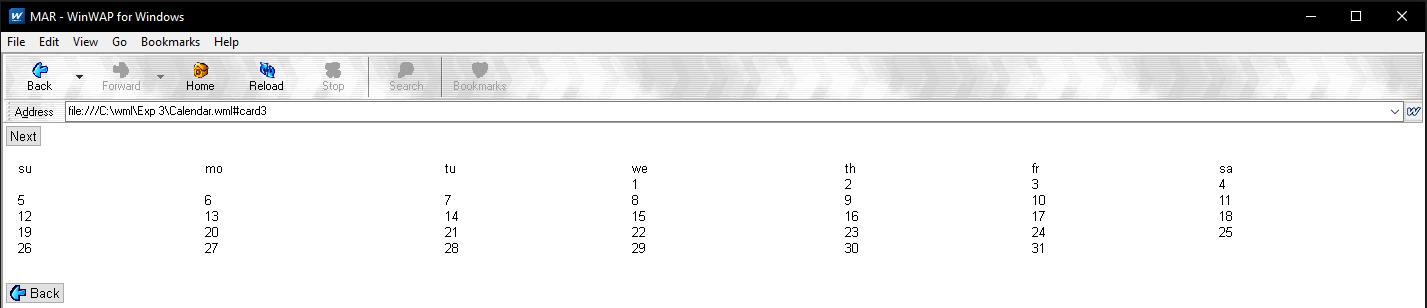
</card>

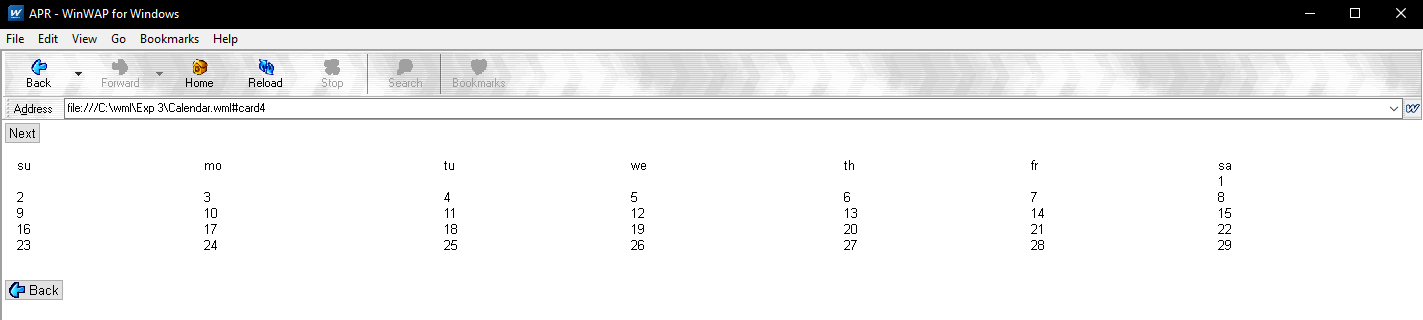
</wml>

**Output**









***Experiment – 4***

***Aim :-*** To write a program to design a timer by using WML/J2ME.

**Code**

**Timer.wml**

<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml\_1.1.xml">

<wml>

<card id="card1" ontimer="#card2" title="Toolkit Demo">

<timer value="50"/>

<p align="center">

<br/><br/><br/>

<big>

Welcome to ...

</big>

</p>

</card>

<card id="card2" ontimer="#card3" title="Toolkit Demo">

<timer value="50"/>

<p align="center">

<br/><br/>

<b>

The WinWap

</b>

...

</p>

</card>

<card id="card3" ontimer="#card1" title="Toolkit Demo">

<timer value="50"/>

<p align="center">

<br/><br/><br/>

<big>

<i>

Toolkit!

