**1.INTRODUCTION**

Using word processors in telugu writing is growing in a very fast pace. Currently, government ministries and departments, legal institutions, business offices, media channels, universities all use word processors in their daily work. This growth is expected to continue as computers and electronic devices become more and more prevalent in Telugu. Spelling is an important aspect of language writing. Poor spelling can interfere with communication between the writer and the reader. Word processer use spell checkers to suggest corrections to misspelled words. Unfortunately, existing word processors do not come with built-in spell checkers for every language. Individual nations create their own customized dictionaries and add them to the word processors for error correction. Currently, Telugu language lacks a reliable spell checker. This limitation need to be resolved. Overcoming the lack of spell checker problem will flourish Telugu writing and helps word spelling standardization. The objective of this project is to build a reliable and comprehensive Telugu spell checker that can be used by public in Ap.Word recognition and automatic correction techniques have been studied in a large spectrum of computer applications. These include word processors, machine translation, search engines, and voice recognition. While almost all modern human spoken language has one or more spell checkers, Telugu language lacks even a very basic one. Hence, building a Telugu spell checker will have an outstanding effect on Telugu language processing applications

1. **SOFTWARE REQUIREMENT SPECIFICATION**
   1. **INTRODUCTION:**

Developing an application for checking the spelling of a word and can know the meaning, synonyms of the word.

**2.1.1 PURPOSE OF THE SYSTEM:**

The purpose of the system is mainly to reduce the misspell words. This system provides service facility to correct misspell word and to improve the communication between the writer and reader.

**2.1.2 SCOPE OF THE SYSTEM:**

* There are Two types of users-Admin, User
* There are mainly three types of Modules

Auto spell checking :

Module which lets the user to check the spelling of the word.

Meaning :

Module which will let the user to know the meaning of the word.

Synonyms:

Module which allows the user to know the different meanings of the word.

* The user can search any spelling of the word based on his interest.
* The Admin will need to update the dictionary and keep it up to date.

**2.2 REQUIREMENTS**

**2.2.1 FUNCTIONAL REQUIREMENTS:**

Functional requirements are the requirements which deals with the operational requirements of the system and the requirements that are requested by the user.

1. Admin manages the entire application.
2. Admin prevents unauthorised access to the system and keeps the list up to date by inserting new meanings,words,synonyms.
3. User needs to register himself to use the system.
4. If the user is a registered user, then he needs to authenticate himself against the system.
5. User needs to check the spellings.
6. If the user entered word is correct, then we can find the meaning of that word
7. If the user entered word is correct, then we can find the synonyms of that word

**2.2.2 NONFUNCTIONAL REQUIREMENTS:**

Non functional requirements describe user visible aspects of the system that are not directly related with the functional behaviour of the system. Non functional requirements include quantitative constraints such as response time or accuracy.

**2.2.2.1 Hardware considerations:**

* Processor: : Intel dual core processor
* RAM : 1GB
* Disk Space : 32GB

**2.2.2.2 Software considerations**:

* Programming language : Python
* UML : Rational rose
* Operating system :Windows XP, Windows 7,Windows 8

.

**3. UML DESIGN**

**3.1 UML:**

Unified Modelling Language is a general purpose modelling language. The main aim of UML is to define a standard way to visualise the way a system has been designed. It is quite similar to blueprints used in other fields of engineering. It is not a programming language, It is rather a visual language.

**3.2USECASE DIAGRAM:**

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

**3.2.1 Identification of Actors:**

Actors represent system users. They help delimit the system and givea clearer picture of what the system should do. It is important to notethat an actor interacts with, but has no control over the use cases.

An actor is someone or something that:

• Interacts with or uses the system.

• Provides input to and receives information from the system.

• Is external to the system and has no control over the use cases.

**3.2.2Identification of Use cases:**

In its simplest form, a use case can be described as a specific way of using the system from a user’s (actors) perspective.

A more detailed description might characterize a use case as:

• A pattern of behaviour the system exhibits.

• A sequence of related transactions performed by an actor in the system.

• Delivering something of value to the actor.

Use cases provide a means to:

• Capture system requirements.

• Communicate with the end users and domain experts.

• Test the system.

Use cases are the best discovered by examining the actors and defining what the actor will be do with the system. Since all the needs of a system typically cannot be covered in one use case, it is usual to have a collection of use case. Together this use case collection specifies all the ways of using the system.

