PROJECT REPORT

on

WORLD FOOD SUPPLY AND WASTE ANALYSIS

for

Digipodium

towards partial fulfillment of the requirement
for the award of degree of

Bachelor of Computer Applications

from

Babu Banarasi Das University
Lucknow

Developed and Submitted by
1190211034
Akash Kumar Yadav

Under Guidance of
Mohammad Mubassir

Academic Session 2021 - 22
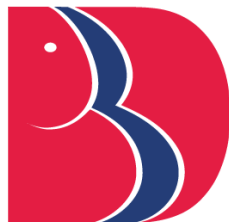## School of Computer Applications

CERTIFICATE

This is to certify that Project Report entitled

WORLD FOOD SUPPLY AND WASTE ANALYSIS

being submitted by

Akash Kumar Yadav

towards the partial fulfillment of the requirement
for the award of the degree of

Bachelor of Computer Applications
to
Babu Banarasi Das University
Lucknow

in the Academic Year 2021-22
is a record of the student's own work carried out at

Digipodium

and to the best of our knowledge the work reported herein does not form a part of any other thesis or work on the basis of which degree or award was conferred on an earlier occasion to this or any other candidate.

Dr. Prabhash Chandra Pathak
HEAD (School of Computer Applications)

Certificate

This is to certify that **Akash Kumar Yadav** has successfully completed the project titled **"World Food Supply & Waste Analysis"** as part of the internship program in our organization.

The project using Data Analytics was done under the guidance and supervision of Mohammad Mubassir from Jan'22 – May'22.

The student has completed the assigned project well within the time frame and the performance and conduct during the project was found good.

A technically sound project has been developed for one of our clients.

Regards,

Director
Digipodium
9415082377

ABSTRACT

Welcome to 'World Food Supply and Waste Analysis'!

'World Food Supply and Waste Analysis' is dynamic and as classified as its name. It is a tool to make your research to start and end at the same place.

Whether to look at the supply trends or the production or the waste, its all in the same dashboard. A real convenience for real researchers. The dashboard provides in depth analysis, like, category division, country divisions and more.

The dashboard is currently public and any interested, curious viewer can view it and explore.

DECLARATION

I, **Akash Kumar Yadav**, hereby declare that this project entitled **'World Food Supply and Waste Analysis'** and the work described in it is my own work, and are carried out in accordance with the regulations of the **Babu Banarasi Das University**.

I declare that the work has been taken out with full dedication. I have taken references of the various text and no part of this project has been submitted for any other degree.

Akash Kumar Yadav

Date: 3 June 2022

BCA 6th Semester

1190211034

ACKNOWLEDGEMENT

TABLE OF CONTENTS

1. <u>INTRODUCTION</u>

1.1. ***INTRODUCTION OF THE PROJECT***

'To waste food is to throw existence out of existence.'

From the early existence of life, food has always been the primary need for human existence. We live, die, fight sometimes kill for it. From Plants to us Humans, we all are dependent on food for survival. Some of us produce it, some of us consume it. Though food – a basic need; exists in an unbalanced state. Some regions produce more than sufficient quantities while others face food scarcity. For example, if food is lost or wasted in the production, storage, or transportation stages, it **reduces food availability and increases the vulnerability of our food system**. Furthermore, food loss also constitutes a waste of other resources used to produce this food, namely water, land, energy, labor, and capital. Different countries consume different food quantities. Roughly one-third of the food produced in the world for human consumption every year - approximately 1.3 billion tones - gets lost or wasted.

This food wastage also negatively impacts the environment in numerous ways. The food dumped causes a foul smell, food-rotting, a breeding ground for bacteria, occupy more land portion and more environment degrading activities. Reducing food waste can also address major climate changes.

Some quick facts about 'food supply and wastage' globally:

- Food wastage increases the presence of greenhouse gases.
- Food Wastage is worse than flying emissions (1.9%), plastic production.
- 1 in 9 people is under-nourished which takes a total to 793 million approximately.
- If one lost Quarter of food can be recovered, it can feed 870 million hungry humans.
- Half of the fruits and vegetables produced are wasted.
- A Lettuce head takes 25 years to decompose in landfill.

This project titled *'World Food Supply and Waste Analysis'* focuses on the data obtained about Food Supply and Food Waste globally to realize the quantity produced and wasted and provide transparent export of over-produced food. The project aims to analyze this informational data and

help us to determine the trends and overall quantity measures that will help us feed the current malnourished population, during difficult times, and the future generation rather than waste it.

1.2. *OBJECTIVE & SCOPE*

The **objective** of this project is to provide trusted data to help us consider the situations like pandemics and support food needs in future.

- *Collecting the data for production and waste globally:*
  The primary objective is to consider the data figures considering production values and wastage amounts and therefore, obtaining the quantities for the same.

- *Determining the reasons for the wastage:*
  Considering the overall reasons for the food wastage and determining the majors and minors of them.
  - Lack of appropriate Storage.
  - Unchecked quantities of purchase.
  - Over-preparation of food in weddings, restaurants, etc.
  - Food Spoilage

- *Export:*
  If the food is over-produced, it will be taken into account for the immediate action of exportation to other regions which witness lesser production.

Our analysis will provide the most trusted and honest data gathered globally to ensure easy proportionate studies about the food quantity.

Following the current prevailing conditions which include COVID-19 that has been in since 2019 and yet not completely vanished, the **scope** for our project is ever-growing.
- Analyzing region for immediate food needs.
- Pandemic or Natural Disaster Relief.