</i>

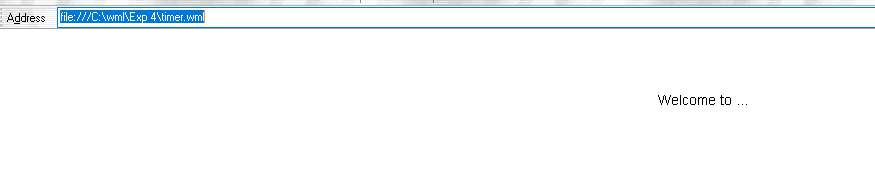
</big>

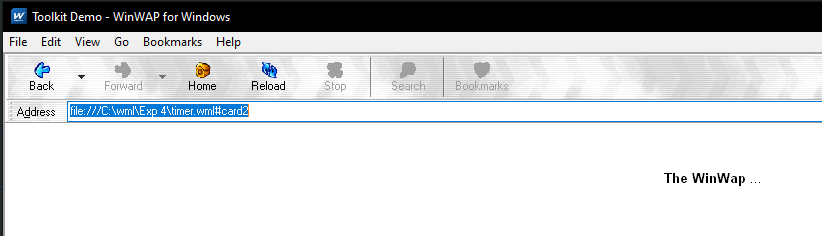
</p>

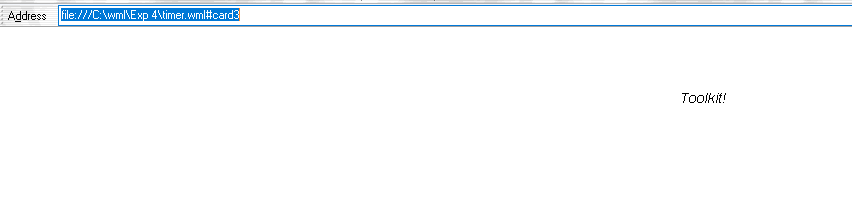
</card>

</wml>

**Output**







***Experiment – 5***

***Aim :-*** Write a program to design and implement asimple game using WML.

**Code**

**hang1.wml**

<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml\_1.1.xml">

<wml>

<card id="init" title="Hangman">

<onevent type="onenterforward">

<go href="hang1.wmls#init()"/>

</onevent>

<p>

Initializing...

</p>

</card>

<card id="status" title="Hangman">

<do type="accept" label="Guess">

<go href="#guess" />

</do>

<p>

&nbsp;|<br/>

$man<br/><br/>

$blank

</p>

</card>

<card id="guess" title="Hangman">

<do type="accept">

<go href="hang1.wmls#guess()" />

</do>

<p>

$blank<br/>

Guess: <input name="guess" maxlength="1" format="a" />

</p>

</card>

<card id="hung" title="Hangman">

<do type="accept" label="Restart">

<go href="#init" />

</do>

<p>

You've been hanged!<br/><br/>

Word was:<br/>

&nbsp;$word<br/>

Your guess:<br/>

$blank

</p>

</card>

<card id="win" title="Hangman">

<do type="accept" label="Restart">

<go href="#init" />

</do>

<p>

You win!<br/><br/>

Word was:<br/>

&nbsp;$word<br/>

</p>

</card>

</wml>

**hang1.wmls**

extern function init()

{

var words = "animal announce banana doctor elephant giraffe";

var idx,x,hang = 0;

var blank,word,man = "";

// Pick a random word from list

idx = Lang.random(6);

word = String.elementAt(words,idx," ");

// Build a blank string (letters all "\*") that

// is the same length as our word

for (x = 1; x <= String.length(word); x++ )

{

blank = blank + "\*";

}

// Pass all values to browser

WMLBrowser.setVar("word",word);

WMLBrowser.setVar("blank",blank);

WMLBrowser.setVar("hang",hang);

WMLBrowser.setVar("man",man);

// Display status card

WMLBrowser.go("hang1.wml#status");

}

// Evaluate current guess

extern function guess()

{

// Init vars

var x = 0;

var temp = "";

var correct = false;

var man = "";

// Pieces for the hanging man,comma delimited

var manpieces = " , 0,-,|,-,/, ";

// Get current values

var word = WMLBrowser.getVar("word");

var hang = WMLBrowser.getVar("hang");

var blank = WMLBrowser.getVar("blank");

var guess = WMLBrowser.getVar("guess");

// Walk one character at a time through word

// If guess = character, reveal character

// If guess != character, keep current value

// (revealed character or blank)

// Also, set "correct" if at least one char found

for (x = 0; x <= String.length(word); x++ )

{

if ( String.subString(word,x,1) == guess )

{

temp = temp + guess;

correct = true;

