



A

PROJECT REPORT

FOR

SUBJECT: LAB II- PROJECT PHASE II

ON

Grains Distribution and Export Optimization System

Submitted in partial fulfilment of the requirement for the award of

Bachelor of Engineering

In

Computer Science and Engineering

Solapur University

By

Abhijeet Kothari Roll No. 210345

Shreyans K. Jain Roll No. 210281

Vikram Kothekar Roll No. 210347

Bahauddin Kalyani Roll No. 210335

Under Guidance Of

Prof. D. P. Gandhmal



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
WALCHAND INSTITUTE OF TECHNOLOGY
SOLAPUR - 413006
(2016-2017)



CERTIFICATE

This is to certify that the project entitled

Grains Distribution and Export Optimization System

Is

Submitted By

Abhijeet Kothari	Roll No. 210345
Shreyans K. Jain	Roll No. 210281
Vikram Kothekar	Roll No. 210347
Bahauddin Kalyani	Roll No. 210335

(Prof. D.P. Gandhmal)
Project Guide

(Prof. R. V. Argiddi)
Head, C.S.E Department

(Dr. S.A. Halkude)
Principal

Acknowledgement

“Task successful” makes everyone happy. But the happiness will be gold without glitter if we didn’t state the persons who have supported us to make it a success.

Success will be crowned to people who made it a reality but the people whose constant guidance and encouragement made it possible will be crowned first on the eve of success.

This acknowledgement transcends the reality of formality when we would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped us for the completion of our project work.

We consider ourselves lucky enough to get such a good project. This project would add as an asset to our academic profile.

We would like to express our thankfulness and gratitude to our project guide **Prof. D.P. Gandhmal** for his constant motivation and valuable help throughout the project work.

Last but not the least, the backbone of our success and confidence lies solely on blessing of our parents. We also express our sincere gratitude towards all others who have been directly or indirectly help towards the completion of this project.

Team:

Abhijeet Kothari

Shreyans Kumar Jain

Vikram Kothekar

Bahauddin Kalyani

Abstract

Big data is a term for data sets that are so large or complex that traditional data processing applications are inadequate to deal with them. Challenges include analysis, capture, data curating, search, sharing, storage, transfer, visualization, querying, and updating and information privacy. Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business.

In this report we propose a Hadoop based platform for **FCI** (Food Corporation of India), which offers information on how much grain can be exported to overseas after storing it for the internal consumption and emergency stock. This platform will take the previous years' data from grain storages, scattered throughout the country and analyse it in various aspects like how much grain was consumed in previous years, how much grain was stored for emergency purpose and which storage exports more grain outside, which will lead to a better and sustainable grain management system for forthcoming years.

Index

Chapter 1: INTRODUCTION

1.1 Introduction	6
1.2 Project Objective	8
1.3 Project Management	8

Chapter 2: PROPOSED SYSTEM

2.1 Proposed System	10
2.2 Purpose of the project	11
2.3 Scope	11

Chapter 3: WORKING ENVIRONMENT

3.1 Linux Operating System	12
3.2Hadoop	13

Chapter 4: MODULE DESCRIPTION

4.1 Basic Mechanism	19
4.2 Factors Affecting The Algorithm	20
4.3 Detailed Mechanism	21
4.4 UML Diagrams	25
4.5 Devised Algorithm	30

Chapter 5: IMPLEMENTATION

5.1 Python Code	33
5.2 Output	34

Chapter 6: PROJECT SETUP

CONCLUSION	35
BIBLIOGRAPHY	39
	41

Chapter 1

INTRODUCTION

1.1 Introduction

Today's Advanced India produces enough grains to not face any kind of grains shortage during tough times & even exports it. According to a report, India's share in World Export of Rice was 17% in 2011. And according to the official data provided on the FCI website the total amount of wheat exported from 2012-16 was approx. 58 Lakh Metric Tons. Despite of this, many states in India still face the food shortage/unavailability problems during non-favourable conditions.

The Food Corporation of India is responsible for the management & distribution of food grains across different parts of the country. Considering the above stated problem, it would be right to consider that the present system used for grain distribution across the country needs more optimization & need to be even more versatile, covering multiple factors & data from the same & previous years, in order to predict the need of grains in a particular area according to various factors such as grain production, rainfall, probability of a disaster, market requirements, exports, etc. Such an environment expects an integration of many technologies to generate results that are required by the user of the system. These results may require a versatile combination of various techniques and workarounds to achieve the designated goals.

The present system for the grain distribution & export management can be optimized a lot using the modern day Big Data technologies such as Hadoop. It can utilize the data from the past & various other factors to get us a more insightful look into the situation at hand & help us in dealing with it. The present system does not gives us any insights as to how the stored grains can be utilized more optimally & transported to other districts/states/UT's to help reduce the food grain shortage problem there, if one arises. So, the system that we are proposing in this paper help us to get more insightful information from the FCI data while also optimizing the distribution system and hence helping in solving the age old problem of food grain shortage in India.

1.2 Project Objectives

This platform will assist FCI (Food Corporation of India) in -

1. Increasing export related transactions.
2. Maintaining satisfactory level of operational and buffer stocks of food grains to ensure National Food Security.
3. Regulate market price to provide food grains to consumers at a reliable price.
4. Effecting price support operations for safeguarding the interests of the farmers.

1.3 Project Management

Management of any project has several steps or processes in it. So, our projects can be described under the following steps:-

Requirement Gathering:

In this phase we gathered the information about necessary equipment's to be used, programming languages, source of input data, similar projects and algorithms.

Planning And Designing:

In this phase we pondered over the problem statement and found out the factors affecting to it and designed an algorithm based on these factors and some other constraints.

Development and Testing:

In this phase a program was developed and individual module testing was performed. The application bugs were identified and removed.

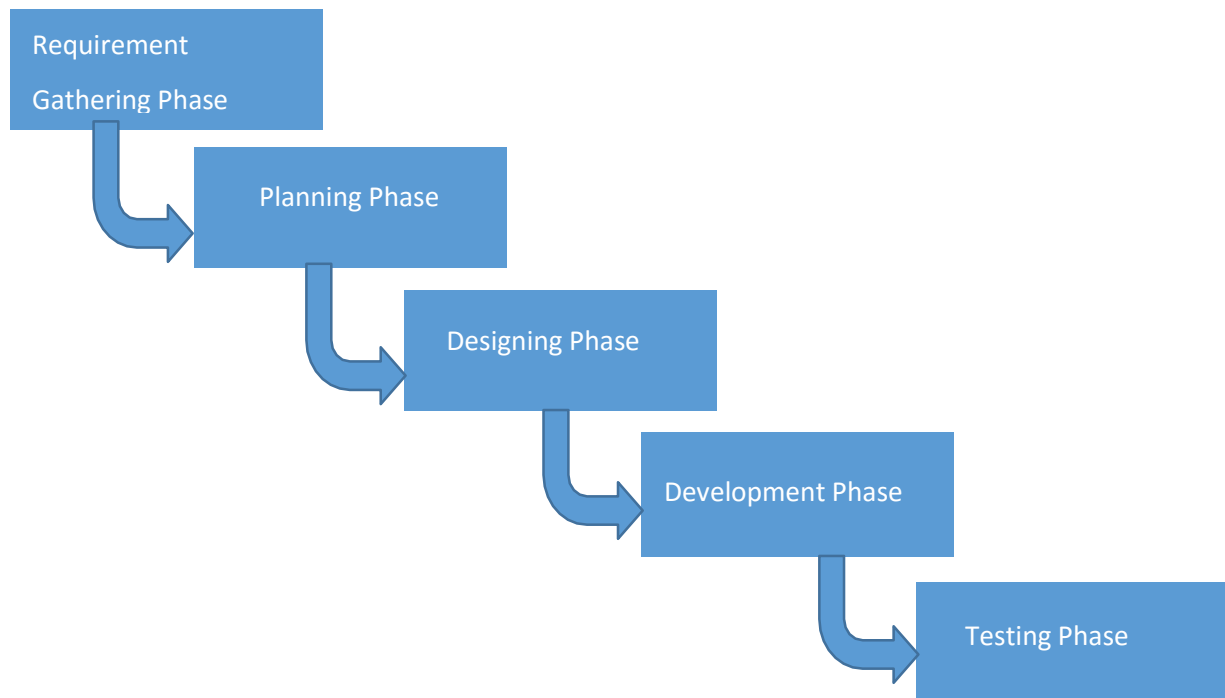


Fig. 1.3 - Project Management Steps

Chapter 2

PROPOSED SYSTEM

2.1 Proposed System:

This system will take raw data from the Disaster Management Cluster, Rainfall Data Cluster and FCI Data Cluster .This data will be analyzed by the devised algorithm resulting into predictions of amount of grains which can be exported in forthcoming years or used at other places where they are needed.