2. <u>SYSTEM ANALYSIS</u>

System Analysis is the study of sets of interacting entities, including computer analysis.

This field is closely related to requirement analysis or operations research. It is also an explicit formal enquiry carried out to help someone to identify a better course of action and make a better decision than he might otherwise have made.

System analysis can also be defined as a series of components that perform an organic function together.

2.1. *IDENTIFICATION OF NEED*

Living on a vast planet with great diversities also calls for many responsibilities. With every small or big platforms, none is unified to present a permanent display of truthful knowledge.

Therefore, rises a need of innovation and enhancement that derives from the roots of pre-existing origin to find its own origin.

The project uses all the data provided from all over the world to present a unified dashboard to help future calculations and predictions.

2.2. *PRELIMINARY INVESTIGATION*

PYTHON

With Python, you can basically do anything and everything. Due to its vast range of ever-present libraries, you can do most of the tasks. It is free and open source, that is, ridiculously fast.

Python was developed to put emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

1. *Improved Productivity*

   Python is a very productive language. Due to the simplicity of python, developers can focus on solving the problem. They don't need to spend too much time in understanding the syntax of the language. You write less code and get more things done.

2. *Free and Open Source*

   Python comes under the OSI approved open-source license. This makes it free to use and distribute. You can download the source code, modify it and even distribute your version of python. This is useful for organizations that want to modify some specific behavior and use their version for development.

3. *Vast Libraries Support*

   The standard library of python is huge, you can find almost all the functions needed for your tasks. So, you don't have to depend on external libraries. But even if you do, a Python package manager (pip) makes things easier to import other packages from the python package index (Pypi). It consists of over 200,000 packages.

VISUAL STUDIO CODE

Visual Studio Code is a free source code editor developed my Microsoft. It feels much more lightweight than traditional IDEs, yet its extensions make it versatile enough to handle just about any type of development work, including Python and the Dash web framework.

PLOTLY DASH

Dash is an open source library released under the permissive MIT license. Plotly develops Dash and also offers a platform for writing and deploying Dash apps in an enterprise environment. Dash is a python framework created by plotly for creating interactive web applications.

PANDAS

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for

manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas isfast and it has high performance & productivity for users.

2.3. *FEASIBILITY STUDY*

Feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In tis simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

A well-designed feasibility study should provide historical background of the business or project, a description of the product or service, accounting statement, details of the operational and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation.

A feasibility study evaluates the project's potential for success: therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions.

It must therefore, be conducted with an objective, unbiased approach to provide information upon which decisions can be based.

Investment proposals, involving huge capital outlay are invariably irreversible. Therefore, before starting a project/proposal, it is necessary and imperative to find out whether the same is feasible or not. The feasibility report of the project holds the advantages and flexibility of the project.

This is divided into:
1. Economic Feasibility.
2. Technical Feasibility.
3. Behavioral Feasibility.
4. Operational Feasibility.

   1. Economic Feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of the candidate system. More commonly known as cost/benefit analysis, the procedure is to be determining the benefits and savings that are expected from a candidate and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system.

A systems financial benefit must exceed the cost of developing that system. i.e. a new system being developed should be a good investment for the organization. Economic feasibility considers the following:

    a. The cost to conduct a full system investigation.

    b. The cost of hardware and software for the class of application.

    c. The benefits in the form of reduced cost or fewer costly errors.

2. *Technical Feasibility*

Technical feasibility centers on the existing computer system (Hardware and Software etc.) and to what extend it support the proposed addition. In this project, all the necessary cautions have

been taken care to make it technically feasible. Using a key the display of text/object is very fast. Also, the tools, operating system and programming language used in this localization process is compatible with the existing one. The technical needs of the system vary considerably but might include:

    a. The facility to produce outputs in a given time.

    b. Response time under certain conditions.

The project is technical feasible because of the availability of the required software hardware and technology. The changes can be made be made in the system as and when required.

3. *Behavioral Feasibility*

People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development of a computerized system. Therefore, it is

understandable that the introduction of a candidate system requires special efforts to educate and train the staff. The software that is being developed is user friendly and easy to learn. In this way, the developed software is truly efficient and can work on any circumstances, tradition, locales. Behavioral study strives on ensuring that the equilibrium of the organization and status quo in the organization neither are nor disturbed and changes are readily accepted by the users. Thus, these factors are considered for a Behavioral feasibility study:

    a.   Need analysis.

    b.   Provide the user information pertaining to the preceding requirement.

4. *Operational Feasibility*

It determines how acceptable the software is within the organization. The evaluations must then determine the general attitude and skills. Such restriction of the job will be acceptable. To the users are enough

to run the proposed budget, hence the system is supposed to the feasible regarding all except of feasibility. In operational Feasibility, we attempt to ensure that every user can access the system easily.

Operational feasibility of the project also exists because in today's world most of the people are using the internet and are purchasing the products online. There is nothing complex in the system that cannot be used by people. It is socially accessible feasible as well because of its usefulness and easiness in getting information.

Time feasibility also exists because it can be developed and implemented in the given time. As far as legal feasibility is concerned there is no such restriction faced by the system.

2.4. PROJECT PLANNING

In planning phase, plan is made and strategies are set, taking into consideration the company policies, procedures and rules.

Planning provides direction, unifying frame work, performance standards, and helps to reveal future opportunities and threats

In Planning, the following steps are followed.

- The objectives of the projects in definite words

- Goals and stages intermediate to attain the final target

- Forecast and means of achieving goals i.e., activities.