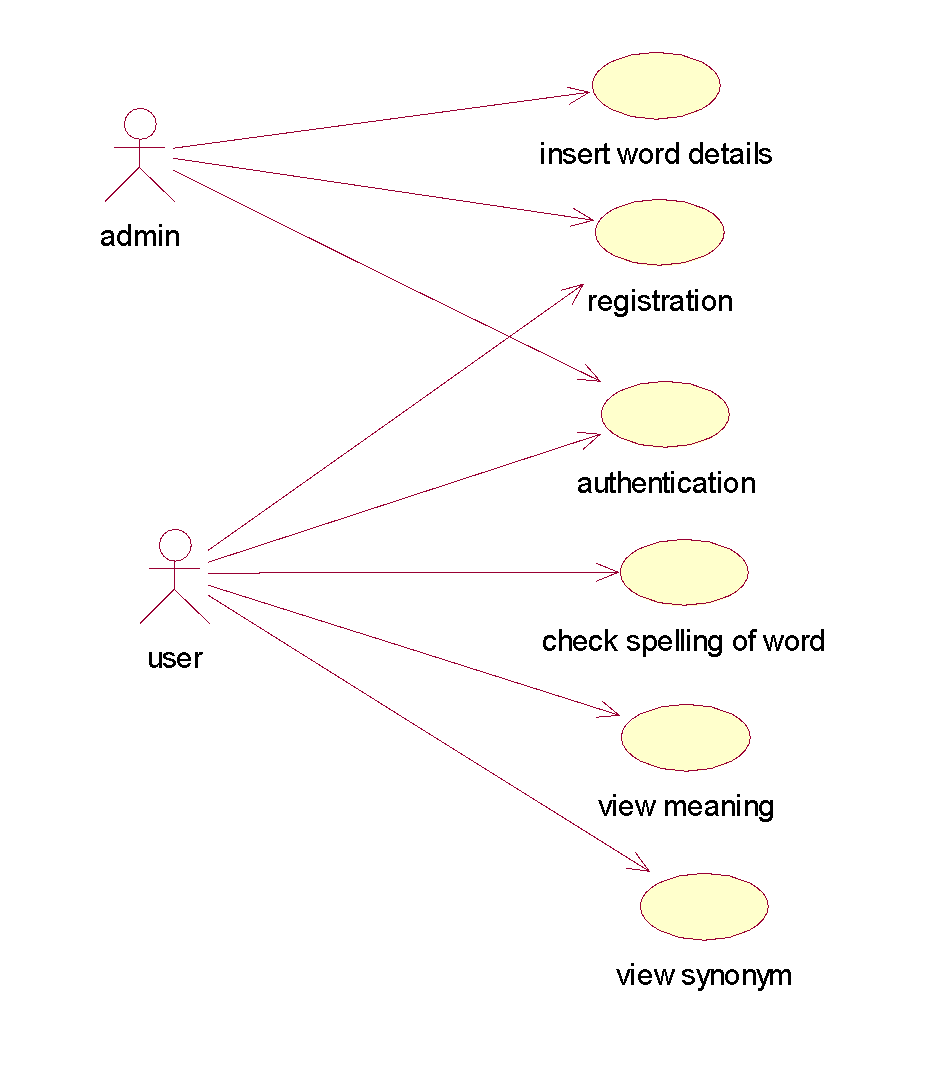


Figure:3.1

**USE CASE: Insert Word details**

**Actor Actions**  **System Responds**

1 .Collect data

2. Click ‘Insert word details’ option

3. Dialog box appears

4. Enter word related details

5. Enter ‘Submit’

6. Dialog box disappears

**USE CASE: Registration**

**Actor Actions**  **System Responds**

1. Click on ‘New user’ option

2. Dialog box appears

3. Enter user details

4. Click on ‘Submit’ button 5. Dialog box disappears

**USE CASE: Authentication**

**Actor Actions**  **System Responds**

1. Click on ‘Login’ button

2. Dialog box appears

3. Enter username and password

4. Click on ‘Submit’ button

5. Dialog box disappears

**USE CASE: check the spelling of the word**

**Actor Actions System Responds**

1 .Click on ‘check spelling ’ button

2. Dialog box appears

3. Enter a word name

4. Click on ‘check’ button

5. Dialog box disappears

**USE CASE: VIEW MEANING**

**ACTOR ACTIONS SYSTEM RESPONDS**

1. After checking the spelling for a word,

click on ‘Meaning’ button

2. System displays the meaning of the word

**USE CASE: VIEW SYNONYM**

**ACTOR ACTIONS SYSTEM RESPONDS**

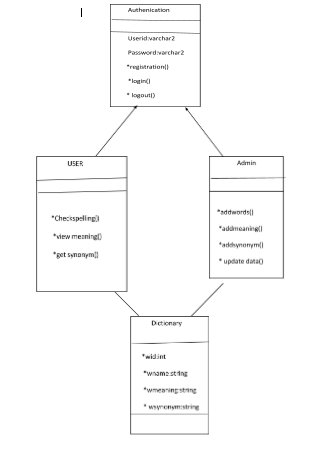
1. After checking the spelling for a word,

click on ‘Synonym’ button

2. System displays the synonym of the word

**3.3 CLASS DIAGRAM:**

Classes diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object oriented languages



**Figure:3.2**

**3.4 SEQUENCE DIAGRAM:**

Sequence diagrams simply depicts the interaction between objects in a sequential order. They capture the interaction between objects in the context of collaboration. Sequence diagrams are time focus and they show the order of interaction visually by using the vertical axis of the diagram.

**SEQUENCE DIAGRAM:REGISTRATION**

Authenication

Userid:varchar2

Password:varchar2

\*registration()

\*login()

\* logout()

USER

\*Checkspelling()

\*view meaning()

\*get synonym()

Admin

\*addwords()

\*addmeaning()

\*addsynonym()

\* update data()

