Chapter 1
INTRODUCTION

1.1. MOTIVATION

A MIDlet is an application that uses the Mobile Information Device Profile(MIDP) of the Connected Limited Device Connection (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 HTTP. Android users can still run midlets with J2ME Loader emulator. The .jad file describing a MIDlet suite is used to deploy the applications in one of two ways. Over the air (OTA) deployment involves uploading the .jad and .jar files to a Web server which is accessible by the device over HTTP. The user downloads the .jad file and installs the MIDlets they require.

Local deployment requires that the MIDlet files be transferred to the device over a non-network connection (such as through Bluetooth or IrDA, and may involve device-specific software). Bot is a computer programs that performs automatic repetative tasks.

They save time for the server and the customer or client and are also very convenient. They save time and money for the business and this results in higher profits for the business.24-hour availability.

Should you get home in the dead of the night and you are either too tired to cook or there is nothing in the fridge, you can turn to the internet for a solution. They cannot be effectively set to perform exact tasks and they can misunderstand instructions. If a customer makes an order that is beyond the bot's scope of knowledge, that could be a problem, especially if there is no human to interpret the order or deal with such a situation should it crop up.

1.2 EXISTING SYSTEM

- T-Rex Runner Game is developed using web crawler, and JavaScript.
- Talking about the gameplay, it is an offline mode game which is played by almost every people on Google Chrome browser.

- The gameplay, as well as controls, are the same. Talking about the Game environment, the graphics are simple with high-quality sprite images. For the development of this gaming project, various Images, sounds, scripts are used, it is developed using JavaScript to bring the final output.

- All the gaming function is set from Java script whereas HTML is set for the layouts and other minor functions.

- To run this project, we recommend you to use Modern browsers such as Google chrome.

1.3 PROBLEM STATEMENT

We want our website to be bug-free, reliable and flexible user interaction. Today many people are facing trouble to obtain various Post Office services like Courier, Post Cards, Stamps etc because of the formation of long queues and lack of time.

We have too many release failures that result in too many rollback failures. If we ignore this problem; resources will need to increase to handle the cascading problems, and we may miss critical customer deadlines which could result in lost revenue, SLA penalties, lost business, and further damage to our quality reputation. (Issue Statement)

We will use our Kaizen Blitz methodology in evaluating the last release to help us improve our processes. (Method)

1.4 PROPOSED SYSTEM

- The T-Rex runner game is developed by python programming language and java.
- In this we are implementing the off-line and online mode. The gameplay design is so simple that user won't find it difficult to use and understand.
- Different images are used in the development of this simple game project, the gaming environment is just like the original T-Rex Runner game.
- We can run this game by using google chrome.

1.5 OBJECTIVE

The primary purpose of the project is to make a crawler that can provide user convenient game to play easily.

The text files are meant to be the input for the Search engine which will try to analyze the data and extract meaningful concepts from the data and store them in an Database.

The crawler will from the dinosaur to connect image processing obtained from crawling and create game. The crawler also aims to systematically store metadata in a different set of files for future use. The crawler thus aims to improve the efficiency of the Concept Based Semantic Search engine for the game play and user friendly.

Chapter 2
LITERATURE SERVEY

2.1 LITERATURE SERVEY

1. Project report on Web Crawler: Extracting the Web Data

AUTHOR: Mr. Mini Singh Ahuja and Mr. Dr Jatinder Singh Bal, Mrs.Varnica

This is a survey of the science and practice of web crawling. While at first glance web crawling may appear to be merely an application of breadth-first-search, the truth is that there are many challenges ranging from systems concerns such as managing very large data structures to theoretical questions such as how often to revisit evolving content sources. This survey outlines the fundamental challenges and describes the state-of-the-art models and solutions. It also highlights avenues for future work.

2.Project report on Efficient Focused Web Crawling Approach for Search Engine

AUTHOR: Mr. Ayar pranav and Sandip chauhan

In this paper, we present Google, most popular search engine, and in-depth description of methods and techniques that the Google uses in searching. Different search engines use different techniques for searching and algorithm to rank the pages. At the end of the paper, we compare major search engines such as Google, Yahoo, Msn etc.

3.Project report on pybot

AUTHOR: Mr. Ravi Kumar and Ashutosh Kumar Singh

PyBot is Web Crawler developed in Python to crawl the Web using Breadth First Search (BFS). The success of the World Wide Web (WWW), which itself built on the open internet, has changed the way how human share and exchange information and ideas. With the explosive growth of the dynamic Web, users have to spend much time just to retrieve a small portion of the information from the Web.

4.Project report on Web Searching and crawling

AUTHOR: Mr. Krishan Kant Lavania, Mrs. Sapna Jain

Today search engines are becoming necessity of most of the people in day to day life for navigation on internet or for finding anything. Search engine answer millions of queries

every day. Whatever comes in our mind, we just enter the keyword or combination of keywords to trigger the search and get relevant result in seconds without knowing the technology behind it. I searched for “search engine” (I intentionally mis-spelled it) and it returned 68,900 results. In addition with this, the engine returned some sponsored results across the side of the page, as well as some spelling suggestion. All in 0.36 seconds. And for popular queries the engine is even faster. For example, searches for World Cup or dance shows (both recent events) took less than .2 seconds each. To engineer a search engine is a challenging task.

4.Project report on A Survey of Web Crawler Algorithms

AUTHOR: Pavalam S M, S V Kashmir Raja, Felix K Akorli and Jawahar M

Due to availability of abundant data on web, searching has a significant impact. On-going researches place emphasis on the relevancy and robustness of the data found, as the discovered patterns proximity is far from the explored. Inspite of their relevance pages for any search topic, the results are huge to be explored. Also the users’ perspective differs from time to time from topic to topic. Usually ones’ want is others unnecessary. Crawling algorithms are thus crucial in selecting the pages that satisfies the users’ needs. This paper reviews the researches on web crawling algorithms used on searching.

Chapter 3
REQUIREMENT SPECIFICATIONS

3. SOFTWARE AND HARDWARE REQUIREMENTS

3.1. Hardware Requirements:

- Processor : Intel core dual process
- RAM : 2GB
- Hard disk : 500GB
- OS Type : 64-bit
- Graphics : Intel @Hawas Desktop

3.2. Software Requirements:

- Operating System : Ubuntu 18.0A LTS (64-bit)
- Platform : Java
- Front end : python programming
  (python IDLE 3.6/PyCharm 2019.2)
- Back end : Data Access/oracle 11g,18c

Chapter 4
DESIGN AND IMPLEMENTATION

4.1 MODULE DISCRIPTION

1 DINO

- Jump method clicks the space button and makes our dino jump. The man method controls the whole process. It makes a jump over obstacles.
- Counter the game will track your progress with a high score count, including pings of deceleration upon every 100 pint you make.
- In the game small graphics that occasionally blinks. You see the blinking and this when you wont to either tap or press space bar.
- The jumping speed will be changed when ever score is increasing 0 to 100, 200, to 300.The changes will be happened in this game. Whenever your dino hits some they its game over.
- Every module having methods these one performing actions like draw, check bounds and up. Every time checks dino is jumping correctly the bounded points or not.

2 CACTUS

- The game shows the cactus for that have methods to draw and update. In this cactus update using image grab to simultaneously comes one another.
- It will change on cactus module code in program.

3 PTERA

- This is related to the bird's wings or feature. It is also having some functions like update and draw.

4 CLOUD

- This is the storage for the score and simultaneously update the high score on the window. How many times it will play on that window only.

5 SCORE BOARD

- Score is important the change the speed of the game.

- It will change the speed of the game by depends on score. The high scare will be changed another side of the recent score.

6 GAME QUIT

- We can also quit the game and it is appearing as game over when playing the game touch, the any cactus and birds on the game.

7 LOAD IMAGE

- Load method is used to load image from resources folder. First, copy your image in the game folder then only image is uploaded in the game.

8 LOAD SPRITES

- Sprites are generally the visual representation of objects with in the game.
- As such, a sprite is either a single image, drawn with any drawing program users need, a set of images that, when played one after another, looks like a looping animation.

4.1.1 ARCHITECTURE OF PROPOSED SYSTEM

The architecture of the smart crawler as shown in the figure below consists of the following components:

1. Data Extractor
2. Extracted URL stack
3. Domain filter
4. URL classifier
5. Valid URL list
6. HTML Analyzer
7. Script Pruning
8. Text extractor
9. Metadata formatter
10. Data Files

4.1.1.1 Data Extractor

The data extractor is the component of the crawler that in the first iteration gets the URL from the user and uses the URL to access the remote server on which the URL is hosted. This module sends a http request to the remote server just like any other http request to the server. The server responds to the request by sending back the requested information. In this case the requested information is the page located in the URL.

Now it is the job of the data extractor to scan through the data and find all the URLs that are in the data. It searches the obtained data for the links to other pages and supplies them to the Initial URL stack. It is also the responsibility of the Data Extractor to supply data to the Data Analyzer for further analysis of the data. The Data Extractor runs iteratively for each URL that the Valid URL list supplies to it. The Data Extractor is the main component of the crawler.