- Organization resources-financial, managerial and operational-to carry out activities and to determine what is feasible and what is not.

- Alternatives-individual courses of action that will allow accomplishing goals.

- For consistency with company's policies.

- An alternative which is not only consistent with its goals and concept but also one that can be accomplished with the evaluated resources.

- Decision on a Plan.

2.5. PROJECT SCHEDULING

- Scheduling is the allocation of resources.
- Resources in conceptual sense are time & energy but in practical sense are the time, manpower, equipment applied to material.
- Scheduling is the process of formalizing the planned functions, assigning the starting and completion dates to each activity which proceeds in a logical sequence and in an orderly and systematic manner.

In Scheduling, the following steps are followed.

- Detailed control information is to be calculated.
- Timings to events & activities are assigned

- Consideration must be given to resources generally concerned with those resources whose availability is limited and which there by impose a constraint on the project. Important ones are skilled, technical and supervisory manpower and capital investment.
- Resource Allocation.

GANTT CHART

Gantt Chart is also known as Timeline Charts. A Gantt Chart can be developed for the entire project or a separate chart can be developed for each function. A tabular form is maintained where rows indicate the tasks with milestones and columns indicate duration (weeks/months). The horizontal bars that span across columns indicate duration of the task.

A Gantt Chart is constructed with a horizontal axis representing the total time span of the project, broken down into increments (for example, days, weeks, or months) and a vertical axis representing the tasks that make up the project.

Task	27Jan-24Feb	25Feb-9Mar	10Mar-12April	13April-16May	17May-22May	23May-28May
Develop Project Proposal	■■■◆ 29 Days					
Analysis		■■■◆ 13 Days				
Designing			■■■◆ 30 Days			
Coding				■■■◆ 34 Days		
Unit Testing					■■■◆ 5 Days	
Implementation						■■■◆ 5 Days

Table 1. Gantt Chart

2.6. SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

HARDWARE REQUIREMENT

1. *Server*

   - Processor   :   Dual Core or above
   - RAM   :   4 GB
   - HDD   :   256 GB
   - Display   :   15 Inch
   - Internet Connection

2. *Client*

   - Processor   :   Dual Core or above
   - RAM   :   4 GB
   - HDD   :   120 GB

- Display : 15 Inch
- Internet Connection

3. *Developer*

- Processor : i5 and above
- RAM : 8GB
- HDD : 256GB
- Display : 15 Inch

SOFTWARE REQUIREMENT

1. *Server*

- Browser : IE 8.0 or later
- Database : SQL Alchemy
- Web server : Internet Information Server (IIS) 8.0
- Operating System : Windows

2. *Client*

- Browser : IE 8.0 or any browser
- Operating System : Any O.S. Windows/Linux/Solaris

3. *Developer*

- Browser : IE 8.0 or any browser
- IDE : Visual Studio Code
- Database : SQL Alchemy
- Operating System : Window 7 or above
- Web server : Internet Information Server (IIS) 7.0
- Documentation tool : MS Word, MS Power point
- Scripting language : HTML, CSS, Bootstrap
- Server-side language : Python
- Framework : Plotly Dash

2.7. *SOFTWARE ENGINEERING PARADIGM APPLIED*

Software engineering is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of <u>software</u> systems.

The field of software engineering applies the disciplined, structured approach to programming that is used in engineering to software development with the stated goal of improving the quality, time and budget efficiency, along with the assurance of structured testing and engineer <u>certification</u>.

Software engineering is typically used for large and intricate software systems rather than single <u>applications</u> or <u>program</u>. Development, however, is simply one phase of the process. While a software engineer is typically responsible for the design of systems, programmers are often responsible for <u>coding</u> its implementation.

Software engineering involves a number of fields that cover the process of engineering software and certification including: requirements gathering, software design, software construction, software maintenance, software configuration management, software engineering management, software development process management and creation, software engineering models and methods, software quality, software engineering professional practices as well as foundational computing and mathematical and engineering study.

The software engineering paradigm which is also referred to as a software process modelor Software Development Life Cycle (SDLC) model is the development strategy that encompasses the process, methods and tools. There are common software process tasks, phases and activities that are modelled by software models.

THE WATERFALL MODEL

The life-cycle of paradigm demands a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing and maintenance.

The life-cycle paradigm encompasses the following activities.

- *Analysis*

  System Engineering and Analysis establishes requirements for all system elements and then allocating some subset of these requirements to the software. Software Requirements Analysis is used to understand the nature of the program to be built, the software engineer (analyst) must understand the document and review with the customer.

- *Design*

  Software Design is actually a multi-step process that focuses on four distinct attributes of the program: Data Structure, Software Architecture, Procedural Detail and Interface Characterization.

  The design process translates requirements into a representation of the software.

- *Coding*

  The design must be translated into a machine-readable form.

  The coding step performs this task.

- *Testing*

  Once code has been generated, program testing begins.

- *Maintenance*

  Software will undoubtedly and ergo change after it is delivered to the customer. Software Maintenance reapplies each of the preceding life-cycle steps to an existing program rather than a new one. Typical phases in the waterfall model are analysis and specification, design, coding, testing, integration and maintenance.

  The Waterfall model has some disadvantages, like,

  1. It works only for systems designed to automate an existing manual system. For absolutely new system determining the requirement is difficult as the user himself does not know them advance what is being built or finalized at each stage.