The Main Working of our project is the Algorithm which is going to process Data from all three clusters to dynamically provide us with the data that we are going to use for calculating the total amount of extra grains in the storage. The Algorithm will cover-up various factors to get us the most accurate amount of grains that can be exported or be used at other places where there is a need of extra grains.

2.2 Purpose of the project:

This first of its kind project will be very conducive to boost export related transactions with overseas and also help to equipoise the balance of grains within the country.

2.3 Scope:

It will assist FCI in accelerating export pertaining transactions, providing adequate supply of food grains to the affected area in case of a catastrophe, maintaining satisfactory levels of operational & buffer stock of food grains across the country to support National Food Security, regulating market prices for safe-guarding the interests of farmers. We heartily encourage all such endeavours and would gladly aid – in our capacity – any who require our foresight for their work, now or in the future.

Chapter 3

WORKING ENVIRONMEMENT

3.1 Linux Based Platform:

The following operating systems are supported:

- Linux MINT 18 (Sarah)
- Red Hat Enterprise Linux (RHEL) v5.x or 6.x (64-bit)
- CentOS v5.x or 6.x (64-bit)
- SUSE Linux Enterprise Server (SLES) 11, SP1 (64-bit)

3.2 Hadoop Framework:

Hadoop is comprised of four things:

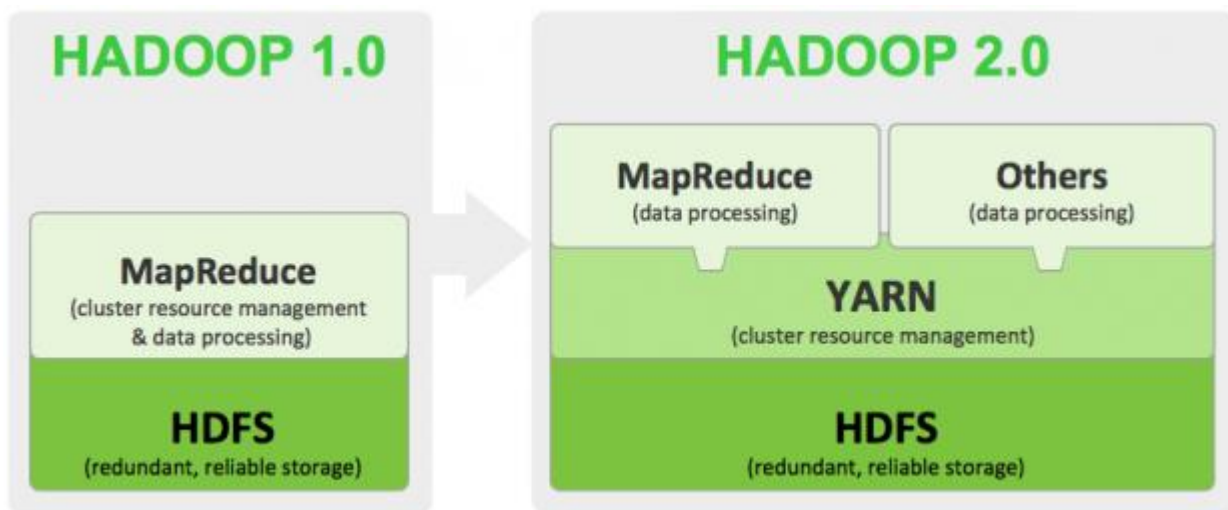


Fig. 3.2.1- Hadoop 1.0 and 2.0 architecture

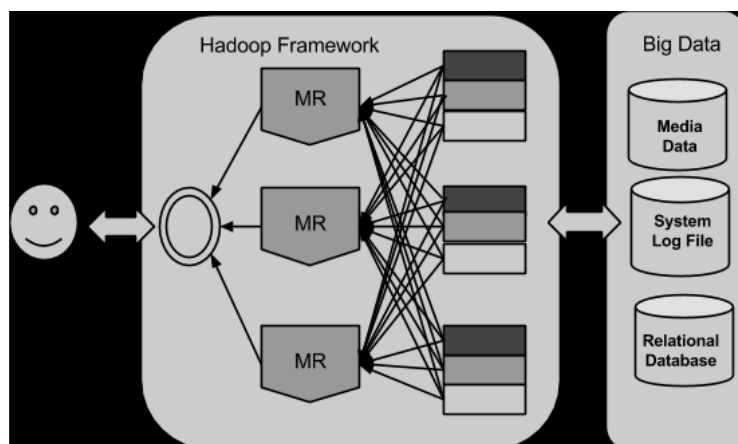
- **Hadoop Common** - A set of common libraries and utilities used by other Hadoop modules.

- **HDFS** - The default storage layer for Hadoop.
- **MapReduce** - Executes a wide range of analytic functions by analysing datasets in parallel before 'reducing' the results. The "Map" job distributes a query to different nodes, and the "Reduce" gathers the results and resolves them into a single value.
- **YARN** - Present in version 2.0 onwards, YARN is the cluster management layer of Hadoop. Prior to 2.0, MapReduce was responsible for cluster management as well as processing. The inclusion of YARN means you can run multiple applications in Hadoop (so you're no longer limited to MapReduce), which all share common cluster management.

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel with others. In short, Hadoop is used to develop applications that could perform complete statistical analysis on huge amounts of data.

MapReduce:

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity



Hadoop Distribute Fig. 3.2.2- MapReduce working

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules:

Hadoop Common: These are Java libraries and utilities required by other Hadoop modules.

Hadoop YARN: This is a framework for job scheduling and cluster resource management.

Working of Hadoop

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs:

1. Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
2. These files are then distributed across various cluster nodes for further processing.

3. HDFS, being on top of the local file system, supervises the processing.
4. Blocks are replicated for handling hardware failure.
4. Checking that the code was executed successfully.

These four components form the basic Hadoop framework. However, a vast array of other components have emerged, aiming to ameliorate Hadoop in some way- whether that be making Hadoop faster, better integrating it with other database solutions or building in new capabilities. Some the more well-known components include:

1. **Spark:**

Used on top of HDFS, Spark promises speeds up to 100 times faster than the two-step MapReduce function in certain applications. Allows data to be loaded in-memory and queried repeatedly, making it particularly apt for machine learning algorithms.

2. **Hive:**

Originally developed by Facebook, Hive is a data warehouse infrastructure built on top of Hadoop. Hive provides a simple, SQL-like language called HiveQL, whilst maintaining full support for MapReduce. This means SQL programmers with little former experience with Hadoop can use the system easier, and provides better integration with certain analytics packages like Tableau. Hive also provides indexes, making querying faster.

3. **HBase:**

Is a NoSQL columnar database which is designed to run on top of HDFS. It is modelled after Google's BigTable and written in Java. It was designed to provide BigTable-like capabilities to Hadoop, such as the columnar data storage model and storage for sparse data.

4. **Flume:**

Flume collects (typically log) data from 'agents' which it then aggregates and moves into Hadoop. In essence, Flume is what takes the data from the

source (say a server or mobile device) and delivers it to Hadoop.

5. **Mahout:**

Mahout is a machine learning library. It collects key algorithms for clustering, classification and collaborative filtering and implements them on top of distributed data systems, like MapReduce. Mahout primarily set out to collect algorithms for implementation on the MapReduce model.

6. **Sqoop:**

Sqoop is a tool which aids in transitioning data from other database systems (such as relational databases) into Hadoop.