Dictionary

\*wid:int

\*wname:string

\*wmeaning:string

\* wsynonym:string

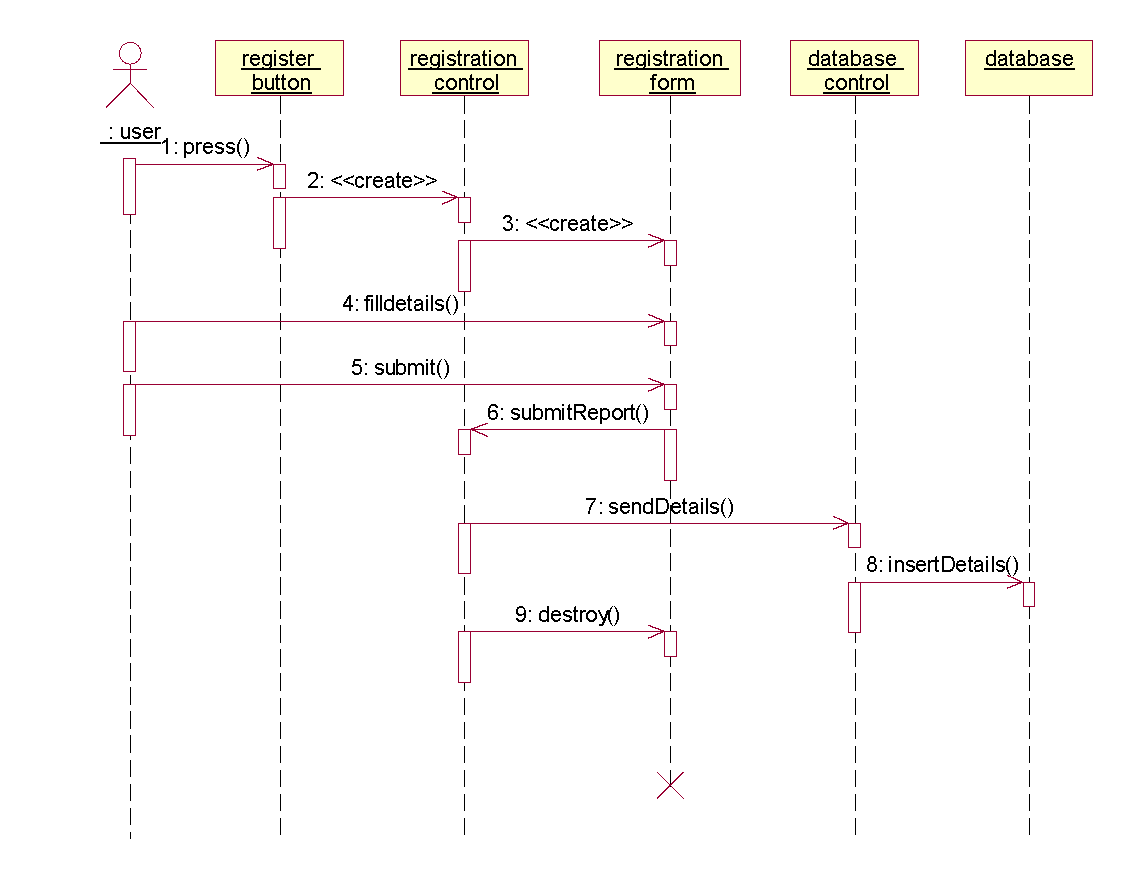


Figure:3.3

**SEQUENCE DIAGRAM:****Authentication**



Figure:3.4