  2. Freezing the requirements means freezing the hardware. A large project might take few years to complete, by the time the product is available the hardware becomes obsolete.

3. It assumes that requirements are frozen before the rest of the development can proceed. In some situation, it might be desirable to develop a part of the system completely, and later enhance the system in phases.

This is often done for the software products that are developed not necessarily for a client, but for general marketing in which case the requirements are likely to be determined largely be developer themselves.
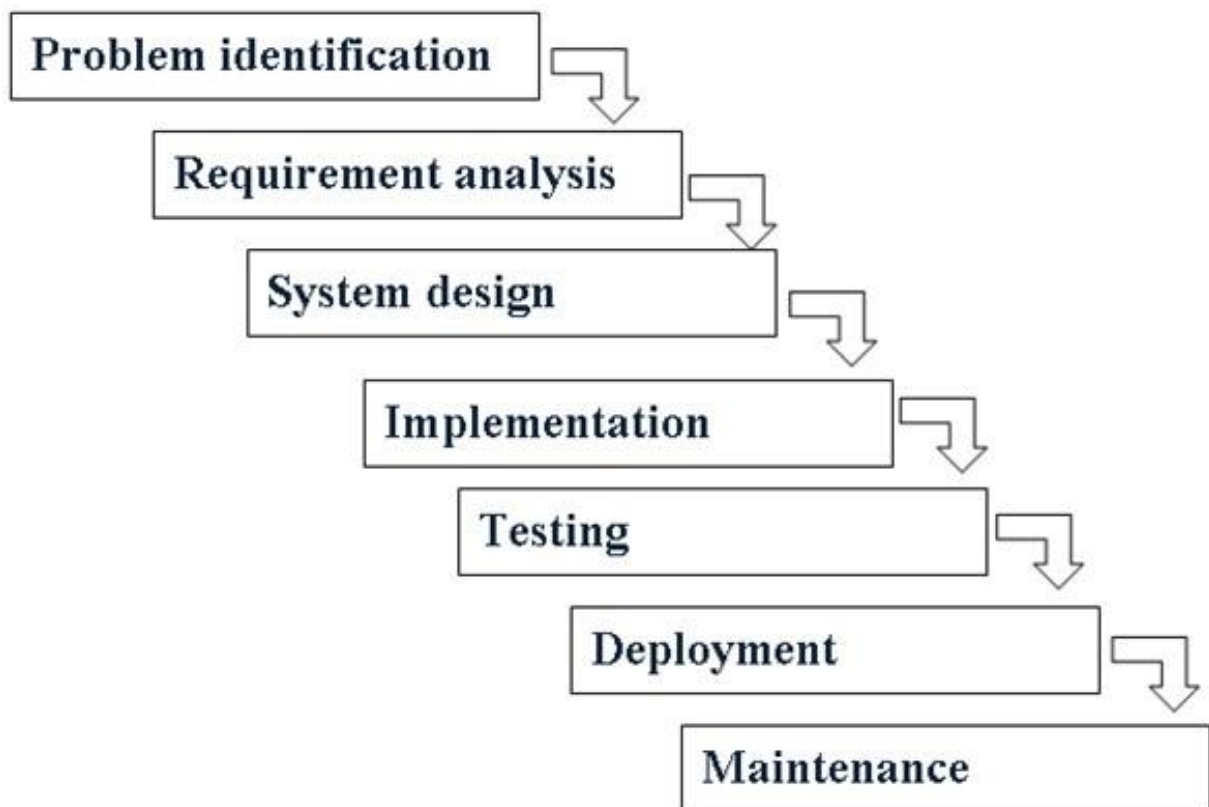

Fig 1. Waterfall Model

UML MODEL

High Level Design

<u>*CLASS DIAGRAM*</u>

The **class diagram** in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of thestructure of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.
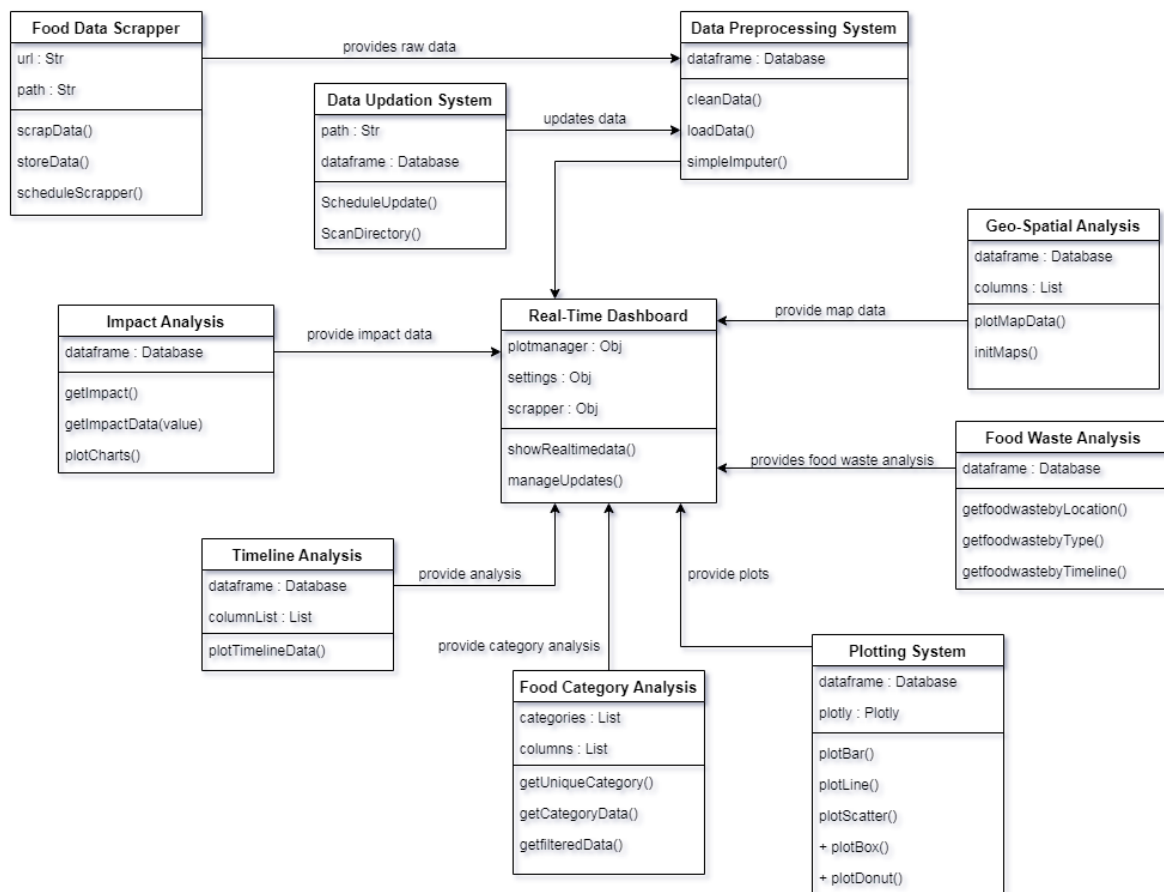