Chapter 4

MODULE DESCRIPTION

4.1 Basic Mechanism:

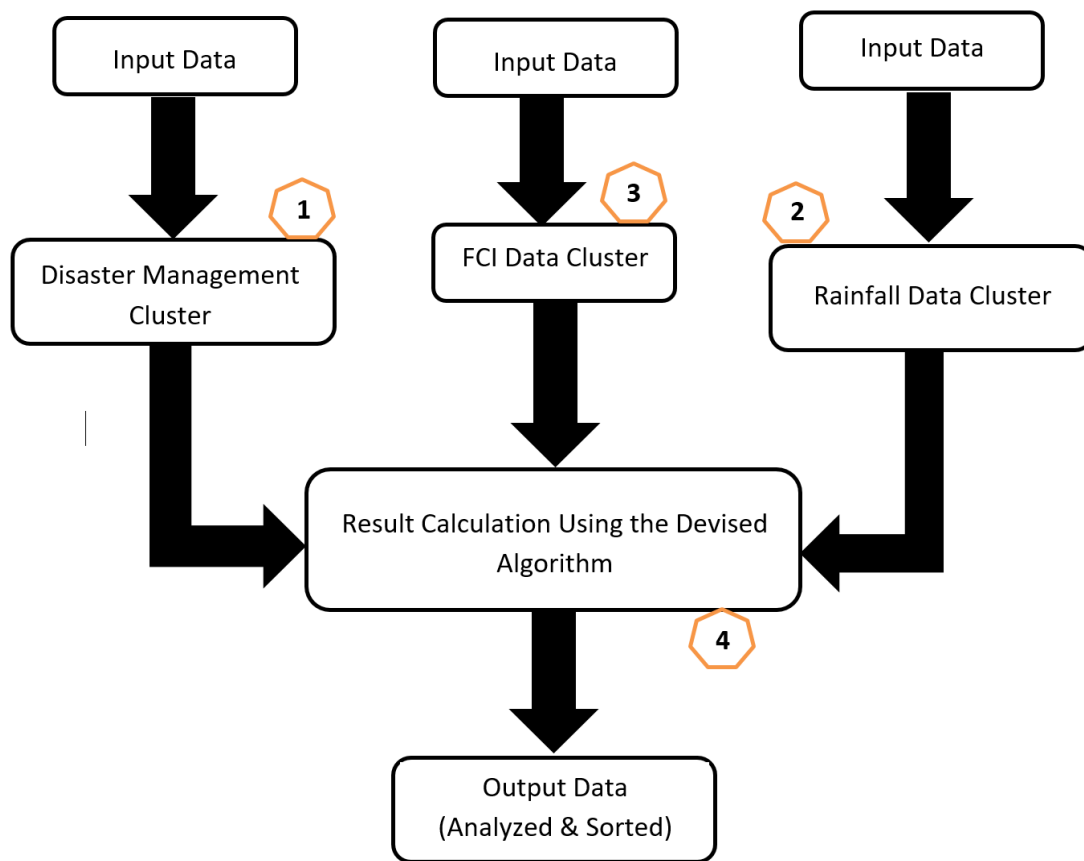


Fig. 4.1- Basic mechanism to get output

This system will take raw data from the Disaster Management Cluster, Rainfall Data Cluster and FCI Data Cluster. In the above figure, [1] represents the Disaster Data Management Cluster, [2] represents the FCI Data Management Cluster, [3] represents the Rainfall Data Management Cluster & [4] represents the master System that will control the working & IO of all the other three clusters. This data will be analysed by the devised algorithm at the master system resulting into predictions of amount of grains which can be exported in forthcoming years or used at other places where they are needed.

The Main Working of our project is the Algorithm which is going to process Data from all three clusters to dynamically provide us with the data that we are going to use for calculating the total amount of extra grains in the storage. The Algorithm

will cover-up various factors to get us the most accurate amount of grains that can be exported or be used at other places where there is a need of extra grains.

4.2 Factors Affecting The Algorithm:

Data sent to Disaster Cluster:

- 1) Zone ID
- 2) Time Period

Data Returned from Disaster Cluster:

- 1) Probability of a Disaster
- 2) Severity of the Disaster (Scale of 1 to 10)

Data sent to Rainfall Data Cluster:

- 1) Zone ID
- 2) Time Period

Data Returned from Rainfall Cluster:

- 1) Rainfall Scale (0 to 10, where 0 is drought & 10 is flood.)
- 2) Probability of the rainfall in the preferred time-period

Data Sent to FCI Cluster:

- 1) Zone ID

Data Returned from FCI Cluster:

- 1) Last year's Total stock of the particular grain
- 2) Last year's Stock left of the particular grain
- 3) Avg. Total stock of the particular grain in last years
- 4) Avg. Stock left of the particular grain in last year
- 5) Avg. Amount of grains exported in last years
- 6) Last year's Total Amount of grains exported
- 7) Avg. Amount of grains kept in Emergency Stock
- 8) Last year's Emergency Stock
- 9) Grains used for Govt. purposes

4.3 Detailed Mechanism:

4.3.1 Disaster Risk Data Cluster:

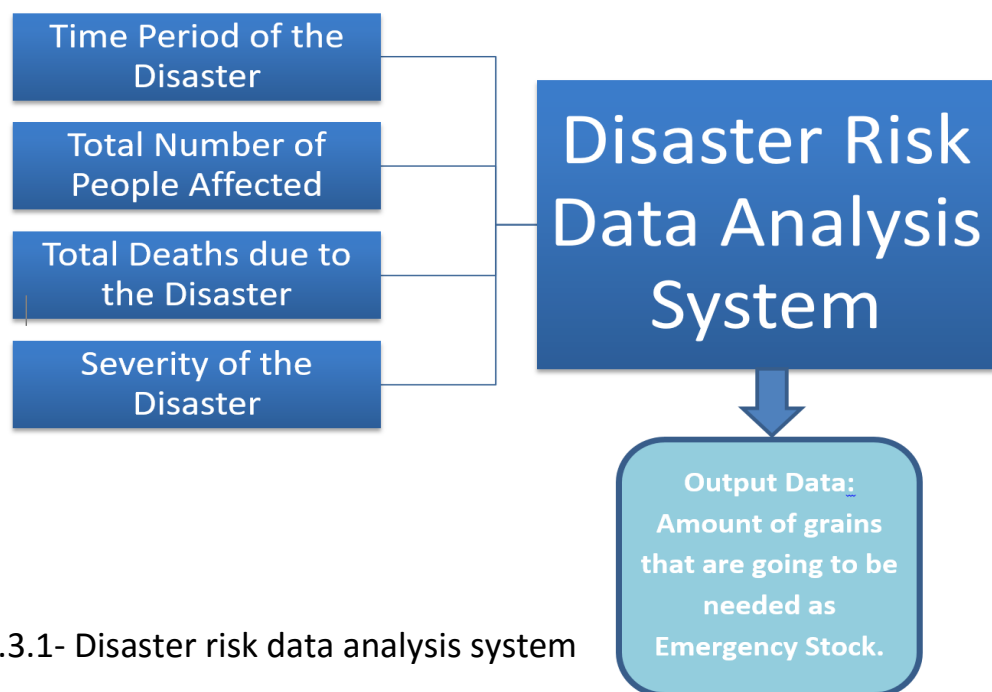


Fig. 4.3.1- Disaster risk data analysis system

This cluster will help us in calculating the total amount of Emergency Stock of grains that is needed in order to ensure the proper supply of grains the whole year.

The Cluster will be using data in the categories of :

- a) time period of the disaster,
- b) total number of people affected during the disaster including deaths,
- c) severity of the disaster,
- d) total destruction in all the categories including vegetation, crops, people, etc..

This data when analysed with a proper algorithm will provide us with the total amount of grain stock needed in case of a natural disaster or calamity & what's the probability of it during a particular period of time.