**SEQUENCE DIAGRAM:****CHECK SPELL OF WORD**

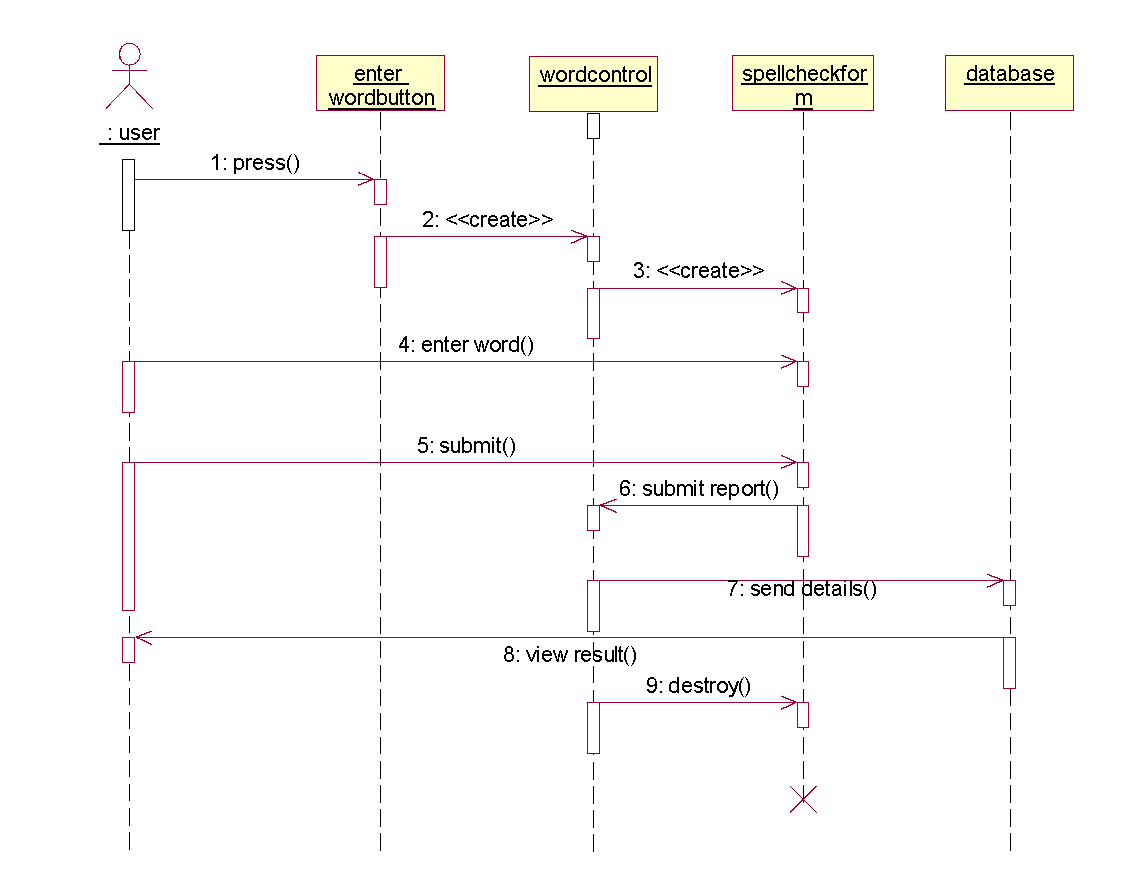


Figure:3.5

**SEQUENCE DIAGRAM:INSERT WORD DETAILS** 