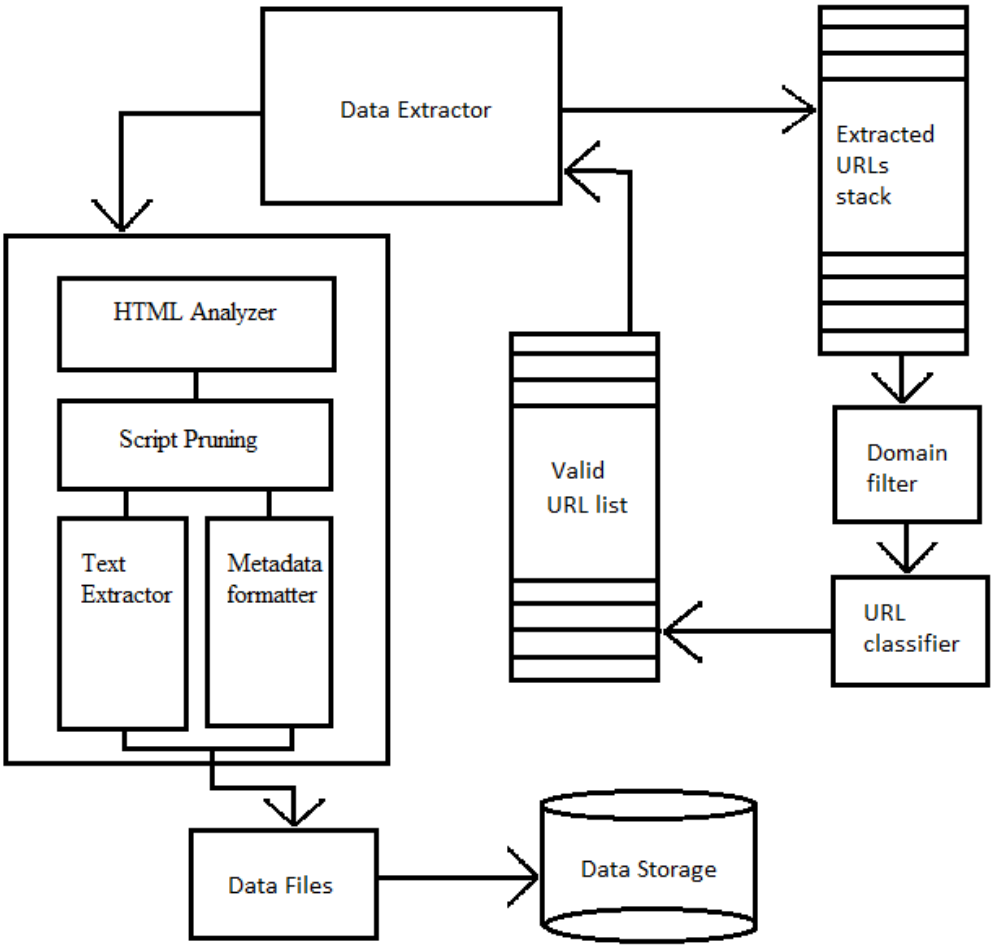

Figure 4.1.1. Architecture of crawler

4.1.1.2 Extracted URL Stack

The Extracted URL Stack stores the URLs that the Data Extractor finds on the pages that it extracts. The URL stack is actually a queue data structure. So, it has a first in first out (FIFO) mechanism.

The job of the Extracted URL stack is to supply URLs for further processing. It sends each URL if a first in first out order to the Domain Filter. So the URL Stack first stores the links from each page and forwards them for processing.

4.1.1.3 Domain Filter

Domain Filter checks whether the URL it gets from the URL stack belongs to the given domain or not. This is necessary to restrict the crawler to the specified domain. The links that the data extractor extracts will belong to servers all over the world. But if we need data of only the given domain, this Classifier filters out all the other URLs. But if we want data from all the links in the page, we can also bypass this filter.

The user will be given an option to choose if he wants to crawl the entire web or only the given domain. If he does not specify that he wants to crawl only the given domain this filter will be bypassed. After filtering out the other domain URLs the Domain Filter sends the remaining URLs to the URL Classifier for further processing. This component handles one URL at a time.

4.1.1.4 URL Classifier

The URL Classifier is the component that checks whether the URL is a useful URL or not. The URLs that the extractor extracts will contain all kinds of files like jpeg, CSS, img, is, etc. These files are useless to crawl as they will not contain any text data. They are files that are meant for styles or adding functionality to the pages. They won't have any data and need to be pruned.

This classifier checks the MIME type of the file in the URL. If the MIME type is a HTML or txt, the Classifier considers the url for next stage. If the URL belongs to any other multimedia data type, it simply filters the URL out of the stack. After this stage the URLs that have been screened and have proved to be useful will be stored in another URL holding list called Valid URL list. It is the responsibility of the URL Classifier to populate the Valid URL List.

4.1.1.5 Valid URL List

The Valid URL list contains urls that have been processed and are ready to be given to the extractor. The urls in the Valid URL List have been checked if they belong to the same domain, if they contain useful text information and if they have already been crawler. It a URL passes all the above tests it if a URL that is fit to be crawled and it is stored in this URL list.

Another feature on this list is that it avoids redundancy of URLs. It makes sure that the content in the list, in this case the URLs are unique. It is this list that supplies valid URLs to the data extractor.

4.1.1.6 Data Analyzer

The Data Analyzer is the component that is responsible for processing the data extracted by the data extractor. This Data Analyzer gets raw data from the extractor and processes it to get some meaningful text information. This has four important sub components The HTML tag Analyzer, the Java Script pruner, the Text Extractor and the Metadata Formatter. This component also writes the processed data to the .txt files on the hard disk.

4.1.1.8 HTML Tag Analyzer

An html page contains a lot of html tags which contain information that is necessary to format the html page and makes it render properly on the browser. But all those tags contain information contain tags that are unnecessary for the data analyzer. Only some tags like the paragraph tag or the header tag contain useful and meaningful information.

So, in this component we remove all that tags from the raw data that the extractor receives. At this stage most of the data is polished but still some traces of garbage data still remain and need to be removed. So this component sends the semi pruned data to the Java Script Remover for get rid of all the java script in the data.

4.1.1.9 Java Script Pruning

The Java Script that is left over in the data after all the tags are removed is filtered from the data in this component. This component scans the script elements from the data and removes them. The residual data sent to the text extractor.

4.1.1.10 Text Extractor

This component extracts important text data from the body part of the HTML page. We internally parse the data to a parse tree using the Beautiful Soup python module. From this parse tree we extract all the paragraph tags that are in the body of the HTML page.

We filter texts from all the remaining tags that contain texts. This is because we consider that the paragraph tags in the body contain text that is unique. Rest of the tags like title tag could be same throughout a given domain.

We then write this data to text files, one text file for each URL, that will be given to the Concept based engine as input.

4.1.1.11 Metadata Formatter

This component also utilizes the parse tree of the html tags and tries to convert the HTLM page into a fixed format which is a requirement of the project. It re-formats the contents of the HTML page. This component will create a separate text file for each URL. This text file will contain information about the data in the first text file and will be stored in a different directory but with the same file name. All of these files like will contain the URL of the page the crawler crawls in the first line. The second line will contain the title of the page.

This will be followed by the data from the meta tags that contain information about the data in the page. From the metadata section we use a five-star delimiter at the starting of each section. The metadata section is followed by the important keyword section followed by the image information section.

The multimedia information section comes after the image information part. The links information section, the table information and the summary follow the image information section. If a page does not contain any of the sections the crawler simple skips that section and goes to the next section. This component handles every HTML tag to place the content from that tag in the appropriate section.

The main purpose of this re-structuring is to provide the Concept based semantic engine with information about the data in the input text files. For example, if the engine wants information regarding multimedia from a particular file that it processed, the engine goes to the metadata directory, open the file with the same file name and searches for the five-star delimiter followed by multimedia information (*****MULTIMEDIA INFORMATION) to quickly get the information. If it wants the URL of the file then it goes to the first line of the file.

However, the Concept based engine, in its present state, will not be able to use these files as it is built only to take input text files and process them. These metadata files are a requirement of the project and will be used in the future.

4.1.1.12 Text Files

The text files contain data from each URL from the text extractor. These text files serve as input to the Concept Based Semantic Search Engine. They are stored on the hard disk at a location known to the Semantic Search Engine.

4.2 SYSTEM DESIGN

4.2.1 DATA FLOW DIAGRAMS

A data flow diagram is a graphical depiction of flow of data through intended software system and is used as 1st step to create an overview of system. It's really useful as it provides overview of data as well as functionality to software designer.

The major components of data flow diagram are
1)external entity
2)processes
3)Data flows
4)Data stores
The general rules are
1)No internal logic
2)To keep it uncluttered
3)Miracles or black holes
4)Data stores sink or sources
5)Labelling
6)No direct data flow between two entities or two data stores

For each data flow, at least one of the endpoints (source/destination) must exist in a process. The data flow diagram is part of the structured analysis modeling tools when using UML, the activity diagram typically takes over the role data flow diagram.

Below figure shows the diagram about game implementation mechanisms. First, player enter into the game and start the game. When ever playing the game score also displayed on the screen and the player scores high then update the score on the screen or the player scores less than high score then the score is not changed.

If the player wants to play one more time then use restart button and also quit the game. These steps are controlled by user.

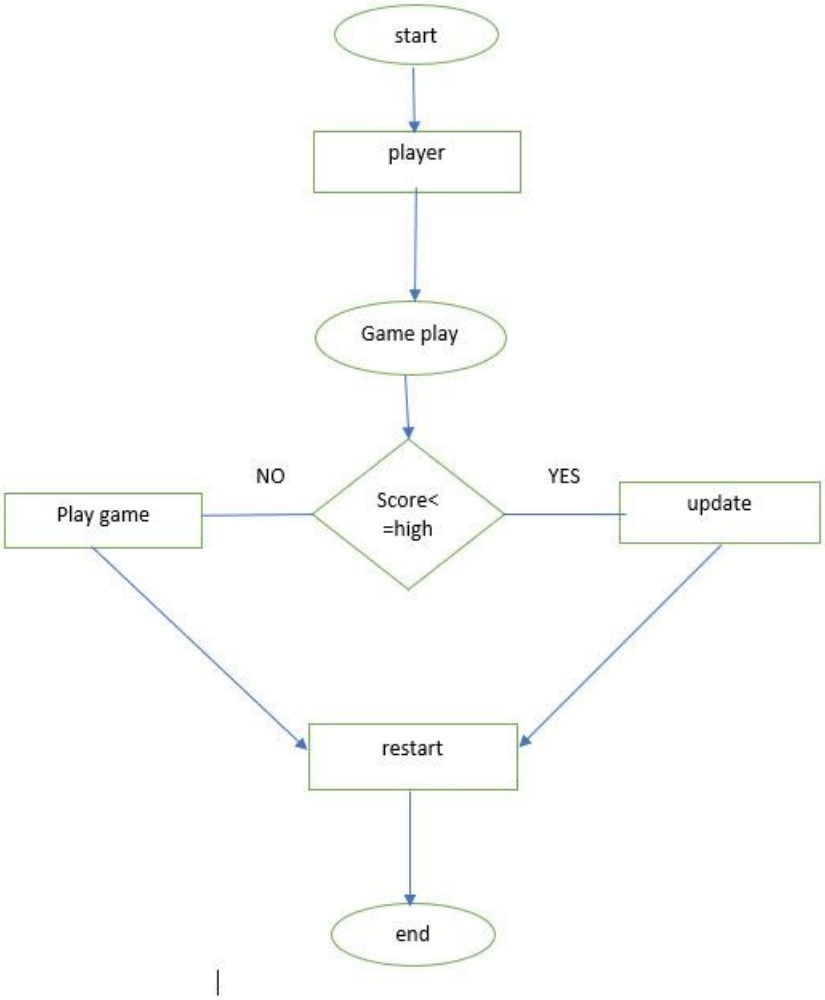

Figure 4.2.1 Data flow diagram

4.2.2 UML DIAGRAMS:

System Models: Unified Modeling Language, The Unified Modeling Language (UML) is a general-purpose graphic language used by software professionals for specifying, visualizing, constructing, and documenting the artifacts of a software intensive system. It is the standard language for writing software blueprints.