}

else

{

temp = temp + String.subString(blank,x,1);

}

}

// If letter wasn't found, add one to hanging counter

if (! correct) { hang++; }

// Build our hanging man

if (hang > 0) {

for (x = 1; x <= hang; x++ )

{

man = man + String.elementAt(manpieces,x,",");

}

}

// Blank the guess so <input> is blank

guess = "";

// Pass current values to browser

WMLBrowser.setVar("blank",temp);

WMLBrowser.setVar("guess",guess);

WMLBrowser.setVar("hang",hang);

WMLBrowser.setVar("man",man);

// Determine whether player has won (no more // "\*" in blank), has lost (6 pieces of man

// displayed, or keep playing (else).

if ( String.find(temp,"\*") == -1 )

{

WMLBrowser.go("hang1.wml#win");

}

else

{

if (hang >= 6)

{

WMLBrowser.go("hang1.wml#hung");

}

else

{

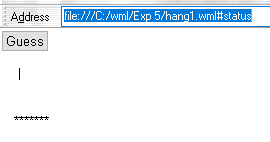
WMLBrowser.go("hang1.wml#status");

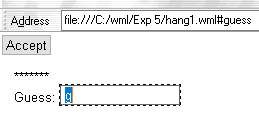
}

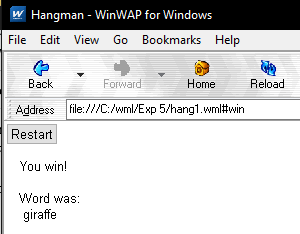
}

}

**Output**









***Experiment – 6***

***Aim :-*** Write a program to animate an image using WML.

**Code**

**Animation.wml**

<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml\_1.1.xml">

<wml>

<card id="card1" title="anime1" ontimer="#card2">

<timer value="50"/>

<br>

<img src="image1.png" alt="" width="50" height="50"/><br><br>

<do name="accept" type="accept">

<go href="#card2"/>

</do>

</card>

<card id="card2" title="anime2" ontimer="#card1">

<timer value="50"/>

<br>

<img src="image2.png" alt="" width="50" height="50"/><br><br>

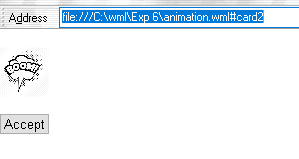
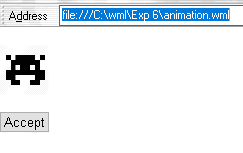
<do name="accept" type="accept">