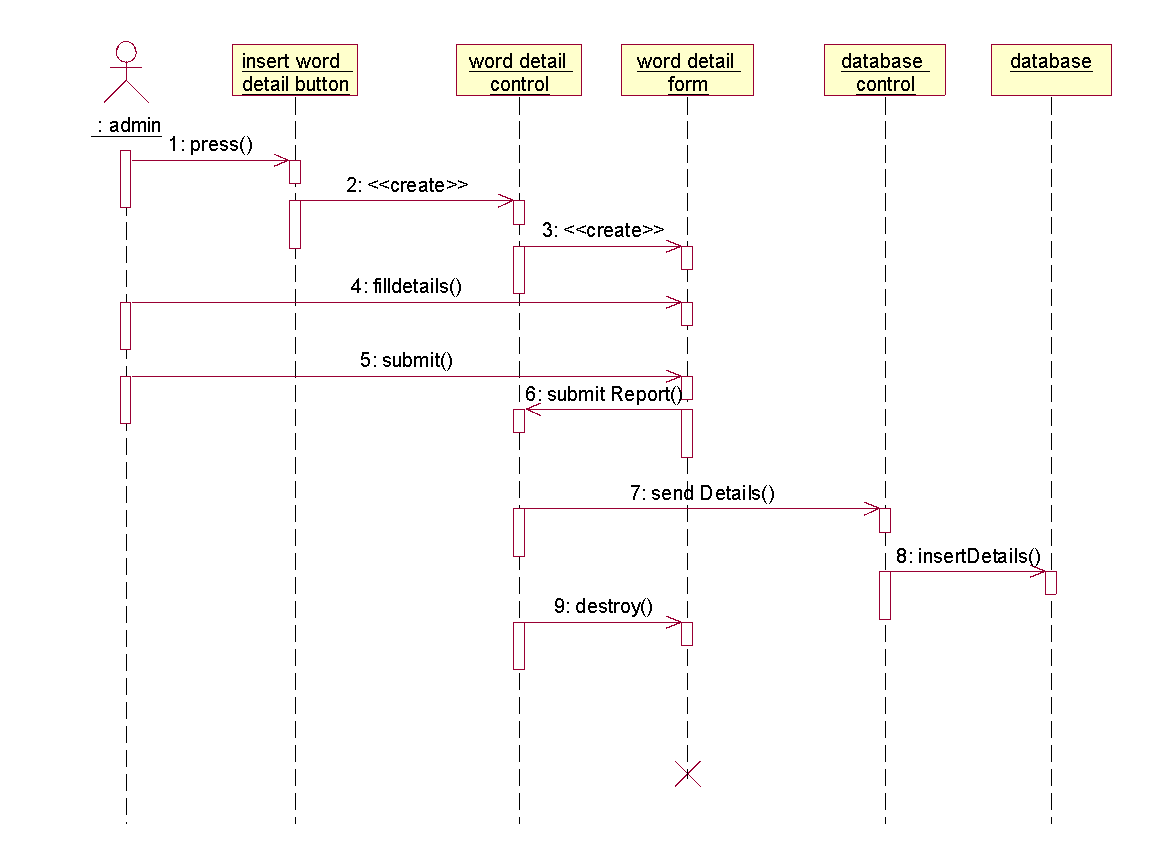


Figure:3.6

**SEQUENCE DIAGRAM:****VIEW MEANING**

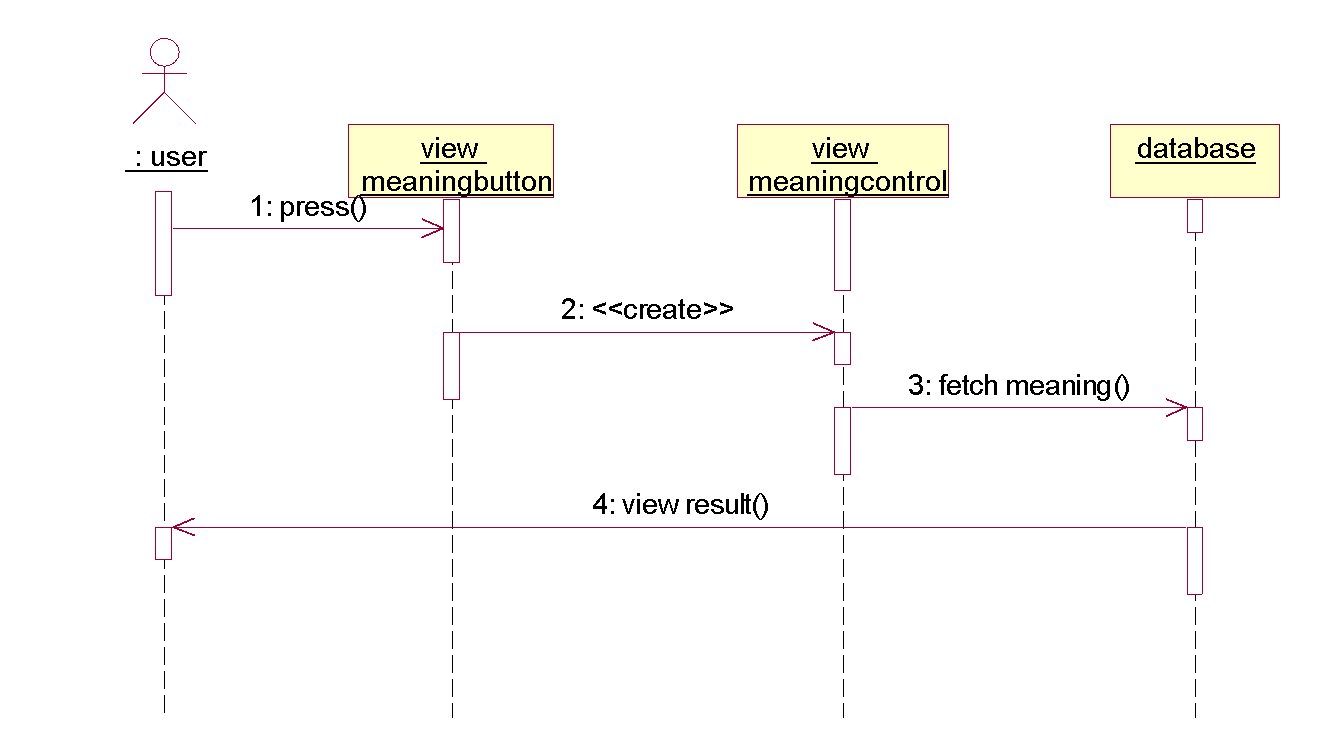
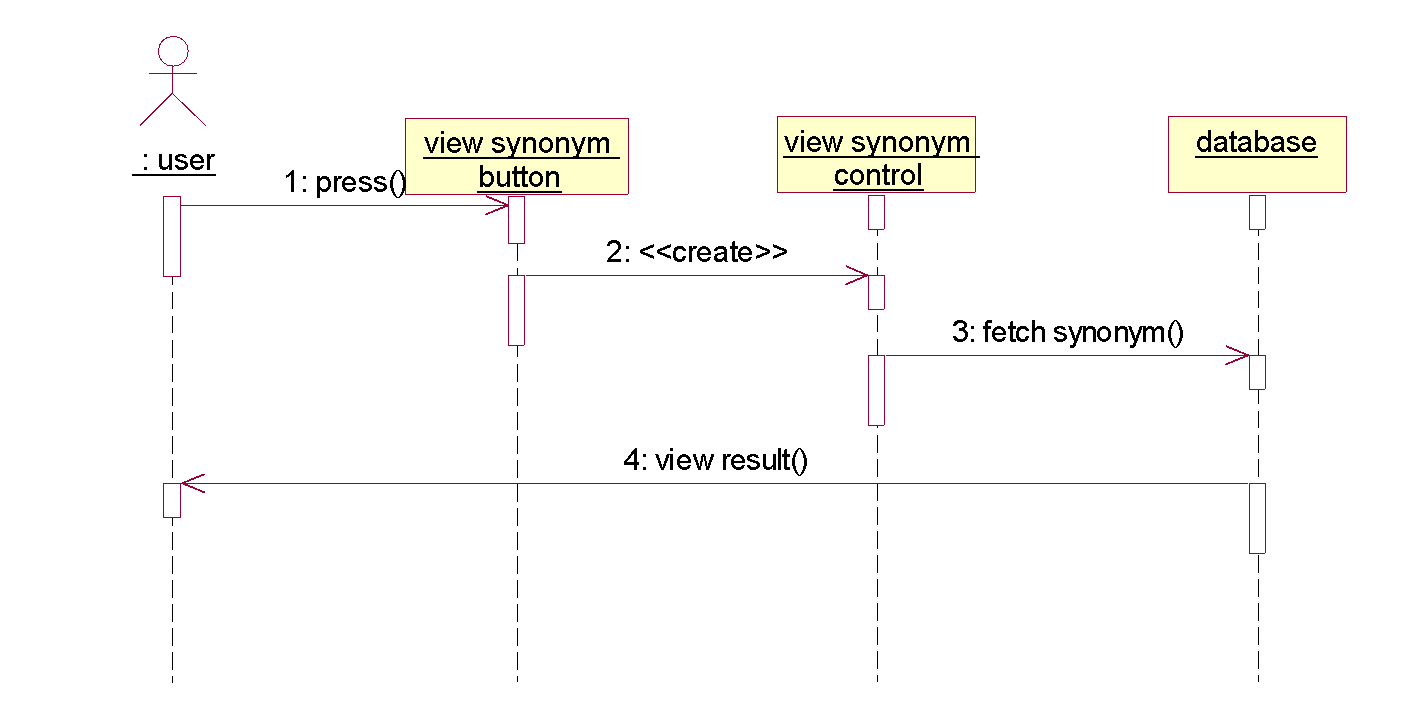