Fig 2. Class Diagram

A use case diagram at its simplest is representation of a user's interaction with the system that shows the relationship between the user and the different use case in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.
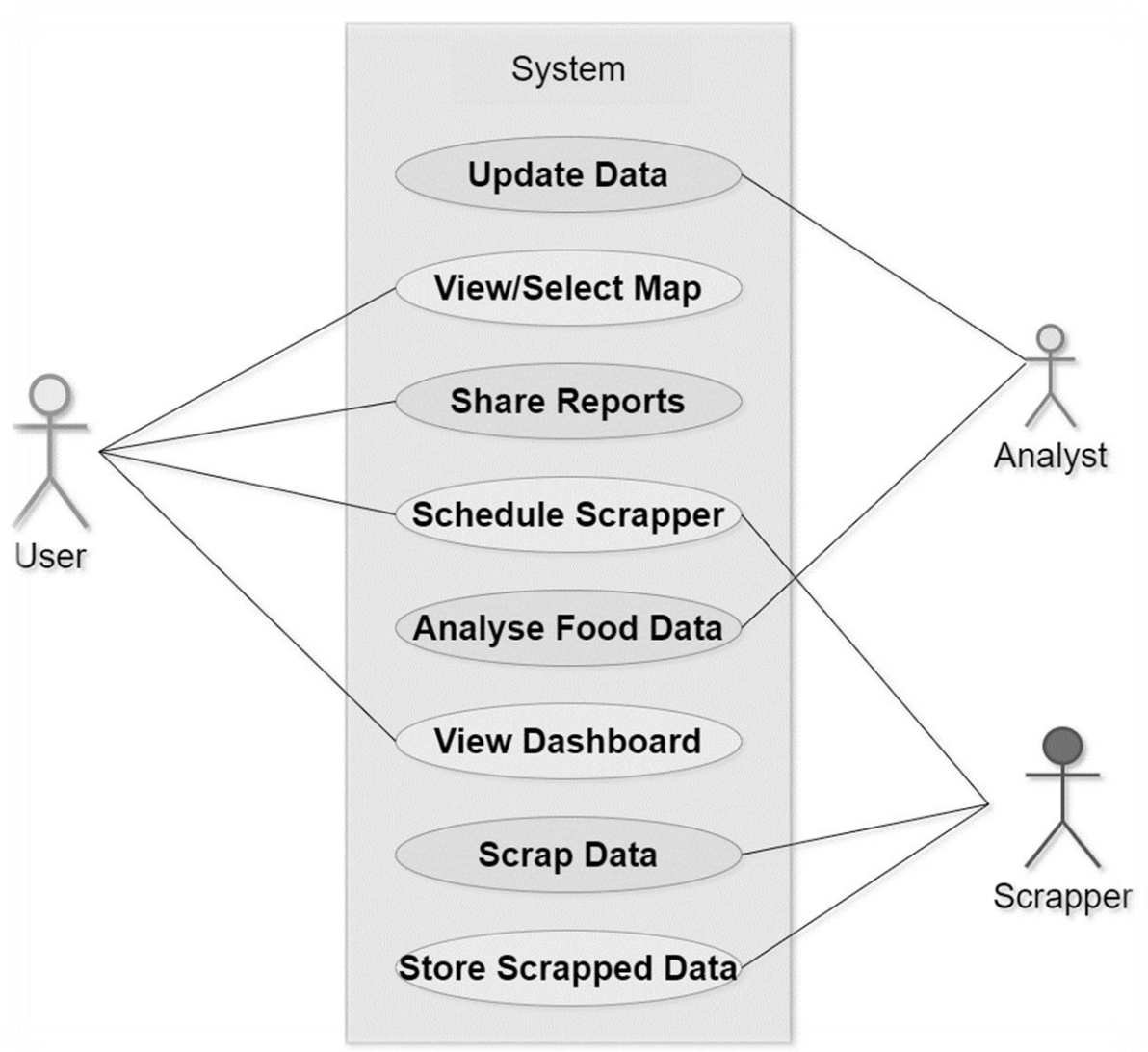

Fig 3. Use Case Diagram

3. <u>SYSTEM DESIGN</u>

3.1. *MODULARIZATION DETAILS*

1. DATA PREPROCESSING SYSTEM
    - Cleans the obtained dataset for visualization using python library pandas.

2. FOOD CATEGORY ANALYSIS
    - Identify unique food categories from the obtained datasets.
    - Plot graphs according to the categories produced, supplied or waste.