The UML has three main models:

1. The User Model
2. The Object Model
3. The Dynamic Model

4.2.2.1 The User Model

The user model consists of Use Case Diagrams which are used to graphically describe the interaction of the user with the system. They represent actors, activities and their relations.

They are primarily used to gather the requirements of the system to be built. These include both internal and external requirements and mostly design related. So when we analyze a system we identify functionalities and actors. In our Smart crawler system we have one use case as the user here is usually a programmer. The following diagram shows the use cases of the Smart crawler.

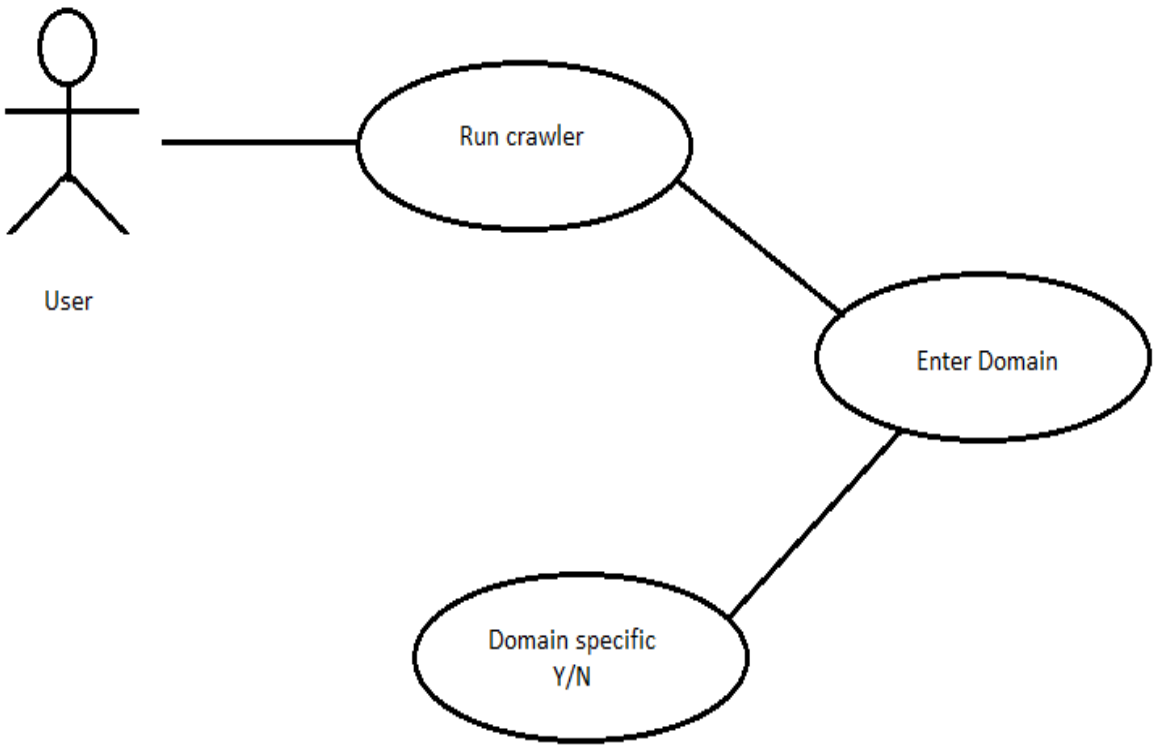

Figure 4.2.2.1: use case diagram

The Use case diagram shown above describes the interactions the user will have with the system. In the first step the user will be required to execute the crawler on his machine. This will be a python file that needs to be run using the python interpreter. He will then be asked to enter the domain he wants to start crawling.

After he enters the domain the user will be asked if he wants the crawler to crawl links only in the given domain or he wants to crawl links outside of the given domain also. He will be prompted to enter Y/N. After he enters his choice, the crawler starts crawling the links. The user will be able to see the proper text files in the directory specified.

4.2.2.2 The Object Model

The object model in the unified modeling language is represented by the class diagrams. A class diagram is used to represents the static view of an application. In addition to visualizing, describing and documenting different aspects of a system they also used for

constructing executable code of the software application. These diagrams describe attributes and operations of a class. They also describe the constraints imposed on the system. They are the only diagrams that map object orientation, so they are extensively used in objects-oriented systems.

A class diagram will show a collection of classes, interfaces, associations, collaborations and constraints. Figure 4.2.2.1 shows the class diagram for the smart crawler project. The class diagram has four main components. The main class is the class that calls all other classes whenever necessary and gets things done. It has functions like sendurls() to the web and fetch data from the internet.

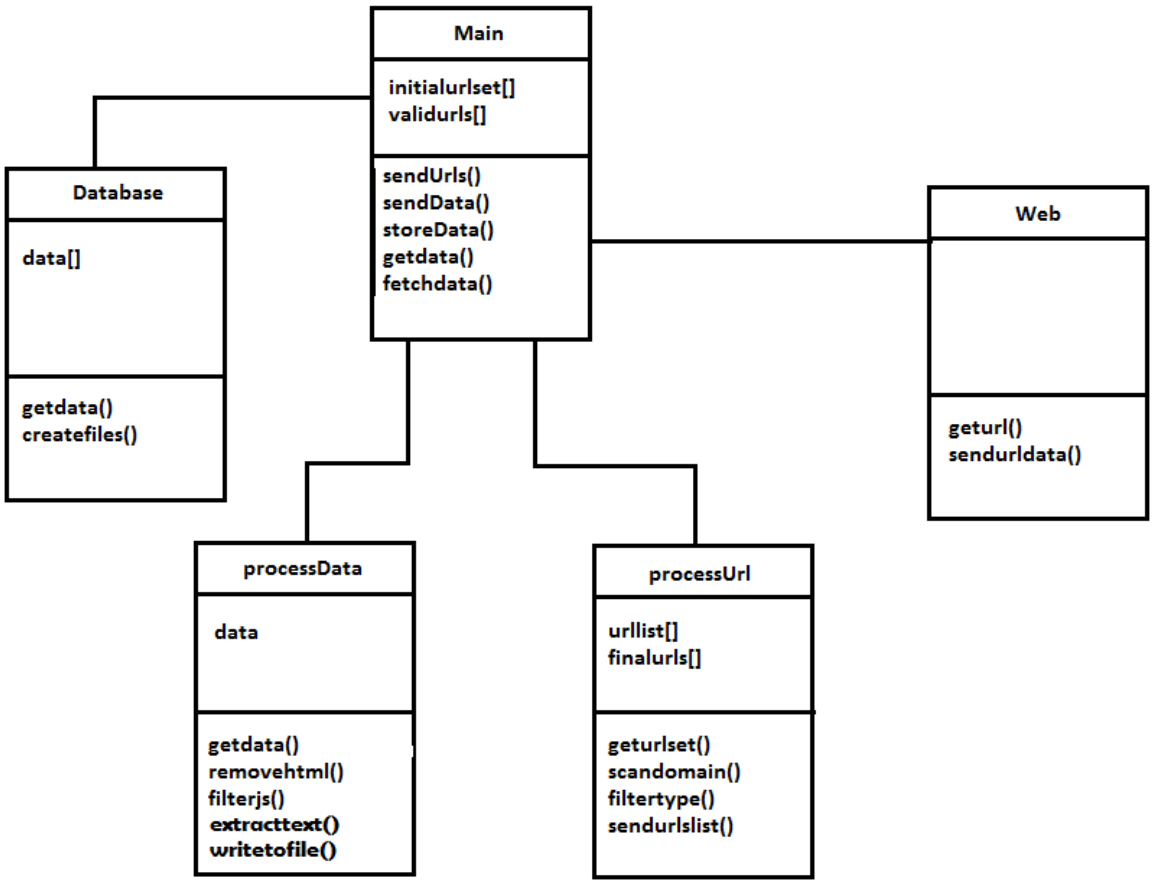

Figure 4.2.2.2 class diagram

It sends the data obtained from the web for processing to the data Process class and receives the data after processing. It sends this data to the database to create files and store the data. The process Data class gets raw data from the main class and processes it to get useful data. It has functions to remove all the html tags in the data and also functions to remove all

the java script data. It also has functions to extract the required text and format meta information. Thus, the residual data is meaningful text data that it sends back to the main class. The process URL class deals with URLs.

It gets URLs from the main class and has functions to process then, the scandomain () function checks if the current URL belongs to the given domain or not. The filterurl() function checks if the URL points to a useful file. After this processing the validUrlList is sent back to the main class.

4.2.2.3 The Dynamic Model

The Dynamic Model in the unified modeling language is represented by sequence diagrams. Sequence diagram are meant for emphasizing the time sequence of interactions between classes. The main purpose is to visualize the interactions between the components of the system.

The following is the sequence diagram for the crawler.