Figure:3.7

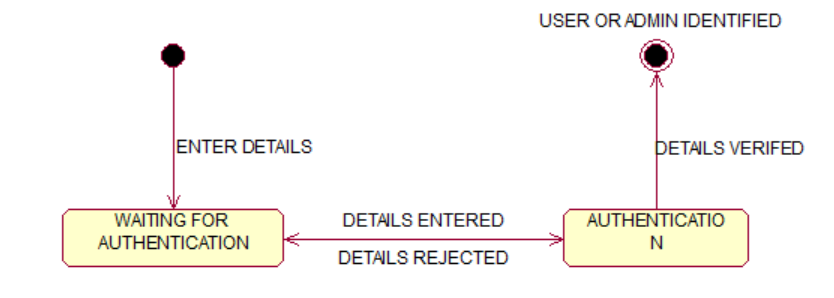
**SEQUENCE DIAGRAM:****VIEW SYNONYM**

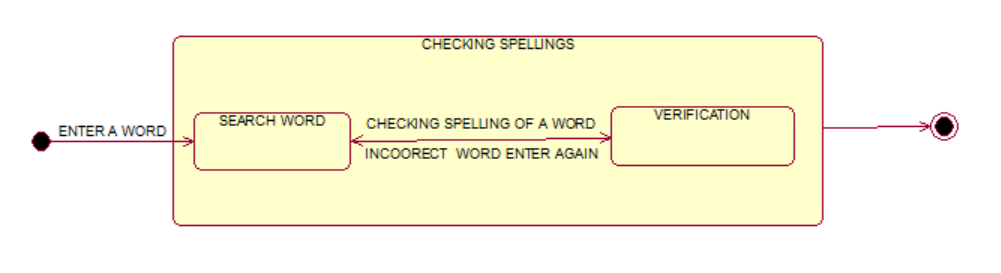


**Figure:3.8**

**3.5 STATE-CHART DIAGRAMS:**

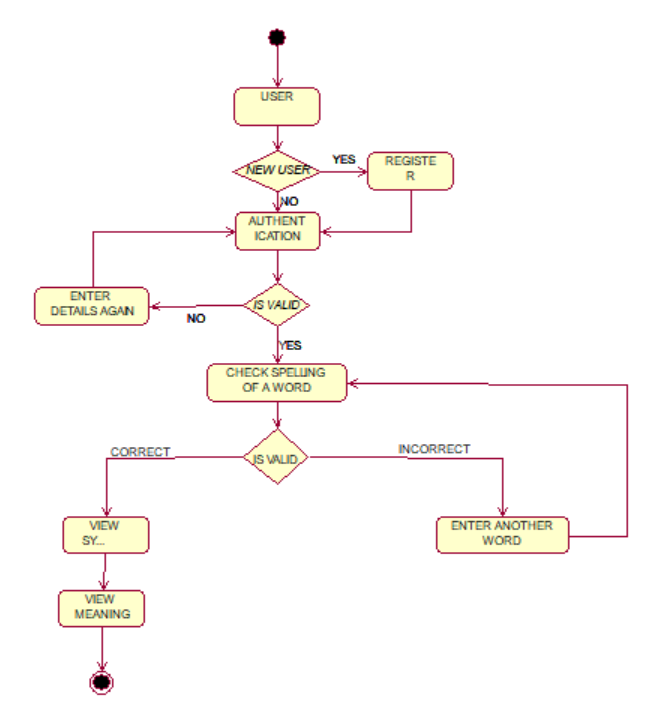
A state diagram is another way of expressing dynamic information about a system, it is used to describe the externally visible behaviour of a system or of an individual object.

****

****

**Figure:3.9**

**3.6 ACTIVITY DIAGRAM:** An activity diagram is like a state diagram,except that it has a few additional symbols and is used in a different context.In a state diagram,most transitions are caused by external events,however,in an activity diagram,most transitions are caused by internal events.

****

**Figure:3.10**

**4. SYSTEM DESIGN DOCUMENT**

**4.1 DATABASE DESIGN:**

Database design is the process of producing a detailed model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. It can be thought of as the logical design of the data structures used to store the data.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data type** | **Constraints** |
| Name | Varchar2(20) | not null |
| Admin\_id | Varchar2(20) | primary key |
| Password | Varchar2(20) | not null |
| Phno | Number(10) | unique key |
| Reference\_id | Varchar2(20) | unique key |

Table:4.1

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data type** | **Constraints** |
| Name | Varchar2(20) | not null |
| User\_id | Varchar2(20) | primary key |
| Password | Varchar2(20) | not null |
| Phno | Number(10) | unique key |

Table: 4.2

**5.IMPLEMENTING SCREENS**

The following are various screenshots of this project.

1.ACTOR ROLES SCREEN:

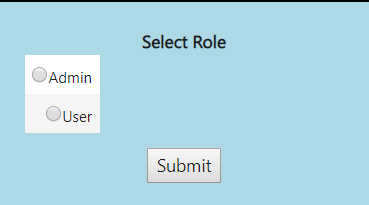


Figure:5.1

2.ADMIN REGISTRATION SCREEN:

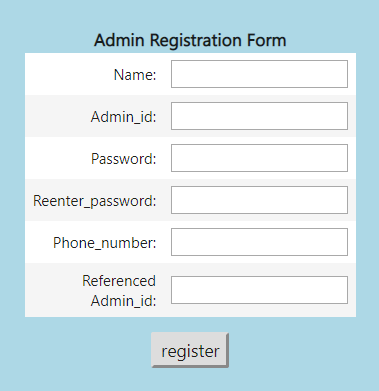


Figure:5.2

3.USER REGISTRATION SCREEN:

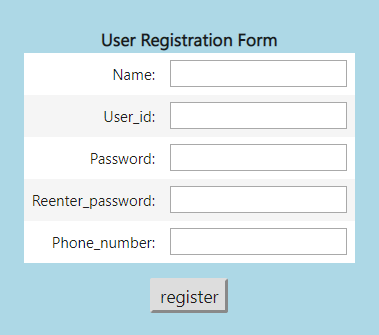


Figure:5.3

4.ADMIN INSERT DETAIL WORD FORM SCREEN:

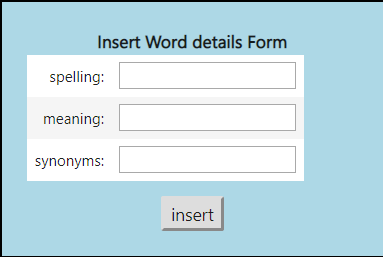


Figure:5.4

5.USER CHECK THE SPELLING SCREEN:

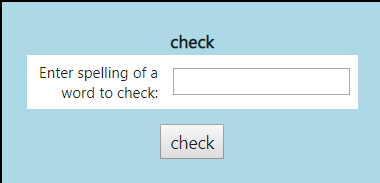


Figure:5.5

6.USER VIEW THE MEANING OR SYNONYMS SCREEN:

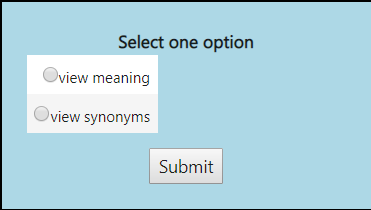


Figure:5.6

**6. TESTING**

Testing is defined as an activity to check whether the actual results match the expected results and to ensure that the system is defect free. It involves execution of components or system to evaluate one or more properties of interest.

**6.1 TESTING ACTIVITIES:**

There are four types of testing activities. They are Inspecting components, Unit testing, Integration testing, System testing.