4.3.2 Rainfall Data Cluster:

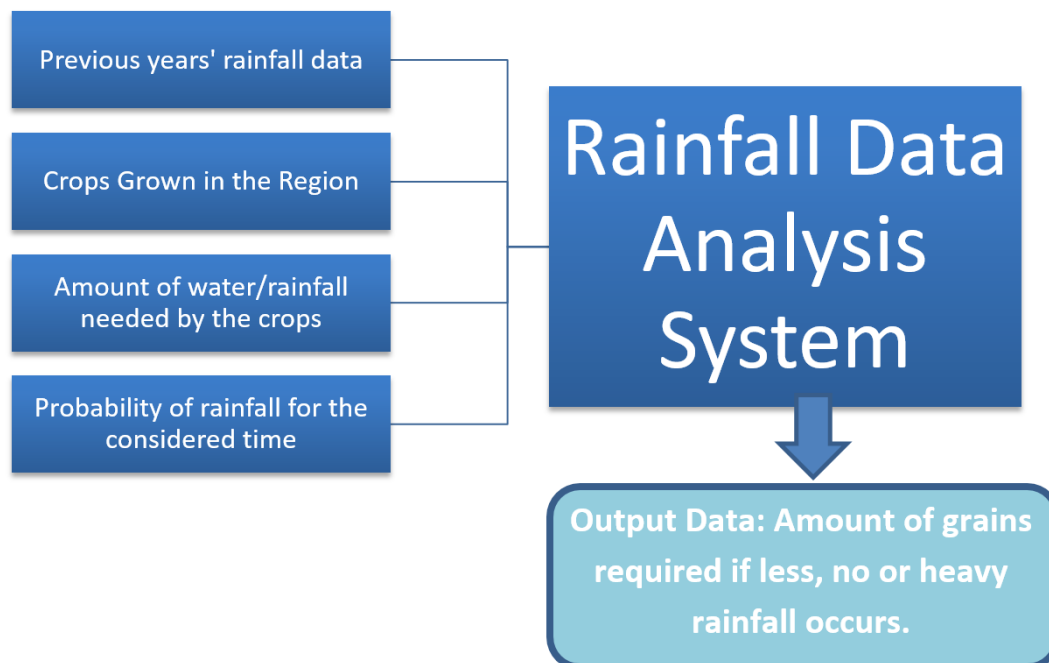


Fig. 4.3.2- Rainfall data analysis system

This cluster will use the data from the Weather forecasts of previous years for a particular area & the rainfall prediction for that area for that particular period of time. It will also consider the types of grains sown during that time & amount of water required by them. By analysing all this data, it will output us the quantity of the extra grains that might be required in case of less or no rainfall for that period of time. It will output a negative value for less, no or heavy rainfall prediction & a positive value for proper and required amount of rains.

4.3.3 FCI Data Cluster:

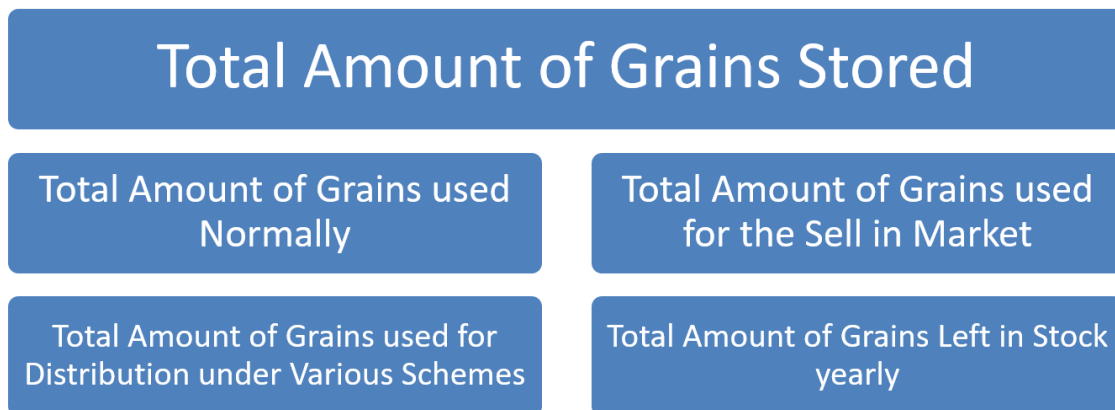


Fig. 4.3.3- FCI data cluster

This cluster will be responsible for the processing of all the data concerning the input & output of grains in the storage go downs, the amount of grains that are used back in the market & Govt. shops & by the distributors. This cluster will provide this data to the main Data Analysis Algorithm to process up this data with the data from the other two clusters to provide us with the output that we require from this system.

4.3.4 Data Analysis Algorithm:

This Algorithm will be responsible for the overall prediction of the data using the data that it got as an input from both the Disaster Risk Data Cluster & Rainfall Data Cluster. It will then use this data with the data from the FCI cluster to calculate the amount of grains that are stored in the godowns and are extra. Using this data, the grains which are of no use in that particular area can be used for various purposes, like:

- a) for providing proper food in areas with less, no or heavy rainfall,
- b) for providing proper food supply to areas affected with a natural calamity like floods,
- c) for export to other countries & foreign markets, to increase the economic growth of the country.

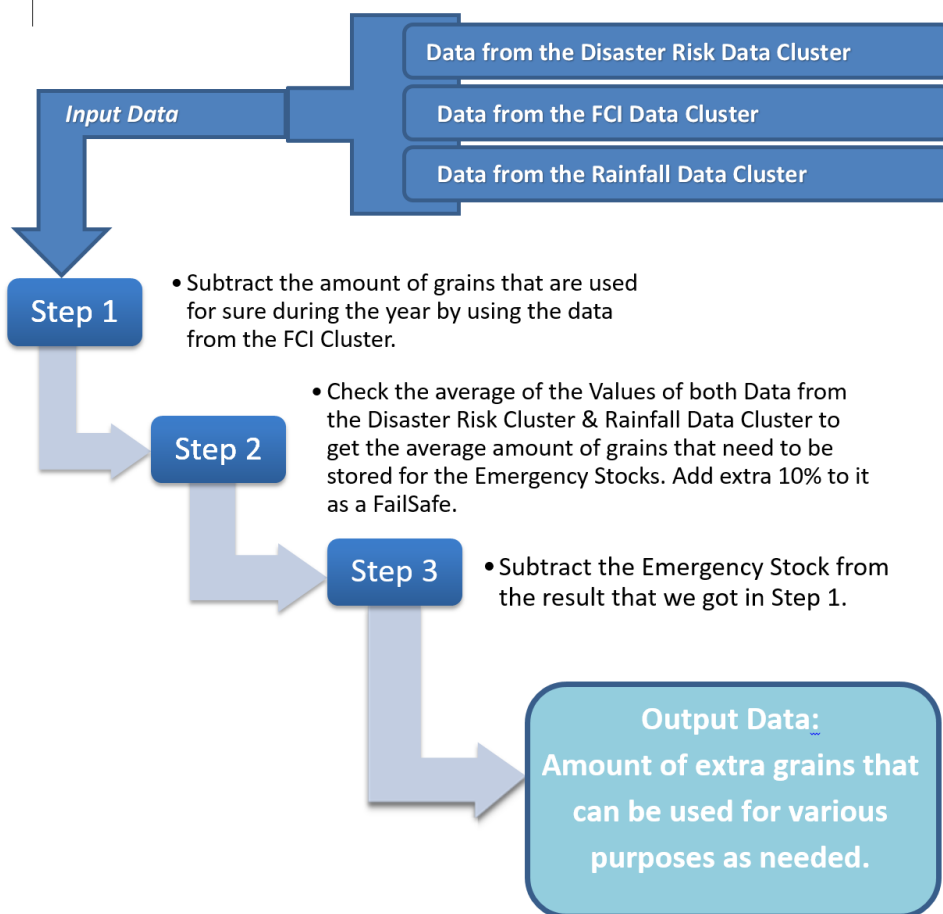
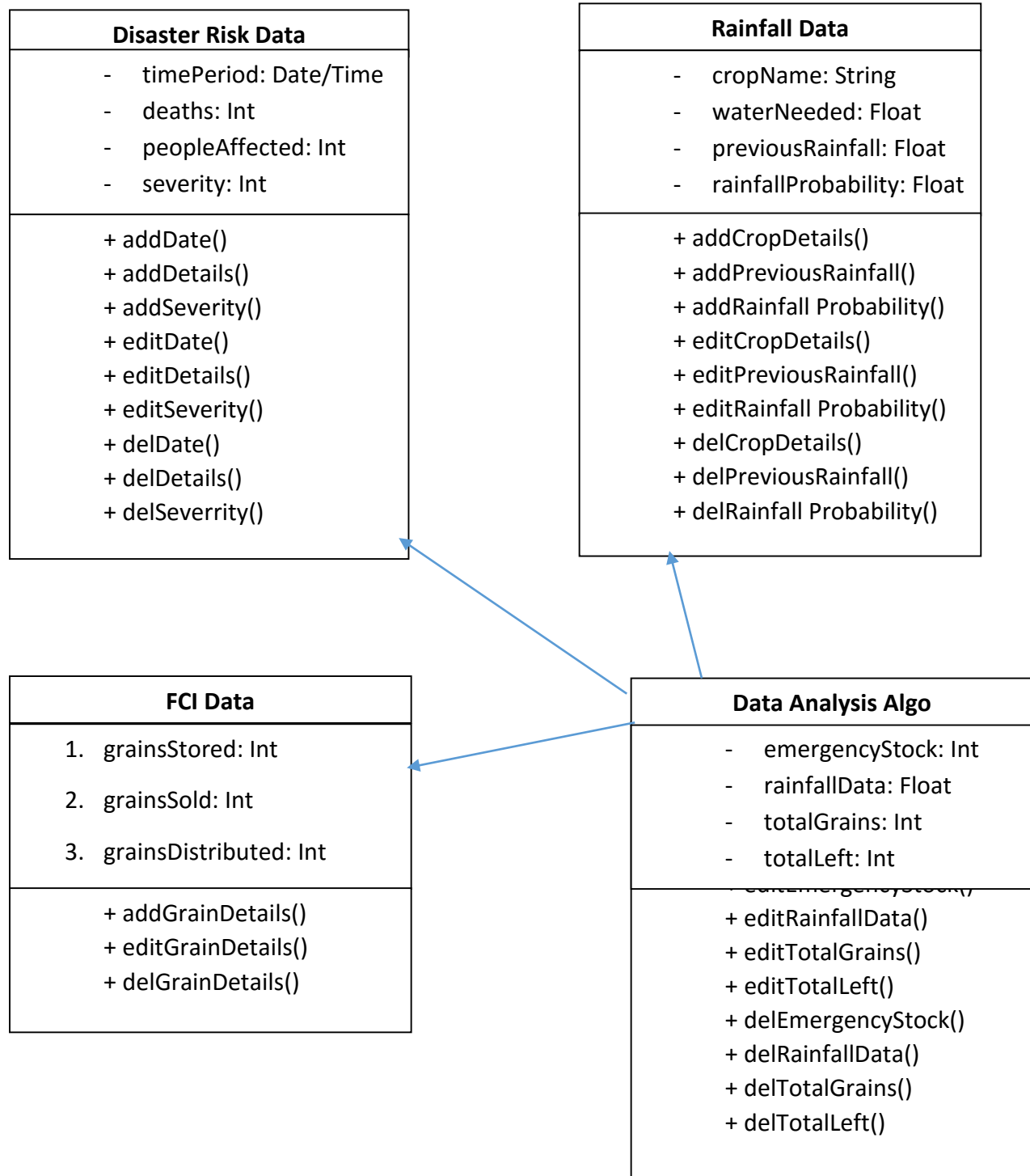