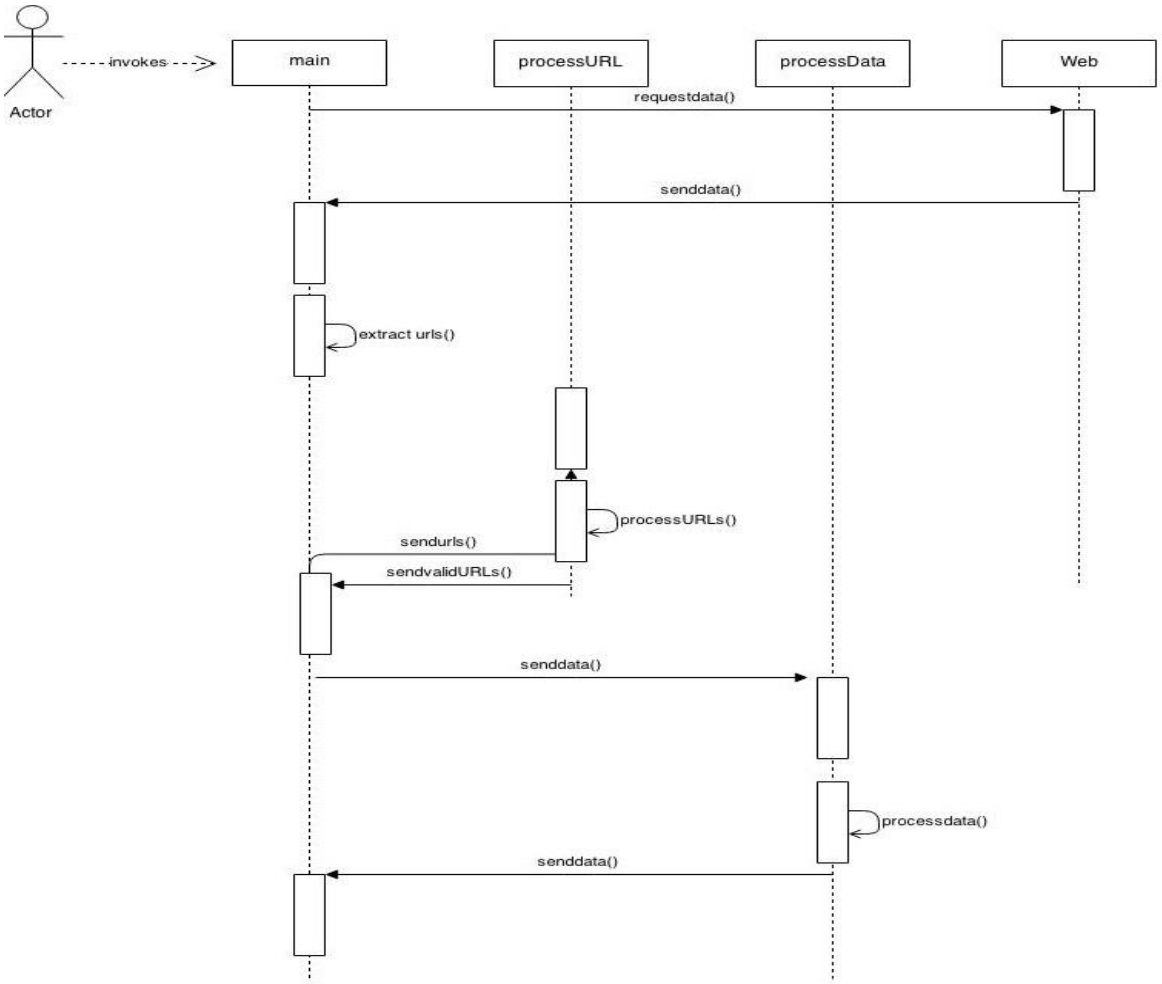

Figure 4.2.2.3: sequence diagram

4.2.3 DATA DICTIONARY

It is the centralized collection of information about data. It stores meaning and origin of data. Its relationship with other data, data format for usage etc. Data dictionary has rigorous definitions of all names in order to facilitate user and software designers. Data dictionary is often referenced as meter data (data about data) repository.

The requirement of data dictionary is

1)Data dictionary removes any chances of ambiguity.

2)It helps keeping work of programmers and designers synchronized while using some object reference everywhere in the program.

3)Data dictionary provides a way of documentation for the complete database system in one place.

Data dictionary should contain information about the

1)Data flow

2)Data structure

3)Data elements

4)Data stores

5)Data processing

Data elements consist of name and descriptions of data and control items, internal or external data stores etc.

Data stores the information from where the data enters into the system and exists out of the system.

There are two types of data processing

1)logical: As user sees it

2)Physical: As software sees it

4.2.3.1 E-R DIAGRAMS

ER Model is used to model the logical view of the system from data perspective which consists of these components:

Entity, Entity Type, Entity Set –

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

An Entity is an object of Entity Type and set of all entities is called as entity set.

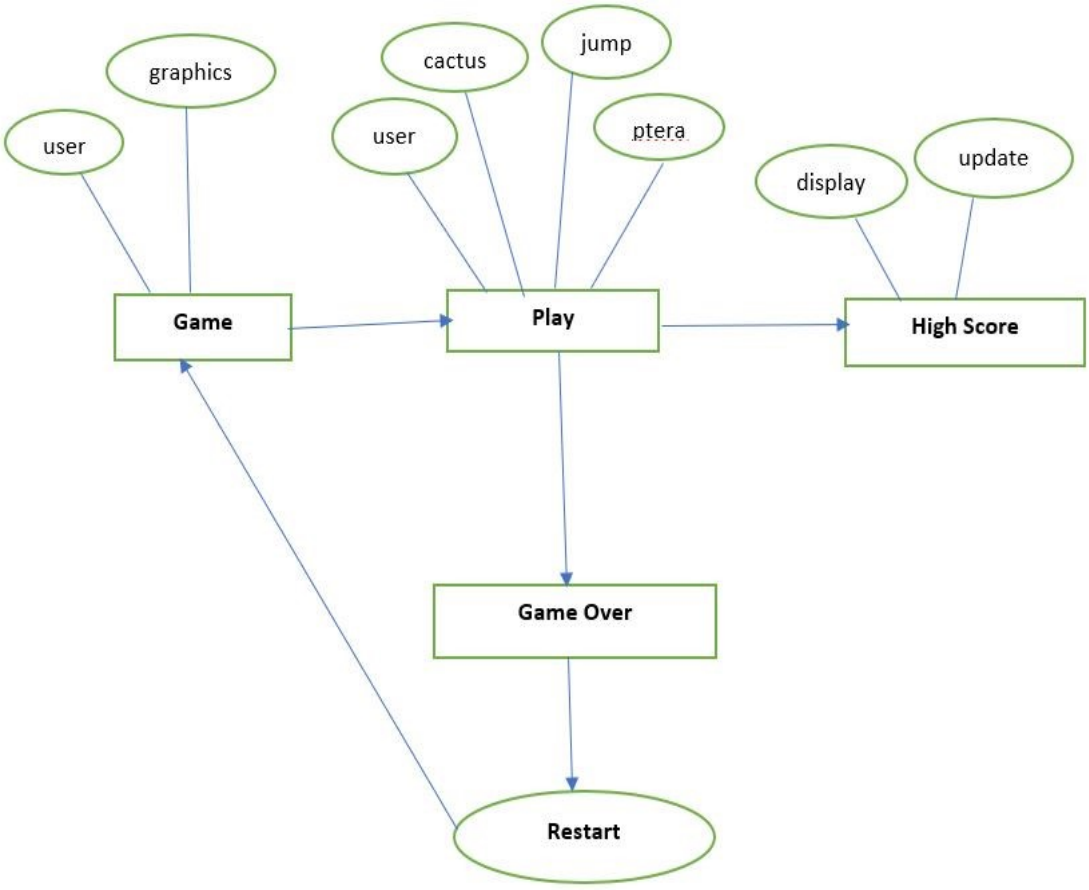

Figure 4.2.3.1 E-R diagram

4.3 TECHNOLOGIES USED

4.3.1 PYTHON

Python is currently one of the world's most used programming languages. Its syntax allows programmers to write code in far fewer lines than languages like C or Java. Code readability is the main focus of its design philosophy. Its constructs enable clean programming.

Python supports a wide range of paradigms like imperative and functional programming, object-orientation and procedural styles. It has automatic memory management, dynamic type system and a comprehensive standard library.

Python can be installed on many operating systems allowing python to be executed on a majority of systems. Third party installers are also available for every popular operating system for python which can be executed to install python. Python is both an object-oriented programming and structured programming. It also has a large number of features that support aspect-oriented and functional programming.

The following are the features of python programming language that sets it apart from other languages.

4.3.1.1 Simple

Python is a surprisingly simple language. Python code is easy to understand almost like English. One of the greatest strengths is its pseudo code nature, it allows programmer to concentrate on finding the solution of the problem without worrying much about the language specifics.

4.3.1.2 Portable

As Python is open-source, it has been ported to many platforms. It is made to work on many platforms. Python programs do not need any changes on different platforms if we can avoid system-dependent features. We can use Python on Windows, Linux, PlayStation, Sharp Zaurus, Windows CE, AROS, AS/400, BeOS, OS/390, z/OS or even PocketPC.

4.3.1.3 Free and Open Source

Python is a FLOSS (Free/LibrÃ© and Open Source Software). You can freely read its source code, distribute copies of this software, and use pieces of it in new free programs, make changes to it.

This is one reason python has been constantly improving over the years. Free access has allowed people to develop python into a language with full-fledged libraries.

4.3.1.4 Interpreted

Python, unlike C or C++, does not need compilation to binary. Compilation is a process of conversion of source language to a language understandable by the computer. Usually a compiler does this job for languages like C or C++.

But python runs the program directly from source code. It internally converts the code in byte code before running it converts it into the native language of the computer. So we don’t have to worry about compiling, linking libraries, etc. This also makes python potable.

4.3.1.5 High level Language

We don’t have to worry about the low-level details like memory management of the programs as Python is a high-level language. It handles memory by itself.

4.3.1.6 Extensive Libraries

Python has a huge Standard Library. We can do various things involving unit testing, documentation generation, regular expressions, HTML, WAV files, XML, XMLRPC, HTML, cryptography, GUI (graphical user interfaces) WAV files, threading, databases, web browsers, CGI, FTP, email, Tk, and other system-dependent stuff.

This is called the 'Batteries Included' Python philosophy. All of this is installed when Python is installed. Apart from this there are many high-quality libraries like Python Imaging Library, Twisted etc.

4.3.1.7 Object Oriented

Python supports object- oriented as well procedure-oriented programming. The program is built around functions which are nothing but reusable pieces of code in procedure-oriented language. And in object-oriented languages, the program is built around objects that combine data and functionality.

Python has a very simple but powerful way of doing object-oriented programming when compared to languages like C++ or Java. Python is a language that is really simple to learn. Even beginners will find it simple to start learning.

4.3.1.9 Extensible

Python is extensible. If there is any part of code or algorithm that is available in other languages, that part of code can be extended and used in python.

4.3.1.10 Embeddable

Python code can be embedded in other languages like C or C++ and the scripting capabilities of python can be used in those languages.