3. FOOD WASTE ANALYSIS
    - Identify unique food categories which are being wasted from the obtained dataset.
    - Identify unique areas which contribute to food waste.
    - Identify the unique cause of food wastage.

4. GEOSPATIAL DATA ANALYSIS
    - Plot geospatial maps on the basis of the unique areas provided by the previous module.

5. TIMELINE ANALYSIS
    - Get unique years of food production, supply and waste.
    - Plot graphs on the basis of the information provided above.

3.2. *DATA INTEGRITY AND CONSTRAINTS*

Data Integrity is the overall accuracy, completeness, and consistency of data. Data Integrity also refers to the safety of data in regards to regulatory compliance – such as GDPR compliance – and security. It is maintained by a collection of processes, rules, and

standards implemented during the design phase. When the integrity of data is secure, the information stored in a database will remain complete, accurate, and reliable no matter how long it's stored or how often it's accessed. Data Integrity also ensures that your data is safe from any outside forces.

- *Entity Integrity*

  Entity Integrity relies on the creation of primary keys, or unique values that identify pieces of data, to ensure that data isn't listed more than once and that no field in a table is null. It's a feature of relational systems which store data in tables that can be linked and used in a variety of ways.

- *Referential Integrity*

  Referential Integrity refers to the series of processes that make sure data is stored and used uniformly. Rules embedded into the database's structure about hoe foreign keys are used ensure that only appropriate changes, additions, or deletions of data occur. Rules may include constraints that eliminate the entry of duplicate data, guarantee that data is accurate, and/or disallow the entry of data that doesn't apply.

  The **constraints** are a rule which works into the table, it allows the data which are not following the rule of data constraints.

  For example, Suppose, you have table there may have more than one column, among them there is one phone_no column and you want, it should be unique then you will have to attach a Unique Key constraint, disallows the duplicity, Oracle engine will check the data for uniqueness, which is going to be enter if it's a unique among them which has already stored into the table, then it will be accepted otherwise it will be rejected.

Types of Data Constraints

1. *Input/Output Constraints*
   - The Primary Key Constraint
   - The Foreign Key Constraint

2. *Business Rule Constraints*
   - Check Constraint
   - Unique Constraint
   - NOT NULL Constraint

3.3. DATABASE DESIGN, PROCEDURAL DESIGN/OBJECT ORIENTED DESIGN

A relational database organizes data in tables. A table is made up of rows and columns. A row is also called a record. A column is also called a field (or attribute). A database table is similar to a spreadsheet. However, the relationships that can be created among the tables enable a relational database to efficiently store huge amount of data, and effectively retrieve selected data.

A language called SQL (Structured Query Language) was developed to work with relational databases.

Database Design Objective

Eliminate Data Redundancy

The same piece of data shall not be stored in more than one place. This is because duplicate data not only waste storage spaces but also easily lead to inconsistencies.

Relational Database Design Process

Step 1: Define the Purpose of the Database (Requirement Analysis)

Gather the requirements and define the objective of your database, e.g., drafting out the sample input forms, queries and reports, often helps.

Step 2: Gather Data, organize in tables and Specify the Primary Keys

In the relational model, a table cannot contain duplicate rows, because that would create ambiguities in retrieval. To ensure uniqueness, each table should have a column (or a set of columns), called primary key, that uniquely identifies every records of the table. For example, a unique number customer ID can be used as the primary key for the Customers table; product

Code for Products table; is bn for Books table. A primary key is called a simple key if it is a single column; it is called a composite key if it is made up of several columns.

Step 3: Create Relationships among Tables

A database consisting of independent and unrelated tables serves little purpose (you may consider to use a spreadsheet instead). The power of relational database lies in the relationship that can be defined between tables. The most crucial aspect in designing a relational database is to identify the relationships among tables.
The types of relationship include:

1. one-to-many
2. many-to-many
3. one-to-one

Step 4: Refine & Normalize the Design Normalization

Apply the so-called normalization rules to check whether your database is structurally

correct and optimal.

3.4.   *USER INTERFACE DESIGN*

User interface is the front-end application view to which user interacts in order to use the software. The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to Understand
- Consistent on all interface Screen

There are two types of User interface:

1. *Command Line Interface*
   Command Line Interface provides a command prompt, where   the user types the command and feeds to the system. The user needs to remember the syntax of the command and its use.