Fig. 4.3.4- Data analysis algorithm

4.4 UML Diagrams:

4.4.1 Class Diagrams



4.4.2 Object Diagram:

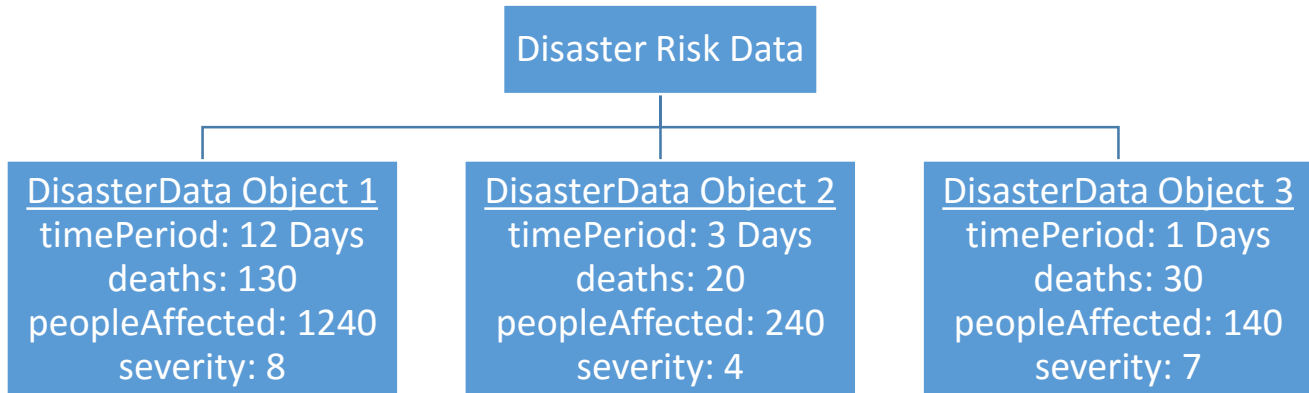


Fig. 4.4.2.1- Disaster Risk Data Object Diagram

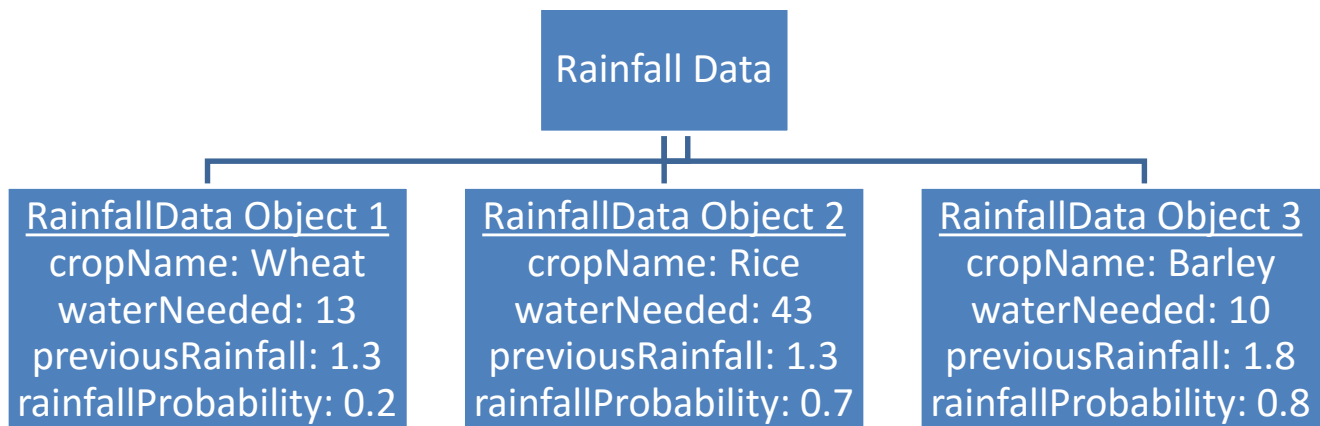


Fig. 4.4.2.2- Rainfall Data Object Diagram

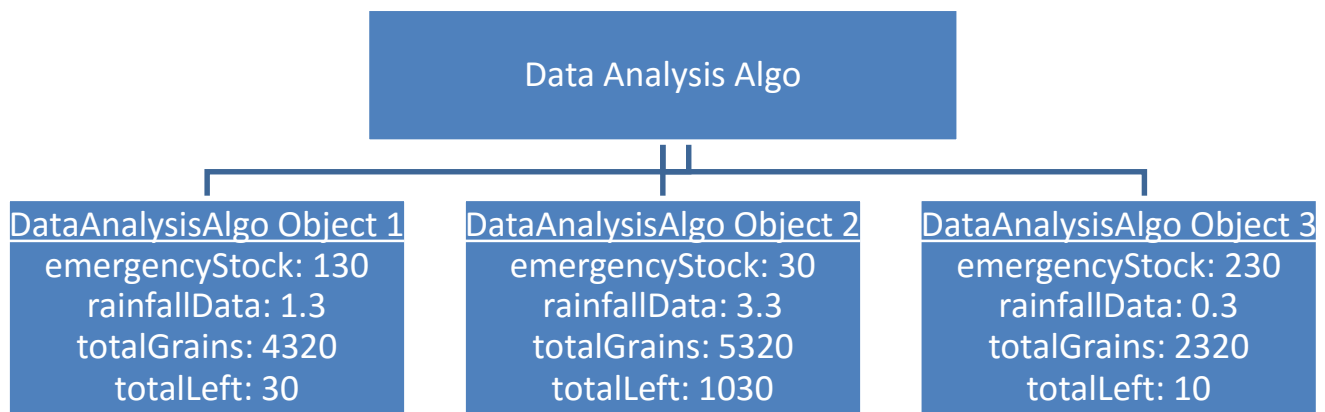


Fig. 4.4.2.3- Data Analysis Algorithm Object Diagram

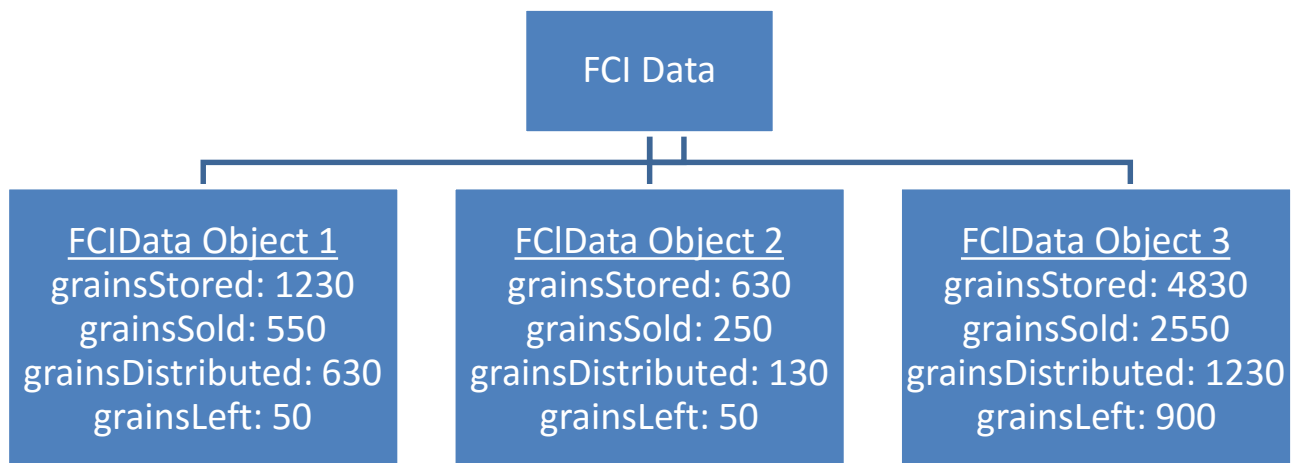
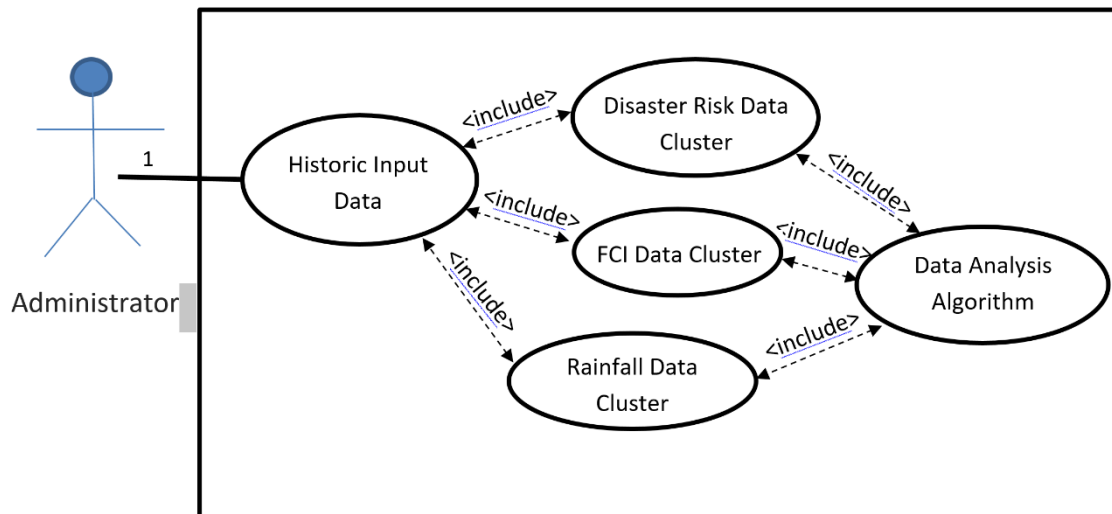
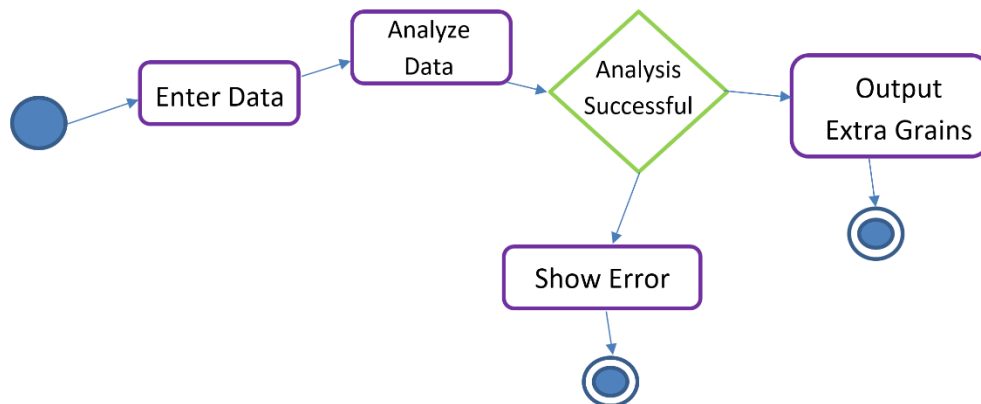