4.3.2 WEB CRAWLER

A web crawler is a program that goes around the internet collecting and storing data in a database for further analysis and arrangement. The process of web crawling involves gathering pages from the web and arranging them in such a way that the search engine can retrieve then efficiently.

The critical objective is to do so efficiently and quickly without much interference with the functioning of the remote server. A web crawler begins with a URL or a list of

URLs, called seeds. The crawler visits the URL at the top of the list. On the web page it looks for hyperlinks to other web pages, it adds them to the existing list of URLs in the list. This methodology of the crawler visiting URLs depends on the rules set for the crawler. In general crawlers incrementally crawl URLs in the list. In addition to collecting URLs the main function of the crawler, is to collect data from the page. The data collected is sent back to the home server for storage and further analysis. The Smart Crawler for the Concept based Semantic Search Engine described here crawls the internet collecting web pages and storing them in the form of text files.

- This is because the Concept based Semantic Search engine will take inputs only in the form of text files. In order to improve the efficiency of the engine the Smart crawler filters the text before storing them.

- In addition to filtering, the Smart crawler skips crawling URLs of files like image or mp3 files that contain non-textual data.

- The main contribution of the Smart crawler in comparison to the existing crawler is that the Smart Crawler performs an advanced level of data analysis on the data extracted from the web.

- It also follows a breadth first mode of link traversal unlike the existing crawler. The Smart crawler also has the ability to filter out URLs like image and multimedia that do not contain any useful text information.

The Smart crawler also effectively manages the metadata by systematically storing it. In this section we describe characteristics and classification of crawlers. We also give some background information about the programming language used to build the crawler. The second section describes the architecture of the Smart crawler. The third section describes the implementation details of the crawler.

The results section compares the performance of the Concept Based semantic Engine when input from the Smart Crawler is given and from the existing crawler is given. This proves how the techniques used to analyze and extract data from HTML pages improve the efficiency of the Concept based Semantic Search engine.

4.3.2.1 Characteristic features of Crawlers

Crawlers that crawl the internet must have the following basic features so that they serve their purpose, the well being of the servers that hold data and also the web as a whole.

4.3.2.2 Robustness

The web contains loops called spider traps, which are meant to mislead the crawler to recursively crawl a particular domain and get stuck in one single domain. They generate an infinite loop of web pages that lead to nowhere.

The crawler needs to be resilient to such traps. These traps may not always be designed to mislead the crawler but may be a result of faulty website development.

4.3.2.3 Politeness

Web servers have policies regulating when a crawler can visit them. These politeness policies must be respected. A server is meant to serve other requests that it is originally designed to serve. Hindering the server may lead to blocking of the crawler by the server altogether. So it is better to respect the policies of the server.

4.3.2.4 Distributed

The crawler should be able to function in a distributed fashion. It could have multiple images of itself working parallelly in proper coordination to crawl the internet as quickly as possible.

4.3.2.5 Scalable

The crawler should be scalable. It should have the flexibility to add new machines and extra bandwidth whenever necessary.

4.3.2.6 Performance and Efficiency

The use of system resources like processing power, network bandwidth and storage should be judicious. These factors determine how efficient the crawler is.

4.3.2.7 Quality

The crawler should be able to differentiate between information that is useful and information that is not. As servers mainly serve other requests that contain a lot of information that may not be useful. Crawlers should filter out this content.

4.3.2.7 Freshness

In many situations, crawlers will need to crawl the same pages again in order to get new content from the old page. For this reason, crawlers should be able to crawl the same

page at a rate that is approximately equal to the rate of change of information on the page. Thus, the crawler will be able to make sure that the concepts on the search engine are the latest and relevant to the present context.

4.3.2.8 Extensible.

The crawlers should be able to adapt to the growing number of data formats that it will encounter on web sites. It also needs to cope up with the new protocols that may be used on some servers.

4.3.3 CHALLENGES FOR A CRAWLER

The basic algorithm for a crawler will look simple at first glance. It simply involves going to a URL, getting URLs from the web page and iteratively going to each of those URLs, get data from them and store the data. Despite the simplicity, there are some intense challenges that we need to tackle to make a crawler. The following are the challenges for a crawler.

4.3.3.1 Scale

The internet is an enormous evolving and ever growing system. The crawlers need to have good freshness; seek broad coverage and extremely high throughput. It is difficult to achieve all of these at the same time. Modern companies use complex high-speed network links and processing power but expert programmers still fall short of a perfect web crawling bot.

4.3.3.2 Content selection tradeoff

Even the best crawlers do not intend to crawl the whole internet, or keep up with all the changes that happen on the web. Instead, crawling is performed selectively and in a controlled order.

- The goal is to acquire valuable content quickly, bypass low-quality, irrelevant, redundant, and malicious content and ensure eventual coverage of all reasonable content.
- The crawler must obey constraints such as per-site rate limitations while balancing competing objectives such as coverage and freshness.
- A trade-off must be achieved between exploitation of content already known to be useful and exploration of potentially useful content.

4.3.3.3 Social Obligation

Crawlers should not impose too much burden on the web to crawl web pages. They should be good bots. They should not over burden the servers hosting the data in times when the load is heavy and lead to crashing of services. They should follow the rules of the servers. Crawlers if used without the right safety mechanisms can even cause denial of-service attacks.

4.3.3.4 Adversaries

Some content providers seek to inject misleading content or useless information into the data assembled by the crawler. This is often done for monetary benefits like commercial advertisements. This may also be done by other competing data miners to outsmart other aggregators.

There are many crawlers written in every programming and scripting language to serve a variety of purposes depending on the requirement, purpose and functionality for which the crawler is built. The first ever web crawler to be built to fully function is the WebCrawler in 1994.

Subsequently a lot of other better and more efficient crawlers were built over the years. The most notable of the crawlers currently in operation are as follows.

- Googlebot: The Google search uses this crawling bot. It is integrated with indexing process as parsing is done for URL extraction and also full text indexing. It has a URL server that exclusively handles URLs. It checks if the URLs have previously been crawled. It they are not crawled they are added to the queue.
- Bingbot: The Bingbot is the crawler that the Microsoft owned search engine Bing Search uses to crawl the web and collect data. It was previously known as Msnbot.
- FAST Crawl: This is the web crawler that the Norway based Fast Search and Transfer uses. It focuses on data search technologies. It was first developed in 1997 and is periodically re-developed based on latest technologies.
- WebRACE: It is a Java based crawler. It acts in part as a proxy server as it gets requests from users to download pages. When pages change, they are crawled again and the subscriber is notified. The feature of this bot is it does not need a set of seeds to start crawling.
- WebFountain: It is a distributed crawler written in C++. It has a controller and ant machines that repeatedly download pages. A non-linear programming method is used to solve freshness maximizing equations.

We also have a lot of open source crawlers that are available online and can be used according to needs for noncommercial purposes.

- DataparkSearch: This is a search engine with a crawler under GNU.
- GNU Wget: It is a command line operated crawler in C language. It is usually used to mirror web and ftp sites.
- GRUB: Wiki search uses this to crawl the web.
- Heritrix: This is a Java based crawler used for archiving large portions of the web.
- HTTrack: It is a C language-based crawler that creates an image of web pages to access then offline.
- ICDL Crawler: A cross-platform web crawler in C++.
- mnoGoSearch: A c-based crawler and indexer with a search engine.
- NOrconex HTTP Collector: A java-based crawler that helps Enterprise Search Integrators.
- Nutch: A Java based crawler used in with a text-indexing package.