<go href="#card2"/>

</do>

</card></wml>

**Output**



***Experiment – 7***

***Aim :-*** Write a program to design and implement a Phonebook using WML.

**Code**

**Link.wml**

<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml\_1.1.xml">

<wml>

<template>

<do type="prev"><prev/></do>

</template>

<card id="card3" title="phone book">

<do type="unknown" label="next"><go href="#card1"/></do>

<p align="left">

<small><b>Name:</b></small><br/>User3<br/>

<small><b>Phone no:</b></small><br/>

333-333-3333<br/>

<small><b>Location:</b></small><br/>

Bangalore<br/>

<small><b>E-mail:</b></small><br/>

user3@example.com

</p>

</card>

<card id="card2" title="phone book">

<do type="unknown" label="next"><go href="#card3"/></do>

<p align="left">

<small><b>Name:</b></small><br/>User2<br/>

<small><b>Phone no:</b></small><br/>

222-222-2222<br/>

<small><b>Location:</b></small><br/>Mumbai<br/>

<small><b>E-mail:</b></small><br/>user2@example.com

</p>

</card>

<card id="card1" title="phone book">

<do type="unknown" label="next"><go href="#card2"/></do>

<p align="left">

<small><b>Name:</b></small><br/>User1<br/>

<small><b>Phone no:</b></small><br/>

111-111-1111<br/>

<small><b>Location:</b></small><br/>Delhi<br/>

<small><b>E-mail:</b></small><br/>user1@example.com

</p>

</card>

</wml>

**Phonebook1.wml**

<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml\_1.1.xml">

<wml>

<card id="main" title="Directory">

<p>

<anchor>First Name

<go href="link.wml"></go>

</anchor><br/>

<anchor>phone no

<go href="link.wml"></go>

</anchor><br/>

<anchor> Location

<go href="link.wml"></go>

</anchor><br/>

<anchor> E-Mail

<go href="link.wml"></go>

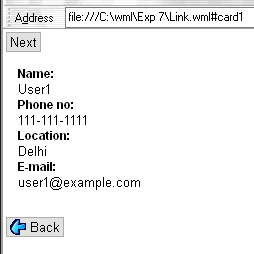
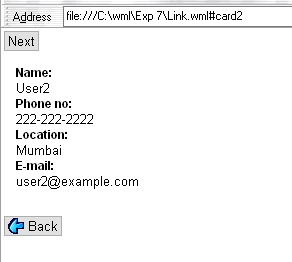
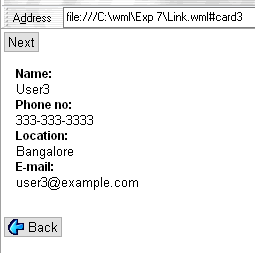
</anchor><br/>

</p>

</card>

</wml>

**Output**



***Experiment – 8***

***Aim :-*** To Study the Authentication and encryption technique used in GSM.

**Authentication** - Whenever a MS requests access to a network, the network must authenticate the MS. Authentication verifies the identity and validity of the SIM card to the network and ensures that the subscriber is authorized access to the network.

**Encryption** - In GSM, encryption refers to the process of creating authentication and ciphering crypto-variables using a special key and an encryption algorithm.

**Ciphering** - Ciphering refers to the process of changing plaintext data into encrypted data using a special key and a special encryption algorithm. Transmissions between the MS and the BTS on the Um link are enciphered.  
  
**Ki** - The Ki is the individual subscriber authentication key. It is a 128-bit number that is paired with an IMSI when the SIM card is created. The Ki is only stored on the SIM card and at the Authentication Center (AuC). The Ki will never be transmitted across the network on any link.

**RAND** - The RAND is a random 128-bit number that is generated by the AuC when the network requests to authenticate a subscriber. The RAND is used to generate the Signed Response (SRES) and Kc crypto-variables.  
  
**Signed Response** - The SRES is a 32-bit crypto-variable used in the authentication process. The MS is challenged by being given the RAND by the network, the SRES is the expected correct response. The MS receives the RAND as a challenge and uses it to calculate the SRES. The SRES is passed up to the network to as a response to the challenge.

**A3 Algorithm** - The A3 algorithm computes a 32-bit Signed Response (SRES). The Ki and RAND are inputted into the A3 algorithm and the result is the 32-bit SRES. The A3 algorithm resides on the SIM card and at the AuC.

**A8 Algorithm** - The A8 algorithm computes a 64-bit ciphering key (Kc). The Ki and the RAND are inputted into the A8 algorithm and the result is the 64-bit Kc. The A8 algorithm resides on the ISM card and at the AuC.

**COMP128** - A keyed hash function that combines the A3 and A8 algorithms into a single function. The 128-bit Ki and 128-bit RAND are input into the COMP128 which generates a 32-bit SRES and a 54-bit Kc in a single function. COMP128 is weak because it can give away information about the Ki.   
  
**Kc** - The Kc is the 64-bit ciphering key that is used in the A5 encryption algorithm to encipher and decipher the data that is being transmitted on the Um interface.  
  
**A5** - The A5 encryption algorithm is used to encipher and decipher the data that is being transmitted on the Um interface. The Kc and the plaintext data are inputted into the A5 algorithm and the output is enciphered data. The A5 algorithm is a function of the Mobile Equipment (ME) and not a function of the SIM card. The BTS also makes use of the A5 algorithm.  
  
There are three versions of the A5 algorithm:  
  
     A5/1 - The current standard for U.S. and European networks. A5/1 is a stream cipher.  
  
     A5/2 - The deliberately weakened version of A5/1 that is intended for export to non-western countries. A5/2 is a stream cipher.  
  
     A5/3 - A newly developed algorithm not yet in full use. A5/3 is a block cipher.  
  
**Triplets** - The RAND, SRES, and Kc together are known as the Triplets. The AuC will send these three crypto-variables to the requesting MSC/VLR so it can authenticate and encipher.