2. Graphical User Interface

Graphical User Interface provides the simple interactive interface to interact with the system. GUI can be a combination of both hardware and software. Using GUI, user interprets the software.
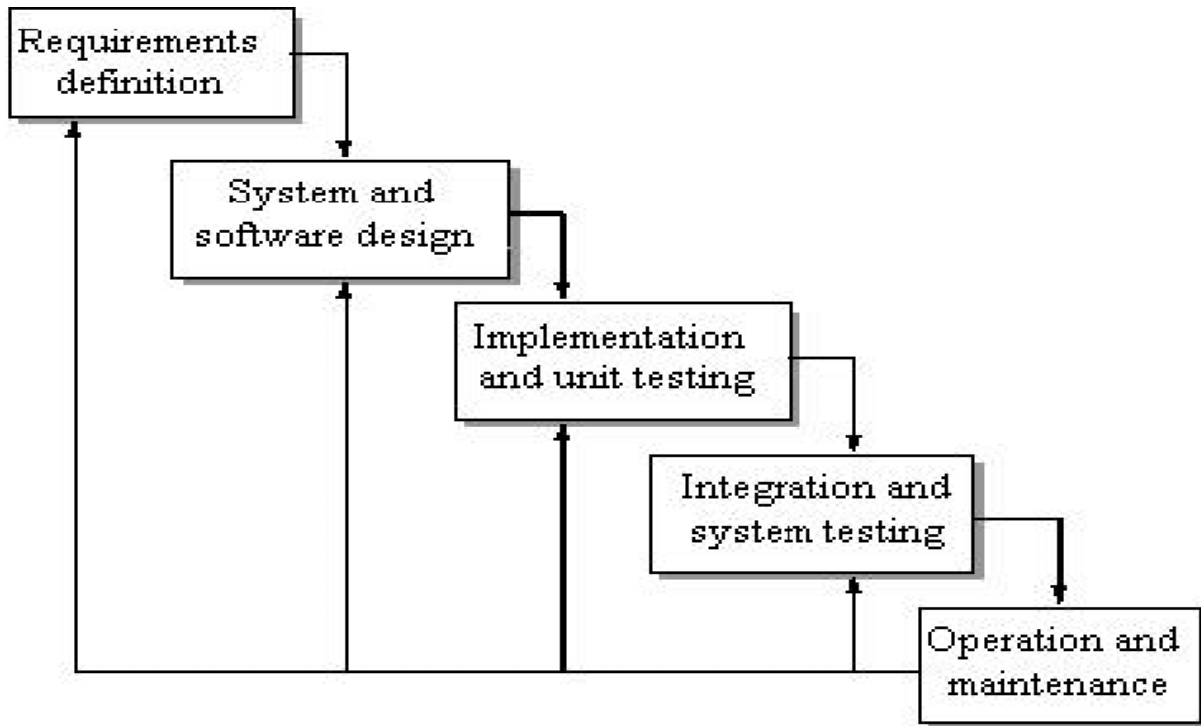

Fig 4. Waterfall Model

The analysis and design process of a user interface is iterative and can be represented by a spiral model. The analysis and design process of user interface consists of four framework activities.

1. *User, task, environmental analysis, and modeling*

Initially, the focus is based on the profile of users who will interact with the system, i.e. understanding, skill and knowledge, type of user, etc., based on the user's profile users are made into categories. From each category requirements are gathered. Based on the requirements developer understand how to develop the interface. Once all the requirements are gathered a detailed analysis is conducted. In the analysis part, the tasks that the user performs to establish the goals of the system are identified,

described and elaborated. The analysis of the user environment focuses on the physical work environment. Among the questions to be asked are:

- Where will the interface be located physically?
- Will the user be sitting, standing, or performing other tasks unrelated to the interface?
- Does the interface hardware accommodate space, light, or noise constraints?
- Are there special human factors considerations driven by environmental factors?

2. *Interface Design*

The goal of this phase is to define the set of interface objects and actions i.e., Control mechanisms that enable the user to perform desired tasks. Indicate how these control mechanisms affect the system. Specify the action sequence of tasks and subtasks, also called a user scenario. Indicate the state of the system when the user performs a particular task. Always follow the three golden rules stated by Theo Mandel. Design issues such as response time, command and action structure, error handling, and help facilities are considered as the design model is refined. This phase serves as the foundation for the implementation phase.

3. *Interface construction and implementation*

The implementation activity begins with the creation of prototype (model) that enables usage scenarios to be evaluated. As iterative design process continues a User Interface toolkit that allows the creation of windows, menus, device interaction, error messages, commands, and many other elements of an interactive environment can be used for completing the construction of an interface.

4. *Interface Validation*

This phase focuses on testing the interface. The interface should be in such a way that it should be able to perform tasks correctly and it should be able to handle a variety of tasks. It should achieve all the user's requirements. It should be easy to use and easy to learn. Users should accept the interface as a useful one in their work.

4. <u>TESTING</u>

Testing is a process of executing a program with the goal of finding errors. So, testing means that one inspects behavior of a program on a finite set of test cases (a set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement, for which valued inputs always exist. In practice, the whole set of test cases is considered as infinite, therefore theoretically there are too many test cases even for the simplest programs. In this case, testing could require months and months to execute. So, how to select the most proper set of test cases? In practice, various techniques are used for that, and some of them are correlated with risk analysis, while others with test engineering expertise. Testing is an activity performed for evaluating software quality and for improving it. Hence, the goal of testing is systematic detection of different classes of errors (error can be defined as a human action that produces an incorrect result, in a minimum amount of time and with a minimum amount of effort.

4.1. TESTING TECHNIQUES

A testing technique specifies the strategy used in testing to select input test cases and analyze test results. Different techniques reveal different quality aspects of a software system, and there are two major categories of testing techniques, **functional** and **structural**.

- *Functional Testing*

  The software program or system under test is viewed as a **"black box"**. The selection of test cases for functional testing is based on the requirement or design specification of the software entity under test. Examples of expected results, sometimes are called test oracles, includes requirement /design specifications, hand calculated values, and simulated results. Functional testing emphasizes on the external behavior of the software entity.