4.4. SAMPLE CODE

Dino.py

```
class Dino():
    def __init__(self,sizex=-1,sizey=-1):
        self.images,self.rect = load_sprite_sheet('dino.png',5,1,sizex,sizey,-1)
        self.images1,self.rect1 = load_sprite_sheet('dino_ducking.png',2,1,59,sizey,-1)
        self.rect.bottom = int(0.98*height)
        self.rect.left = width/15
        self.image = self.images[0]
        self.index = 0
        self.counter = 0
        self.score = 0
        self.isJumping = False
        self.isDead = False
        self.isDucking = False
        self.isBlinking = False
        self.movement = [0,0]
        self.jumpSpeed = 11.5
        self.stand_pos_width = self.rect.width
        self.duck_pos_width = self.rect1.width
    def draw(self):
        screen.blit(self.image,self.rect)
    def checkbounds(self):
        if self.rect.bottom > int(0.98*height):
            self.rect.bottom = int(0.98*height)
            self.isJumping = False
    def update(self):
        if self.isJumping:
            self.movement[1] = self.movement[1] + gravity
        if self.isJumping:
            self.index = 0
        elif self.isBlinking:
            if self.index == 0:
                if self.counter % 400 == 399:
```

```
                self.index = (self.index + 1)%2
            else:
                if self.counter % 20 == 19:
                    self.index = (self.index + 1)%2
        elif self.isDucking:
            if self.counter % 5 == 0:
                self.index = (self.index + 1)%2
        else:
            if self.counter % 5 == 0:
                self.index = (self.index + 1)%2 + 2
        if self.isDead:
            self.index = 4
        if not self.isDucking:
            self.image = self.images[self.index]
            self.rect.width = self.stand_pos_width
        else:
            self.image = self.images1[(self.index)%2]
            self.rect.width = self.duck_pos_width
        self.rect = self.rect.move(self.movement)
        self.checkbounds()
        if not self.isDead and self.counter % 7 == 6 and self.isBlinking == False:
            self.score += 1
            if self.score % 100 == 0 and self.score != 0:
                if pygame.mixer.get_init() != None:
                    checkPoint_sound.play()
        self.counter = (self.counter + 1)
class Cactus(pygame.sprite.Sprite):
    def __init__(self,speed=5,sizex=-1,sizey=-1):
        pygame.sprite.Sprite.__init__(self,self.containers)
        self.images,self.rect = load_sprite_sheet('cacti-small.png',3,1,sizex,sizey,-1)
        self.rect.bottom = int(0.98*height)
        self.rect.left = width + self.rect.width
        self.image = self.images[random.randrange(0,3)]
        self.movement = [-1*speed,0]
```

```
    def draw(self):
        screen.blit(self.image,self.rect)
    def update(self):
        self.rect = self.rect.move(self.movement)
        if self.rect.right < 0:
            self.kill()
class Ptera(pygame.sprite.Sprite):
    def __init__(self,speed=5,sizex=-1,sizey=-1):
        pygame.sprite.Sprite.__init__(self,self.containers)
        self.images,self.rect = load_sprite_sheet('ptera.png',2,1,sizex,sizey,-1)
        self.ptera_height = [height*0.82,height*0.75,height*0.60]
        self.rect.centery = self.ptera_height[random.randrange(0,3)]
        self.rect.left = width + self.rect.width
        self.image = self.images[0]
        self.movement = [-1*speed,0]
        self.index = 0
        self.counter = 0
    def draw(self):
        screen.blit(self.image,self.rect)
    def update(self):
        if self.counter % 10 == 0:
            self.index = (self.index+1)%2
        self.image = self.images[self.index]
        self.rect = self.rect.move(self.movement)
        self.counter = (self.counter + 1)
        if self.rect.right < 0:
            self.kill()
class Cloud(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self,self.containers)
        self.image,self.rect = load_image('cloud.png',int(90*30/42),30,-1)
        self.speed = 1
        self.rect.left = x
        self.rect.top = y
        self.movement = [-1*self.speed,0]
```

```
    def draw(self):
        screen.blit(self.image,self.rect)
    def update(self):
        self.rect = self.rect.move(self.movement)
        if self.rect.right < 0:
            self.kill()
class Scoreboard():
    def __init__(self,x=-1,y=-1):
        self.score = 0
        self.tempimages,self.temprect = load_sprite_sheet('numbers.png',12,1,11,int(11*6/5),-1)
        self.image = pygame.Surface((55,int(11*6/5)))
        self.rect = self.image.get_rect()
        if x == -1:
            self.rect.left = width*0.89
        else:
            self.rect.left = x
        if y == -1:
            self.rect.top = height*0.1
        else:
            self.rect.top = y
    def draw(self):
        screen.blit(self.image,self.rect)
    def update(self,score):
        score_digits = extractDigits(score)
        self.image.fill(background_col)
        for s in score_digits:
            self.image.blit(self.tempimages[s],self.temprect)
            self.temprect.left += self.temprect.width
        self.temprect.left = 0
 if gameQuit:
            break
        while gameOver:
            if pygame.display.get_surface() == None:
                print("Couldn't load display surface")
```

```
            gameQuit = True
            gameOver = False
        else:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    gameQuit = True
                    gameOver = False
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_ESCAPE:
                        gameQuit = True
                        gameOver = False
                    if event.key == pygame.K_RETURN or event.key == pygame.K_SPACE:
                        gameOver = False
                        gameplay()
        highsc.update(high_score)
        if pygame.display.get_surface() != None:
            disp_gameOver_msg(retbutton_image,gameover_image)
            if high_score != 0:
                highsc.draw()
                screen.blit(HI_image,HI_rect)
            pygame.display.update()
        clock.tick(FPS)
pygame.quit()
quit()
```

Chapter 5
SYSTEM TESTING

Many would argue unit-testing improves your productivity as a developer. While working the display driving code, I discovered that the display had no built-in font, so I'd need to store the bitmaps for all the characters that I wanted to display, and send them down the wire whenever I wanted to draw some character. Furthermore, it turned out that I in some cases, depending on where I wanted the character to appear on the screen, I had to split the bitmap payload into several chunks, due to how the protocol worked.

This all turned out to be pretty complicated. The development cycle of changing one or two lines of code, sending it to the hardware and watching the display to figure out the result was exhausting. At that point, I decided to set up unit-tests so that I'd be able to test the logic of my code without having to run it on the actual hardware.

For this task, I went with Jest, a JavaScript test runner from Face book, which is also what I use when I write Frontend code. Jest only runs the tests that were affected by files modified since your last commit, and it can run tests in parallel, making it much more efficient for larger projects.

It has several others features that I like, such as snapshot support and a nice mocking library. My setup also included Wallaby.js, which allows me to see the test results update in my IDE in real-time, as I write the code.

Once I configured Jest, I was able to quickly iterate with the more complex display control functions. If you look into the test code, you can see how I mocked the Espruino functions that interface with the hardware - such as digital Write and pin Mode. This allowed me to make assertions on the actual bytes that would go down the wire whenever I call the write Char function.

Another place where the unit-testing proved very useful was persisting the high scores. The embedded processor had Flash Memory, just like your smart phone. Flash memory has an interesting property that is usually abstracted by the Operating System (meaning we, as developers, don't have to worry about it).

Generally speaking, you can only write to each memory location once, and then, if you ever want to update that memory location, you need to erase the entire 4KB memory block, which is a time-consuming operation, and also slowly wears out the memory cells.

5.1 Different Types of Game Testing Techniques

Game testing is the most important part in a game development process. This is the final component that analyses whether your gaming application is ready for launch or not. Such services give the development process a critical eye to focus on constant searches like inconsistencies, errors, coherence and completeness, etc.So, want to know about the types of game testing methods? Following are some game testing techniques with respect to mobile software testing:

1.Combinatorial Testing:

This is a method of experimental design that is used for commercial software testing and to generate test cases. Applying combinatorial testing to game testing increases test execution efficiency, provide better quality, reduce cost and better phase containment.

2.Clean Room Testing:

This is a software development process intended to develop gaming software with a certifiable level of reliability.

3.Functionality Testing:

This literally means the method to identify bugs or errors in a game that may affect the user-experience.

4.Compatibility Testing:

This is used to find whether a game is functioning properly or not with respect to the hardware, graphics and software configuration that the device is built with. It is one of the essential mobile app testing services that checks if a game title is able to run on specific devices.

5.Tree Testing:

This type of testing is almost same to usability testing used to organize the test cases. It also helps to select the proper set of tests for the given set of code changes.

6.Regression Testing:

Regression testing is done to retest the unchanged parts of the software. Here test cases are re-checked to analyze the working of the previous functions of the app works fine and that new

Changes have not introduced any new errors or vulnerabilities.

7.Ad hoc Testing:

This is quite an unplanned testing method generally used to break down the system. Testers randomly test the app without test cases or any documents.

8.Load Testing:

It is a type of performance testing to determine the performance of a system under real-time loads. Load testing shows the reaction of an app when multiple users use the app simultaneously.

9.Play Testing:

Play testing is the method of game testing by playing the game to analyze non-functional features like fun factors, difficulty levels, balance, etc. Here a selected group of users plays the unfinished versions of the game to check the work flow.

5.2 Python Testing

Unit testing:

The unit test unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

To achieve this, unit test supports some important concepts in an object-oriented way:

test fixture

A test fixture represents the preparation needed to perform one or more tests, and any associated cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.

test case

A test case is the individual unit of testing. It checks for a specific response to a particular set of inputs.

Unit test provides a base class, Test Case, which may be used to create new test cases.

test suite

A test suite is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.

test runner

A test runner is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

The unit test module provides a rich set of tools for constructing and running tests. This section demonstrates that a small subset of the tools suffices to meet the needs of most users.

Here is a short script to test three string methods:

```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a
string
        with self.assertRaises(TypeError):
            s.split(2)

if __name__ == '__main__':
```

```
    unittest.main()
```

A test case is created by sub classing unittest.TestCase. The three individual tests are defined with methods whose names start with the letters test. This naming convention informs the test runner about which methods represent tests.

The crux of each test is a call to assertEqual() to check for an expected result; assertTrue() or assertFalse() to verify a condition; or assertRaises() to verify that a specific exception gets raised. These methods are used instead of the assert statement so the test runner can accumulate all test results and produce a report.

The setUp() and tearDown() methods allow you to define instructions that will be executed before and after each test method. They are covered in more detail in the section Organizing test code.

The final block shows a simple way to run the tests. unittest.main() provides a command-line interface to the test script. When run from the command line, the above script produces an output that looks like this:

```
...
----------------------------------------------------------------------
-----
Ran 3 tests in 0.000s

OK
```

Passing the -v option to your test script will instruct unittest.main() to enable a higher level of verbosity, and produce the following output:

```
test_isupper (__main__.TestStringMethods) ... ok
test_split (__main__.TestStringMethods) ... ok
test_upper (__main__.TestStringMethods) ... ok

----------------------------------------------------------------------
-----
Ran 3 tests in 0.001s

OK
```

The above examples show the most commonly used unittest features which are sufficient to meet many everyday testing needs. The remainder of the documentation explores the full feature set from first principles.

The unit test module can be used from the command line to run tests from modules, classes or even individual test methods:

```
python -m unittest test_module1 test_module2
python -m unittest test_module.TestClass
python -m unittest test_module.TestClass.test_method
```

You can pass in a list with any combination of module names, and fully qualified class or method names.Test modules can be specified by file path as well:

```
python -m unittest tests/test_something.py
```

This allows you to use the shell filename completion to specify the test module. The file specified must still be importable as a module. The path is converted to a module name by removing the '.py' and converting path separators into '.'. If you want to execute a test file that isn't importable as a module you should execute the file directly instead.