Fig. 4.4.2.4- FCI Data Object Diagram

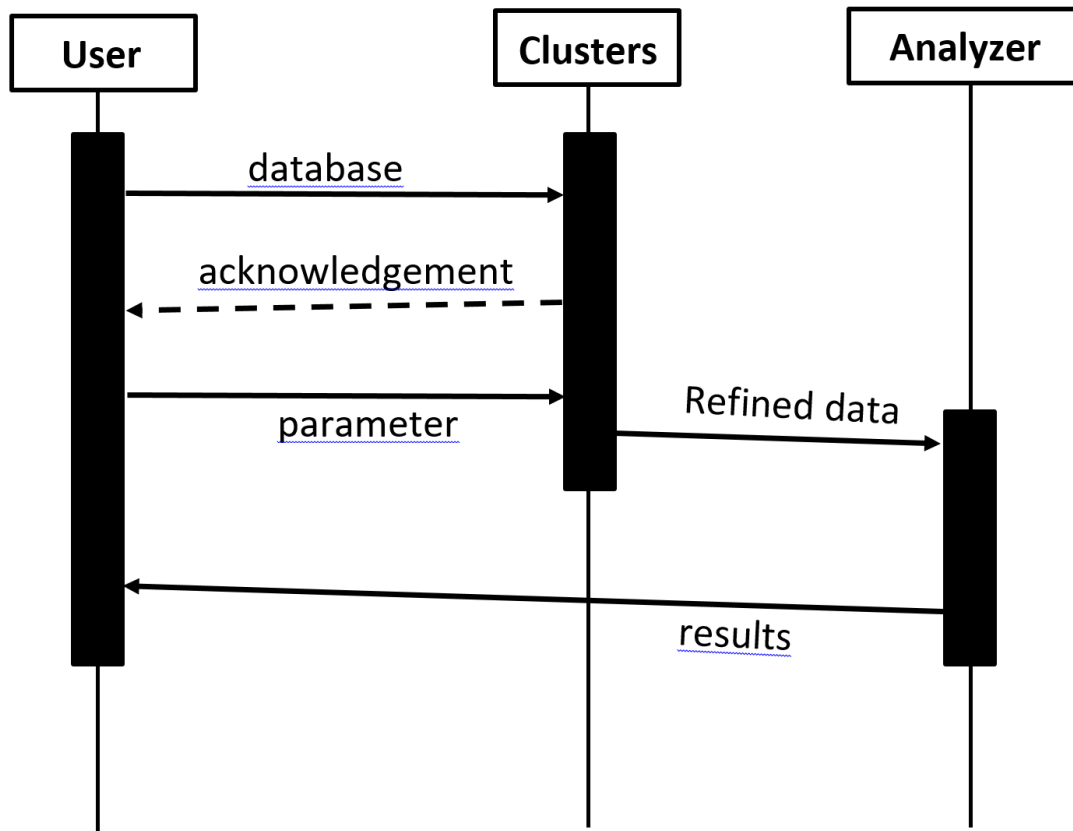
4.4.3 Use Case Diagram



4.4.4 Activity Diagram:



4.4.5 Sequence Diagram:



4.5 Devised Algorithm:

```
var total_stock = get(fci_cluster);
```

```
var base_em_stock = 15 * total_stock / 100 ;
```

```
var avg_years_em_stock = get(fci_cluster);
```

```
var last_em_stock = get(fci_cluster);
```

```
var avg_stock = ( avg_years_em_stock + last_em_stock ) / 2 ;
```

```
// if production is less this year, add extra 5%.
```

```
if ( base_em_stock < last_em_stock ) {
```

```
    base_em_stock = last_em_stock + ( 5 * total_stock / 100 );
```

```
}
```

```
// if production is proper this year, minify it.
```

```
if ( base_em_stock >= last_em_stock ) {
```

```
    base_em_stock = avg_stock;
```

```
}
```

```
// including disaster factors
```

```
var disaster_prob = get(disaster_cluster);
```

```
var disaster_severity = get(disaster_cluster);
```

```
if ( disaster_prob > 0.5 ){
```

```
        base_em_stock = last_em_stock + ( (5 * (ln disaster_severity)) * total_stock  
/ 100 );  
    }
```

```
// including rainfall factors
```

```
var rainfall_prob = get(rainfall_cluster);
```

```
var rainfall_severity = get(rainfall_cluster);
```

```
if ( !( rainfall_severity > 3.5 && rainfall_severity < 8 ) )    // Rain disaster
```

```
        base_em_stock = last_em_stock + ( (5 * (ln rainfall_severity)) * total_stock  
/ 100 );  
    }
```

```
// Including self-country usages
```

```
var govt_grains = get(fci_cluster);
```

```
total_stock -= govt_grains;
```

```
var export;
```

```
export = total_stock - base_em_stock;
```

```
console.log(export);
```

Chapter 5

IMPLEMENTATION

5.1 Python Code:

```
import json;

fci_file = open("FCI_Data.json").read()
fci_data = json.loads(fci_file)
disaster_file = open("Disaster_Data.json").read()
disaster_data = json.loads(disaster_file)
rainfall_file = open("Rainfall_Data.json").read()
rainfall_data = json.loads(rainfall_file)
current_stock = fci_data[0]["current_total_stock"]
base_emergency_stock = 0.15 * current_stock
avrg_em_stock = fci_data[0]["avrg_grain_emergency_stock"]
last_yr_em_stock = fci_data[0]["last_yr_grain_emergency_stock"]
avg_stock = (avrg_em_stock + last_yr_em_stock) / 2;

if(base_emergency_stock < last_yr_em_stock):
    base_emergency_stock = last_yr_em_stock + ( 5 *
current_stock / 100 )

if(base_emergency_stock >= last_yr_em_stock):
    base_emergency_stock = avrg_em_stock
dis_prob = disaster_data[0]["Probability"]
dis_severity = disaster_data[0]["Severity"]
```

```

if(dis_prob > 0.5):
    base_emergency_stock = last_yr_em_stock + ( 5 * (
dis_severity * current_stock / 100 ))
rainfall_prob = rainfall_data[0]["Probability"]
rainfall_scale = rainfall_data[0]["Scale"]
if(rainfall_prob > 0.5):
    if (not(rainfall_scale > 3.5 and rainfall_scale < 8.9)) :