**6.1.1 INSPECTING COMPONENTS:**

Inspections find faults in a component by reviewing its source code in a formal meeting. Inspections can be conducted by a team of developers, including the author of the component, a moderator, who facilitates the process and one or more reviews who find faults in the component.

**6.1.2 UNIT TESTING:**

It focuses on the building blocks of the software systems like objects and subsystems. The following are various types of unit testing activities

**6.1.2.1 Equivalence Testing:**

It is a black box testing technique that minimises the number of test cases. The possible inputs are partitioned into equivalence classes and a test case is selected from each class.

**6.1.2.2 Boundary Testing:**

It is a special case of equivalence testing and focuses on conditions at the boundary of the equivalence classes.

**6.1.2.3 Path Testing:**

It is a white box testing that identifies faults in the implementation of the component. The assumption behind this is that by exercising all possible paths through the code atleast once.

**6.1.2.4 State based testing:**

It is based on object orientation. It focuses on object oriented systems

**6.1.3 INTEGRATION TESTING:**

It detects faults that have not been detected during unit testing by focussing on small groups of components. The following are various types of integration testing

**6.1.3.1 Big bang Testing:**

It assumes that all components are first tested individually and then tested together as a single system

**6.1.3.2 Bottom up Testing:**

It individually tests each component of bottom layer and then integrates them with the components of next higher layer. Test drivers are used to simulate the components of higher layers. No test stubs are used.

**6.1.3.3 Top down Testing:**

It individually tests each component of top layer and then integrates them with the components of next lower layer. Test stubs are used to simulate the components of higher layers. No test drivers are used.

**6.1.3.4 Sandwich Testing:**

It combines top down and bottom up testing strategies, attempting to make use of the best of both strategies. Here in this strategy no test stubs and test drivers are used.

**6.1.4 SYSTEM TESTING:**

The following are various system testing activities

**6.1.4.1 Functional testing:**

It is also called requirements testing, finds differences between the functional requirements and the system. It is a black box technique.

**6.1.4.2 Performance Testing:**

It finds the differences between the design goals selected during system design and the system. The following tests are performed during performance testing

* Stress Testing: It checks if the system can respond to many simultaneous requests.
* Volume Testing: It attempts to find the faults associated with large amounts of data, such as static limits imposed by data structure, high complexity algorithms, high disk fragmentation
* Security Testing: It attempts to find the security flaws. Usually this test is accomplished by tiger teams.
* Timing Testing: It attempts to find behaviours that violates timing constraints described by the non functional requirements.
* Recovery Testing: It evaluates the ability of the system to recover from errors such as the unavailability of resources, hardware failures.

**6.1.4.3 Pilot Testing:**

During this, system is installed and used by a selected set of users. No explicit guidelines are given to the users. This is also called Field testing.

**6.1.4.4 Acceptance Testing:**

It is done by the client and checks whether he can accept the system or not. It consists of bench mark test, competitor test, shadow test.

**6.1.4.5 Installation Testing:**

After the system is accepted, it is installed in the target environment. The desired outcome of the installation test is that the installed system correctly address all the requirements.

**6.2 TEST CASE**

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is an indispensable component of the software testing lifecycle that helps us to validate a particular functionality.

For the convenience purpose, only Unit testing is done on this project.

**6.2.1 TEST CASE SCENARIOS:**

A test case has 6 major components. They are Test steps, Test Input, Oracle, Log, Remarks

The following are the various test case scenarios

**1. Scenario for Login functionality**

Here we test the Login component for a valid user or an invalid user by taking his username and password as test input data.

The following are various test cases for the login functionality

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Test steps | Test Input | Oracle | Log | Remarks |
| 1 | Select Log in option |  | User should be able to login | User is taken to his profile | PASS |
| 2 | Enter Username | User:  Manikanta |  |  |  |
| 3 | Enter Password | Password:  MKALSLRD |  |  |  |
| 4 | Click Enter |  |  |  |  |

Table: 6.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Test steps | Test Input | Oracle | Log | Remarks |
| 1 | Select Log in option |  | User should be able to login | User is taken to his profile | PASS |
| 2 | Enter Username | User:  Pawankalyan |  |  |  |
| 3 | Enter Password | Password:  pspk@123 |  |  |  |
| 4 | Click Enter |  |  |  |  |

Table: 6.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Test steps | Test Input | Oracle | Log | Remarks |
| 1 | Select Log in option |  | User should not able to login | Incorrect Username or password | PASS |
| 2 | Enter Username | User:  Chaitanya |  |  |  |
| 3 | Enter Password | Password:  Gaali |  |  |  |
| 4 | Click Enter |  |  |  |  |

Table: 6.3

**2. Scenario for check functionality:**

Here we test thecheck component for a valid word name.

The following are various test cases for the check functionality

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Test steps | Test Input | list | Log | Remarks |
| 1 | Select ‘check spelling for a word’ option |  | Valid word name |  |  |
| 3 | Enter a word name | Word name:'నవ్వడం |  |  |  |
| 3 | Click on enter |  |  | User will see the spelling of that word | PASS |

Table:6.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Test steps | Test Input | list | Log | Remarks |
| 1 | Select ‘Check spelling for a word’ option |  | Valid word name |  |  |
| 2 | Enter a word name | word name:'నవ్వడం |  |  |  |
| 3 | Click on enter |  |  | User will see the spelling of that word. | PASS |

Table: 6.5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Test steps | Test Input | list | Log | Remarks |
| 1 | Select ‘Check spelling for a word’ option |  | valid stock name |  |  |
| 2 | Enter a stock name | Word name:'నవ్వడం |  |  |  |
| 3 | Click on enter |  |  | User will see the spelling of that word. | FAIL |

Table: 6.6

**7. CONCLUSION**

The purpose of this project is mainly reduce the effort of finding the spelling of a word.As Poor spelling can interfere with communication between the writer and the reader. Word processer use spell checkers to suggest corrections to misspelled words.