You can run tests with more detail (higher verbosity) by passing in the -v flag:

```
python -m unittest -v test_module
```

When executed without arguments Test Discovery is started:

```
python -m unittest
```

For a list of all the command-line options:

```
python -m unittest -h
```

Chapter 6
RESULTS AND OUTPUT SCREENS

1 GAME PAGE:

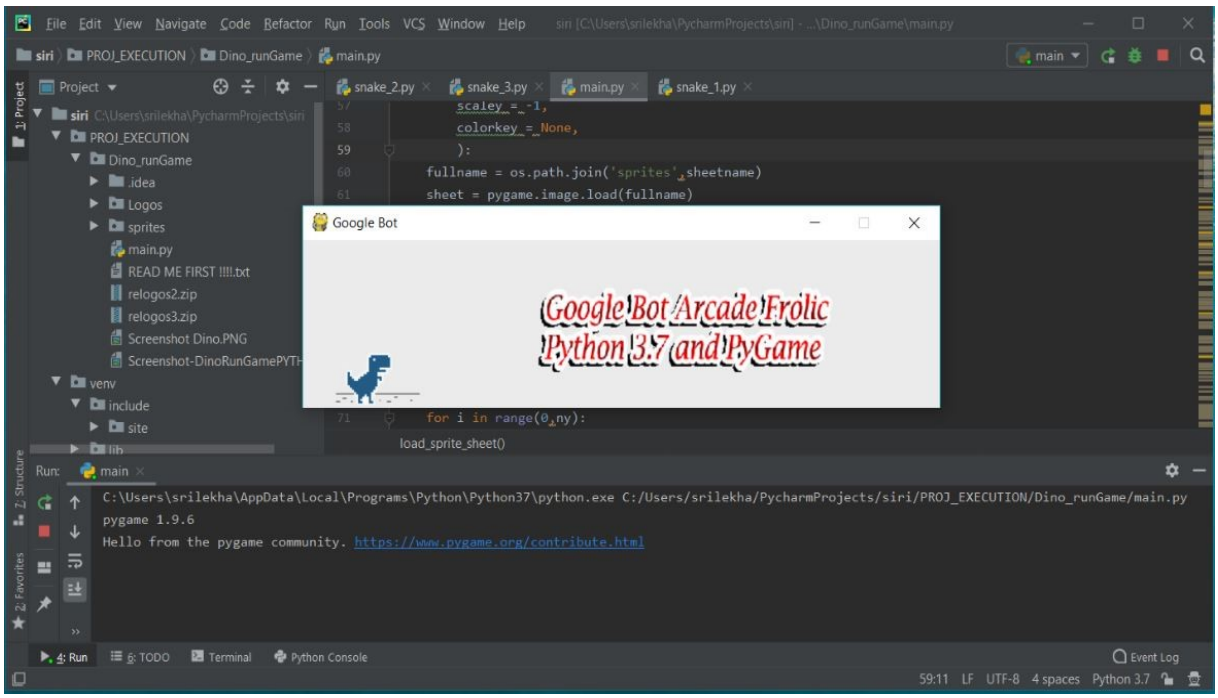

Figure 6.1.1 screenshot for open a game page

The program execution completed then the first output screen is display on the window like above figure.

2.USER ENTERING THE GAME

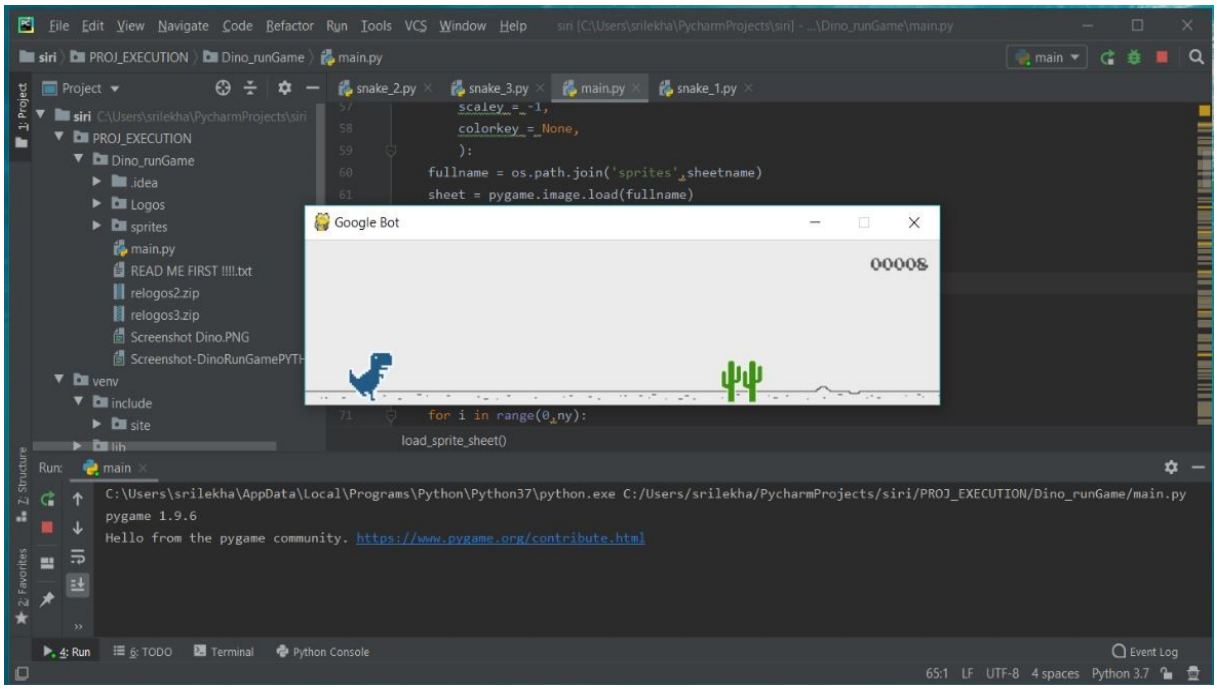

Figure 6.1.2: screenshot for user entering the game

When output screen displayed on the window if the player wants to play the game then click the space button, game started like above figure and score will be displayed on the top of the window. The score is updated every second while playing the game.

3.DINO JUMPING ON CACTUS

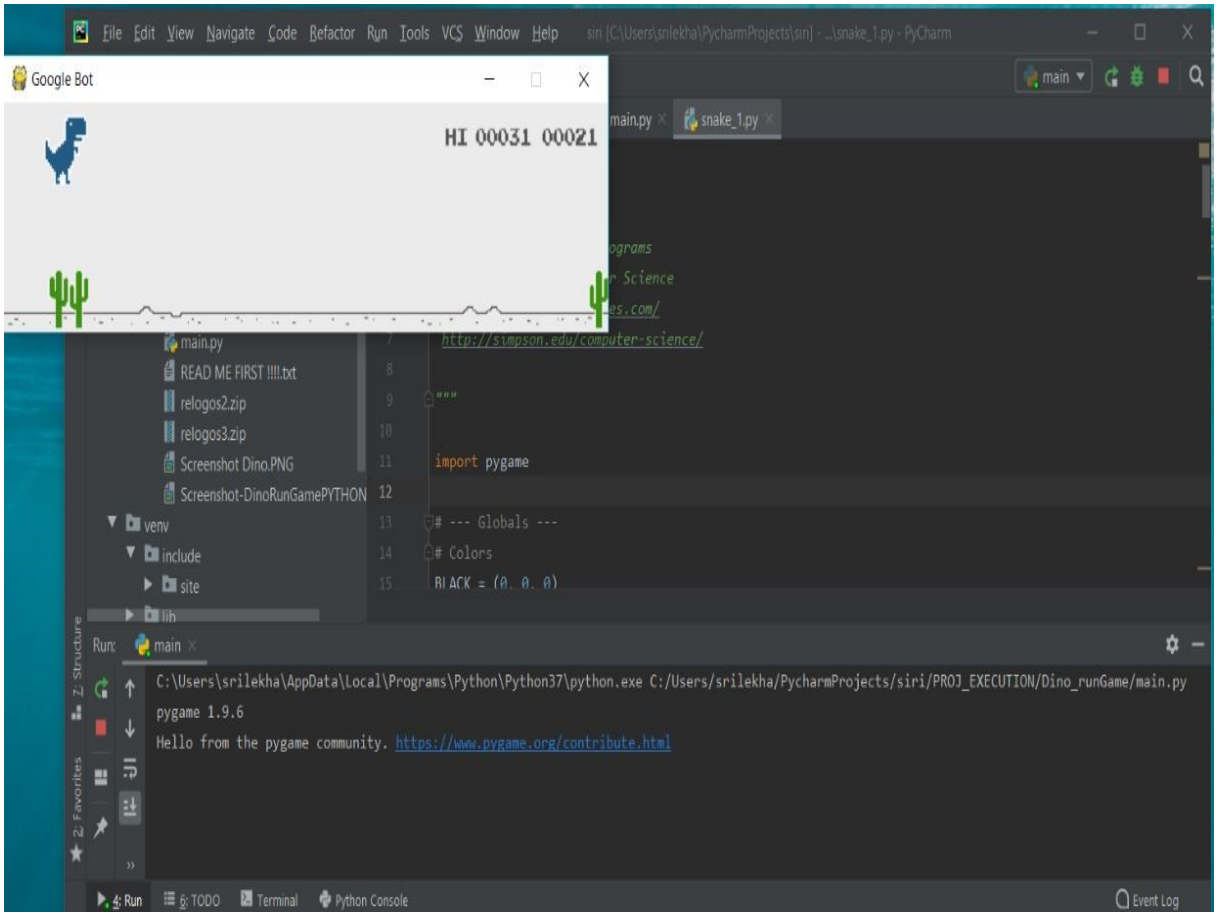

Figure 6.1.3: screenshot for dino jumping on cactus

The game is a simple infinite runner, which sees you jump over cacti, and underneath obstacles. Controls are basic. Press space to jump (and to start the game), and the down arrow to duck.

4. DINO WITH PTERA

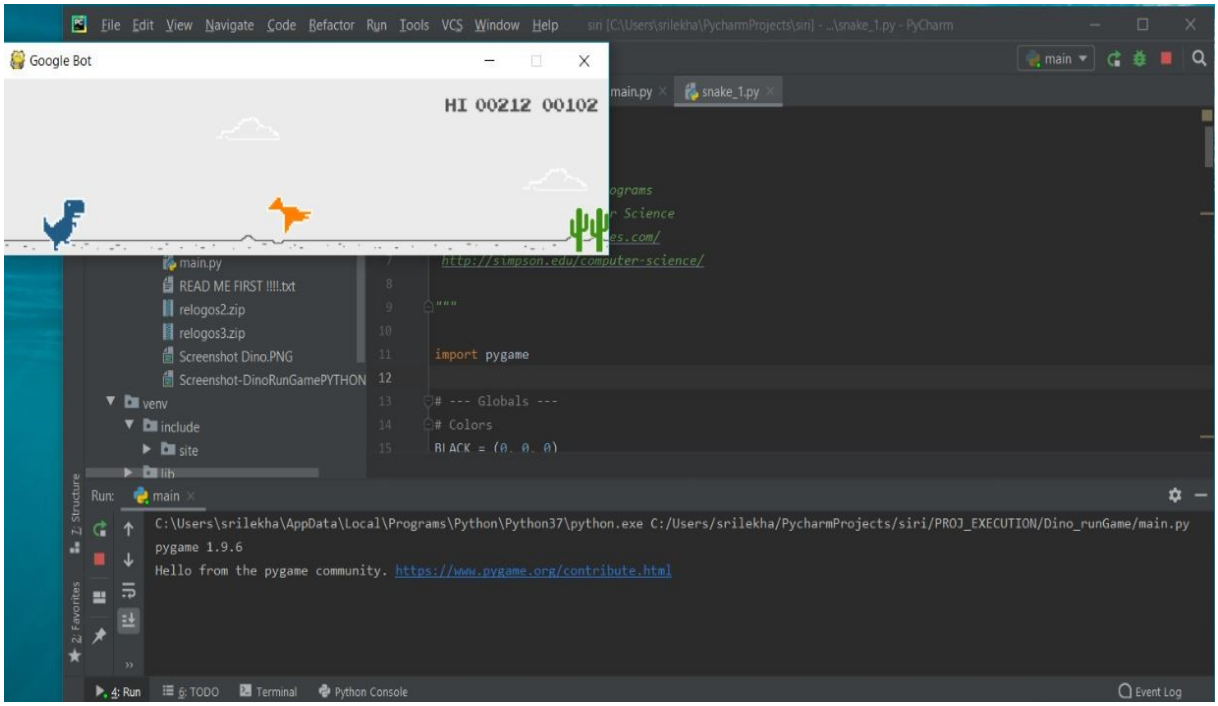

Figure 6.1.4: screenshot for dino with ptera

we are adding the new feature ptera the birds are coming randomly while playing the game and whenever score is become 100 on that time indication sound is also added.