        base_emergency_stock = last_yr_em_stock + (5 *
(rainfall_scale * current_stock / 100 ))
gov_stock = fci_data[0]["grain_used_by_gov"]
current_stock = current_stock - gov_stock
export = current_stock - base_emergency_stock
print ("grains to be exported")
print (export)

```

5.1 Python Code:

```

>> Grains to be exported:
752915.725

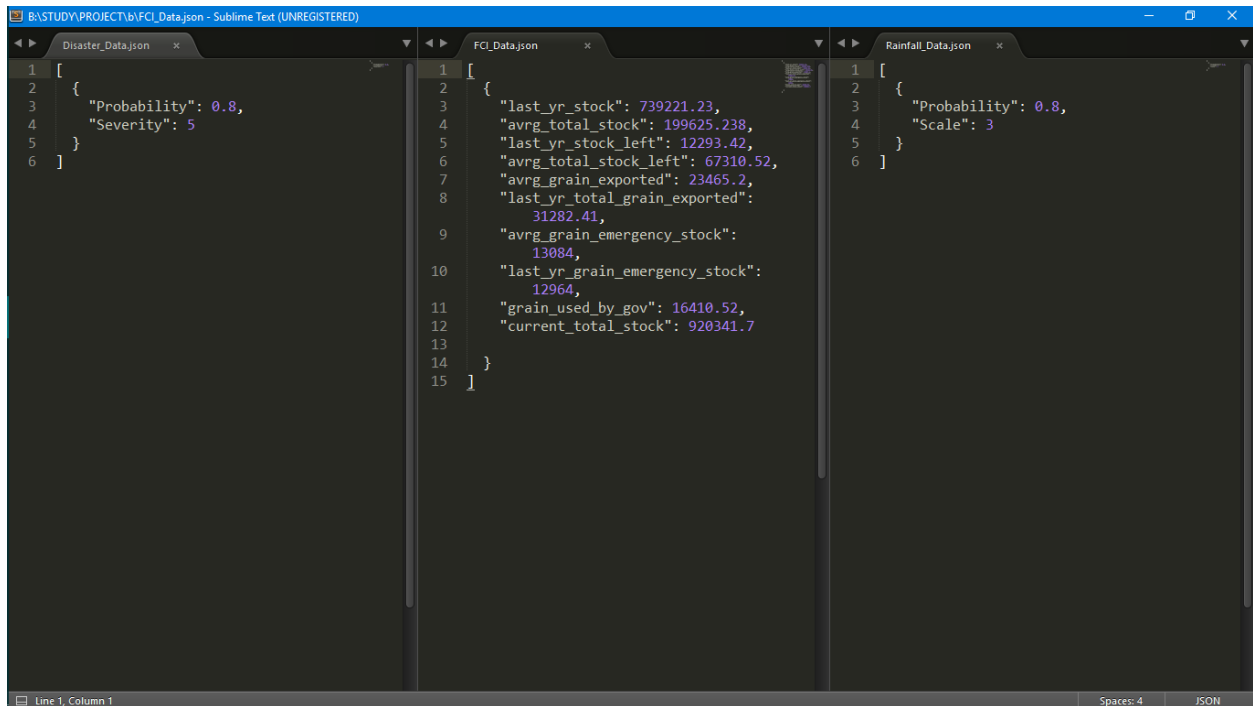
```

Chapter 6

PROJECT SETUP

```
B:\STUDY\PROJECT\b\algo.py - Sublime Text (UNREGISTERED)
1 import json;
2
3 fci_file = open("FCI_Data.json").read()
4 fci_data = json.loads(fci_file)
5
6 disaster_file = open("Disaster_Data.json").read()
7 disaster_data = json.loads(disaster_file)
8
9 rainfall_file = open("Rainfall_Data.json").read()
10 rainfall_data = json.loads(rainfall_file)
11
12
13
14 current_stock = fci_data[0]["current_total_stock"]
15
16 base_emergency_stock = 0.15 * current_stock
17
18 avrg_em_stock = fci_data[0]["avrg_grain_emergency_stock"]
19
20 last_yr_em_stock = fci_data[0]["last_yr_grain_emergency_stock"]
21
22 avg_stock = (avrg_em_stock + last_yr_em_stock) / 2;
23
24 if(base_emergency_stock < last_yr_em_stock):
25     base_emergency_stock = last_yr_em_stock + ( 5 * current_stock /
26         100 )
27
28 if(base_emergency_stock >= last_yr_em_stock):
29     base_emergency_stock = avrg_em_stock
30
31
32 dis_prob = disaster_data[0]["Probability"]
33 dis_severity = disaster_data[0]["Severity"]
34
35 if(dis_prob > 0.5):
36     base_emergency_stock = last_yr_em_stock + ( 5 * ( dis_severity *
37         current_stock / 100 ))
38
39 rainfall_prob = rainfall_data[0]["Probability"]
40 rainfall_scale = rainfall_data[0]["Scale"]
41
42 if(rainfall_prob > 0.5):
43     if (not(rainfall_scale > 3.5 and rainfall_scale < 8.9)) :
44         base_emergency_stock = last_yr_em_stock + (5 * (rainfall_scale
45             * current_stock / 100 ))
46
47 gov_stock = fci_data[0]["grain_used_by_gov"]
48
49 current_stock = current_stock - gov_stock
50
51 export = current_stock - base_emergency_stock
52
53 print ("grains to be exported")
54 print (export)
```

Fig 6.1-
Code
snippet



The screenshot shows a Sublime Text editor window with three tabs open: Disaster_Data.json, FCI_Data.json, and Rainfall_Data.json. The Disaster_Data.json file contains a single JSON object with "Probability": 0.8 and "Severity": 5. The FCI_Data.json file contains a single JSON object with multiple fields including "last_yr_stock", "avrg_total_stock", "last_yr_stock_left", "avrg_total_stock_left", "avrg_grain_exported", "last_yr_total_grain_exported", "avrg_grain_emergency_stock", "last_yr_grain_emergency_stock", "grain_used_by_gov", and "current_total_stock". The Rainfall_Data.json file contains a single JSON object with "Probability": 0.8 and "Scale": 3. The status bar at the bottom indicates "Line 1, Column 1", "Spaces: 4", and "JSON".