**8. REFERENCES**

* Object Oriented Software Engineering by Allen H. Dutoit and Bernd Bruegge.
* [www.google.com](file://C:\Users\user\Desktop\OOSE\www.google.com)

**APPENDIX**

**Sample Implementation code:**

from nltk.corpus import indian

s=set()

dic={}

dic1={}

l1=[]

l2=[]

l3=[]

l4=[]

lit=[]

q1=[]

dic4={}

s1=set()

#l10=list(indian.words('telugu.pos'))

l10=['ఆడిట్', 'నిర్వహణ', 'ఆడిటర్', 'ఒక', 'కొత్త', 'చేపట్టే', 'ముందు', 'సక్రమ', 'పద్ధతి', 'కార్య', 'ప్రణాళికను',

'పిలిపించి ', 'మాట్లాడించారు', 'విని', 'పరిహసించినట్లు', 'నవ్వడం', 'ఊరుకున్నారు', 'ఒకరు', 'ఇద్దరు', 'భాషలు',

'పరిహసిస్తూ', 'మనస్సులో', 'వినపడేట్టు', 'అందరికీ', 'తెలియకపోవడం', 'చెయ్యవలెనో', 'మృగంలో', 'విశేషం', 'కనిపించింది',

'తొణకదు', 'బెణకదు', 'చుట్టుపట్ల', 'నిరాదరణగా', 'చూస్తుంది', 'ఇంతసేపూ', 'పెట్టుకొని', 'ఇప్పుడు', 'తుపాకిమీద',

'ఎందుచేత','పంజరాన్ని', 'అంటుకొని', 'నిలుచున్నది', 'సైనికుడు', 'భయపడి', 'వెనుకకు', 'తుపాకి', 'ఎత్తి',

'పట్టుకొన్నాడు','తిరగడం', 'తటస్థించింది', 'మారణ', 'సాధనం', 'తెలిసికొంది', 'దానివంక', 'సాభిప్రాయంగా', 'చేతికి',

'ఇమ్మని', 'అడిగింది','ఇవ్వలేదు', 'ఆశ్చర్యపోయి', 'చూశారు', 'క్రుంకింది', 'దారిన', 'వెళ్లిపోతున్నారు','అనవసరం',

'కొంతమంది', 'పనీ','ఎక్కువని', 'భావించవచ్చు', 'లోపాలను', 'నివారించటానికి', 'కార్యక్రమాన్ని', 'మార్గదర్శిగా',

'ఎంచవలెనేగాని''కారణ', 'మన్న', 'సంగతి', 'ఎవరికీ', 'ప్రపంచానిక', 'మేలు', 'చేయడాని', 'కని', 'మహానీయులు', 'పుడుతూ',

'జన్మించిన', 'మహా', 'పురుషులలో', 'మహాశయుడు', 'ఎట్లు', 'సర్దుకుపోవలె', 'అంశాన్ని', 'పరిశీలిస్తుంది', 'రాజకీయాలకు',

'మనిషే', 'మూలము', 'వ్యక్తికి', 'హక్కులను', 'స్వేచ్ఛలను', 'యిచ్చి', 'చూడటం', 'సంభనించిందన్నారు', 'పోలీసులంటే',

'భయం', 'సాధ్యమైనంత', 'పోలీసులకు', 'దూరంగా', 'జరిగిందన్నారు', 'విచారణ', 'గ్రామానికి', 'కాలంపాటు', 'తిష్ఠ', 'వేయగా',

'గ్రామస్తులు', 'అవసరాలు', 'రాజ్యం', 'విధులను', 'నిర్వహిస్తుంది', 'పరస్పర', 'పోషకాలు', 'లేనిదే', 'ఉనికి',

'సాధ్యంకాదు', 'కారణం', 'చేతనే', 'లాస్కీ', 'మొదలగు', 'చెప్పినారు', 'సిద్ధాంత', 'రాజనీతిశాస్త్రంలో']

for i in l10:

if i not in lit:

lit.append(i)

for i in lit:

if i not in [".", '4', '14', '"', "(",")", "!", "Z", "`", "'", ";"]:

for x in list(i):

if x in s:

if i not in dic[x]:

dic[x].append(i)

else:

s.add(x)

dic[x]=[]

dic[x].append(i)

else:

pass

def spellcheck(query):

q=list(query)

dup=list(q)

for i in q:

if i not in q1:

q1.append(i)

else:

pass

for i in q1:

l1 = list(dic[i])

for j in l1:

if j in s1:

dic1[j]+=1

else:

s1.add(j)

dic1[j]=1

l1.clear()

print(dic1)

max\_freq=max(dic1.values())

for key,value in dic1.items():

if value==max\_freq:

l2.append(key)

print(max\_freq)

print(l2)

for word in l2:

ed=0

q=list(dup)

l3=list(word)

x=y=-1

for i in range(len(l3)):

for j in range(len(q)):

if (l3[i] == q[j]) and (j != y):

x = y

y = j

l = 1

if i == (len(l3) - 1):

q[y] = "\*"

else:

pass

break

elif (j == (len(q) - 1)):

ed = ed + 1

l = 0

else:

pass

if l != 0:

if x < y:

if (x != -1):

q[x] = "\*"

l = 1

else:

pass

else:

ed = ed + 1

else:

pass

if l == 0:

q[y] = "\*"

else:

pass

for j in q:

if j != "\*":

ed += 1

dic4[word]=ed

l3.clear()

print(dic4)

min\_ed = min(dic4.values())

for key, value in dic4.items():

if value == min\_ed:

print(key)

query=input("enter the mis spelt word\t")

spellcheck(query)