5. GAMEOVER BY PTERA:

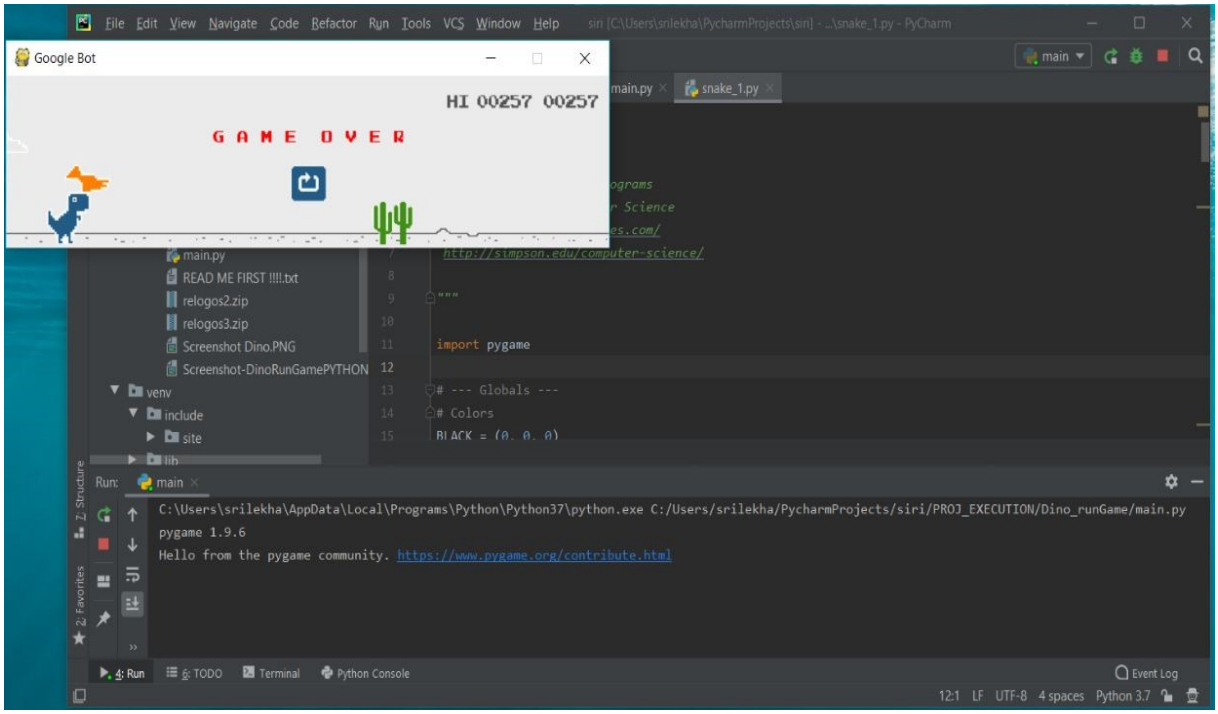

Figure 6.1.5: screenshot for game over by ptera

The dino is touched on the ptera (bird), game will be over and it is displayed on the screen, like above figure.

6.GAME OVER BY CACTUS

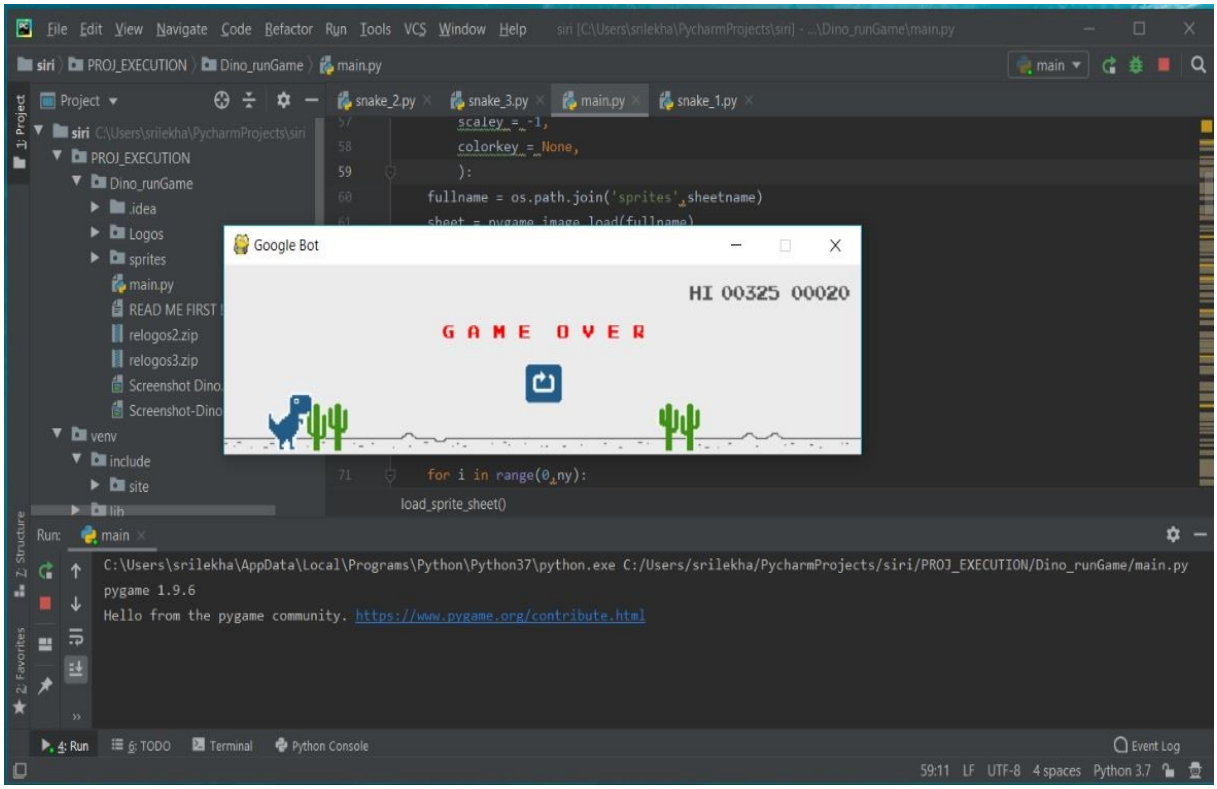

Figure 6.1.6: screenshot for game over by cactus

The dino is touched on the cactus also game will be over and it is displayed on the screen, like above figure.

CONCLUSION

Several approaches were used to achieve the AI that can play Dinosaur Game. For the feature extraction-based algorithm, computer version methods can recognize the T-Rex and obstacles from the images. Carefully designed feature extraction algorithms can successfully abstract the state and AI built upon them can improve its performance significantly compared with naive baseline. MLP (master limited partnership) learned from online training can strengthen the AI further for it refines the parameters automatically by experience.

However, feature-extraction based algorithm have their limits and cannot outperform the human experts. For end-to-end Deep-Q learning method, our result shows that it can successfully play the game by learning straightly from the pixels without feature extraction, and is much stronger than the feature-based method. Finally, specially designed training method can help us overcome the training difficulties caused by the properties of our game, which further improves our AI's performance and helps achieve super-human results.

FUTURE ENHANCEMENT

In our project, neither MLP nor Deep Q-learning handles velocity well enough. In Deep Q-learning, we notice velocity makes great impact over the jumping position selection. During the process of training agent with different acceleration and velocity, we observe that the agent tends to use policies incorrectly fitting the current velocity when the relative speed changes. Two possible explanations are as follows. Firstly, we simplify the computation in neural networks by resizing the raw game image into 80×80 pixels. In this way, the edge of the obstacles would be obscure which negatively influenced the prediction. Another problem is that we use four images in stacking to infer the relative velocity based on their differences. If more channels are added in the game image, more deviation would be detected in both MLP and Deep Q-learning in order to capture the change of velocity.

REFERENCES

- Enabling Reusable Alternative Text Descriptions using Reverse Image Search Alexa top 500 global sites on the web, 2017. https://www.alexa.com/topsites.
- Ayar Pranav et al, International Journal of Computer Science and Mobile Computing, Vol.4 Issue.5, May- 2015, pg. 545-551© 2015, IJCSMC All Rights Reserved 545. Efficient Focused Web Crawling Approach for Search Engine, Available at https://www.ijcsmc.com/docs/papers/May2015/V4I5201599a17.pdf .
- International Journal of Computer Trends and Technology (IJCTT) – volume 13 number 3 – Jul 2014 ISSN: 2231-280,Web Crawler: Extracting the Web Data, https://www.ijcttjournal.org/Volume13/number-3/IJCTT-V13P128.pdf .
- International Journal of Computer Theory and Engineering, Vol. 5, No. 2, April 2013. Krishan Kant Lavania, Sapna Jain, Madhur Kumar Gupta, and Nicy Sharma,Google: A Case Study (Web Searching and Crawling) http://www.ijcttjournal.org .

Referred books:

The Python.org document section

- Current python 2.x documentation

Python Wiki at python.org

- Python IDEs
- www.python.org
- Learn python programming book
- Python programming: An introduction to computer science (3rd edition)