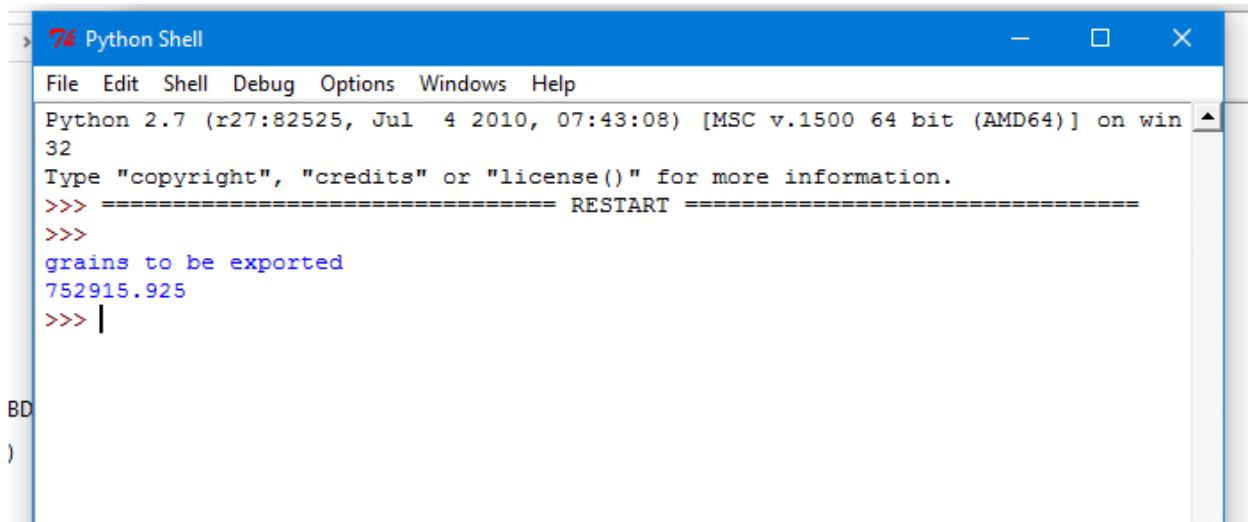
```
1 [
2   {
3     "Probability": 0.8,
4     "Severity": 5
5   }
6 ]
```

```
1 [
2   {
3     "last_yr_stock": 739221.23,
4     "avrg_total_stock": 199625.238,
5     "last_yr_stock_left": 12293.42,
6     "avrg_total_stock_left": 67310.52,
7     "avrg_grain_exported": 23465.2,
8     "last_yr_total_grain_exported":
9       31282.41,
10    "avrg_grain_emergency_stock":
11      13084,
12    "last_yr_grain_emergency_stock":
13      12964,
14    "grain_used_by_gov": 16410.52,
15    "current_total_stock": 920341.7
16  }
17 ]
```

```
1 [
2   {
3     "Probability": 0.8,
4     "Scale": 3
5   }
6 ]
```

Fig 6.1 – JSON input file code snippet

Output:

A screenshot of a Python Shell window. The title bar is blue with the text 'Python Shell' and standard window controls. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The main text area shows the following content: 'Python 2.7 (r27:82525, Jul 4 2010, 07:43:08) [MSC v.1500 64 bit (AMD64)] on win 32', 'Type "copyright", "credits" or "license()" for more information.', a prompt '>>>' followed by a line of equals signs and the word 'RESTART' in all caps, another prompt '>>>', a line of blue text 'grains to be exported', a line of blue text '752915.925', and a final prompt '>>>' with a vertical cursor. To the left of the shell window, the text 'BD' and a closing parenthesis ')' are partially visible.

```
Python 2.7 (r27:82525, Jul 4 2010, 07:43:08) [MSC v.1500 64 bit (AMD64)] on win 32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
grains to be exported
752915.925
>>> |
```

Fig 6.2- Code output snippet

CONCLUSION

Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business. This big data technology based project will be very facilitative for **FCI** (Food Corporation of India) and will assist to enhance export pertaining transaction and business with overseas markets. It will obviate any hasty actions during any calamities and will balance the total amount of grains in the country. This system can be improvised by providing more accurate data and using Hadoop extensions like Spark, Hive etc.

Thus, we see that Big Data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business. This big data technology based project will be very facilitative for FCI (Food Corporation of India) and will assist to enhance export pertaining transaction and business with overseas markets. It will obviate any hasty actions during any calamities and will balance the total amount of grains in the country.

This first of its kind project will be very conducive to boost export related transactions with overseas and also help to equipoise the balance of grains within the country. It will assist FCI in accelerating export pertaining transactions, providing adequate supply of food grains to the affected area in case of a catastrophe, maintaining satisfactory levels of operational & buffer stock of food grains across the country to support National Food Security, regulating market prices for safe-guarding the interests of farmers. We heartily encourage all such endeavours and would gladly aid – in our capacity – any who require our foresight for their work, now or in the future.

BIBLIOGRAPHY

- [1] [Food Corporation of India - Wikipedia](#)
- [2] [Imports & Exports - Food Corporation of India](#)
- [3] Commonwealth Secretariat, ["Grain Crops"](#)
- [4] [Understanding Hadoop Clusters And The Network](#)

- [5] Big Data-Naren Ramakrishnan; Ravi Kumar (Computer Science & Engineering)
- [6] A Review Paper on Map Reducing Using Hadoop
Ms Tarunpreet Chawla, Mr Lalit, Dept. of CSE, Vivekananda Institute of Technology, Jaipur
- [7] Design of Self-Adjusting algorithm for data-intensive MapReduce Applications
Mr Amin Nazir Nagiwale, Mr Manish R. Umale, Computer Engineering, Lokamanya Tilak College of Engineering, Mumbai
- [8] SigCO: Mining Significant Correlations via a Distributed Real-time Computation Engine
Tian Guo, EPFL, Lausanne, Switzerland
- [9] Accelerating Apache Spark Big Data Analysis with FPGAs
Ehsan Ghasemi, Paul Chow, Dept. of Electrical and Computer Engineering, University of Toronto
- [10] A Review Paper on Map Reducing Using Hadoop
Ms. Tarunpreet Chawla, Mr. Lalit, Dept. of CSE, Vivekananda Institute of Technology, Jaipur
- [11] Hadoop, MapReduce and HDFS: A Developers Perspective
Mohd Rehan Ghazia, Durgaprasad Gangodkara, Graphic Era University, Clement town

- [12] Hadoop Distributed File System (HDFS)
Thomas Kiencke Institute of Telematics, University of Lubeck, Germany
- [13] Big Data and Hadoop-A Study in Security Perspective
B. Saraladevia, N. Pazhanirajaa, P. Victor Paula, M.S. Saleem Bashab, P. Dhavachelvanc
- [14] Toward Scalable Systems for Big Data Analytics:
A Technology Tutorial Han Hu, Yonggang Wen, Tat-Seng Chua, Xuelong Li
- [15] Serialized Optimization of Supply Chain Model Using
Genetic Algorithm and Geometric Predictions
- Prakhar Sharma, Umarfaruk Ansari, Jonathan Lobo