- *Structural Testing*

  The software entity is viewed as a **"white box"**. The selection of test cases is based on the implementation of the software entity. The goal of selecting such test cases is to cause the execution of specific spots in the software entity, such as specific statements, program branches or paths. The expected results are evaluated on a

set of coverage criteria. Examples of coverage criteria include path coverage, branch coverage, and data-flow coverage. Structural testing emphasizes on the internal structure of the software entity.

TESTING STRATEGIES

- *Unit Testing*

  Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

  Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

  Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

- *Integration Testing*

  Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is

considered a better practice since it allows interface issues to be localized more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

- *System Testing*

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification.

- *Acceptance Testing*

Acceptance testing is a test conducted to determine if the requirements of a specification or contract are met. It may involve chemical tests, physical tests, or performance tests.

In systems engineering it may involve black-box testing performed on a system (for example: a piece of software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery.

Software developers often distinguish acceptance testing by the system provider from acceptance testing by the customer (the user or client) prior to accepting

transfer of ownership. In the case of software, acceptance testing performed by the customer is known as **user acceptance testing (UAT)**, end-user testing, site (acceptance) testing, or field (acceptance) testing. A smoke test is used as an acceptance test prior to introducing a build to the main testing process.

4.2. TEST PLAN

1. Static Testing

Static testing is the testing of the objects in a web browser that do not change, or are not transaction based. This type of testing is done on a web page that has already been loaded into a web browser. There are several types of static testing, and they will be discussed in this section.

- *Content Checking*

  Once the web page has been loaded, it has to be tested for accuracy, completeness, consistency, spelling and accessibility.

  These terms have the traditional meanings, and the tests are as elementary as they sound. However, it is in areas like these where the site is first judged by the website visitor. For example, if the site has numerous misspellings, the product that the website is offering may come into question as the visitor may feel that if the attention to detail is not given to the site, it may not be given to the product either. These tests are mentioned in this research paper as these are simple things that may not automatically be on a web tester's test plan, as most of these are unique to the web.

- *Browser Syntax Compatibility*

  This test is one level below the actual content. It is the technology of how to represent the content, whether that content consists of text, graphics, or other web objects. This is an important test as it determines whether or not the page under test works in various browsers. Even regarding only Microsoft's Internet Explorer and Netscape's Navigator web browsers, this is a significant issue due to the fact that there are many versions of both still in use. These versions do not work the

same, and depending on what the minimum version and browser type requirements are, the pages need to be tested in each supported browser.

- *Visual Browser Validation*

  Simply, does the content look the same, regardless of supported browser used? If the requirements are that only one browser and version will be supported by the application, this test is not necessary. However, even if more than one version of the same browser will be supported, the page under test should loaded into both browsers, and they should be visually checked to see if there are any differences in the physical appearance of the objects in the page. If there are, they may be things such as the centering of objects, table layouts, etc. The differences should be reviewed by the users to see if there is any need to change the page so that it appears exactly the same (if possible) in all of the supported browsers.

- *Test Browsing*

  Test browsing tests aim to find the defects regarding navigation through web pages, including the availability of linked pages, and other objects, as well as the download speed of the individual page under test. The integration of web pages to server- based components is tested, to ensure that the correct components are called from the correct pages.

- *Browsing the Site*

  When traversing links and opening new pages, when a new page is opened, several questions should be addressed on each and every page the system links to. Can the page be downloaded and displayed? Do all objects load in an acceptable time ("acceptable" would be based on the business requirements)? When user turns the browser option of "images-load" to "off" – does the page still work? Do all of the text and graphical links work? All of these questions are important, as if the answer to any of them is in the negative, it would be considered a defect.

  Other issues to validate are whether the site still works if JavaScript or Java is disabled, or if a certain plug-in is not loaded or disabled. A good test case is to use a browser with no plug-ins loaded during testing, and when the tester is queried to download a plug-in, they should not load them, and see how the site reacts without the plug-in.

2. Functional Testing

- *Browser-Page Tests*

  This type of test covers the objects and code that executes within the browser, but does not execute the server-based components. For example, JavaScript and VBScript code within HTML that does rollovers, and other special effects. This type of test also includes field validations that are done at the HTML level. Additionally, browser-page tests include Java applets that implement screen functionality or graphical output.

3. Non-Functional Testing

- *Configuration Testing*

  Beyond the browser validation, this type of test takes into consideration the operating system platforms used, the type of network connection, internet service provider type, and browser used (including version). The real work for this type of test is ensuring that the requirements and assumptions are understood by the development team, and that a test environment with those choices is put in place to properly test it.

- *Usability*

  For usability, the tests can be subjective, but there are standards and guidelines that have been established throughout the industry and it would be easy for a project team to blindly follow them, and feel that the site will be acceptable since the standards are followed. However, human-computer interaction standards and guidelines cannot guarantee a usable website. Designers should not rely on them for all or even most of the design decisions, and project managers should not let standards compliance lull the team into complacency, thinking that since the standards are followed, that the site will automatically meet the needs of the users, their tasks, and their work environment.

  A proactive suggestion is that while establishing the design guidelines, to define requirements that can be positively identified and measured. A way to do this is to capture and quantify the meaning of learn ability, understand ability, and operability in a testable form.

- *Performance*

Performance testing is the validation that the system meets performance requirements. This can be as simplistic as ensuring that a web page loads in less than eight seconds, or can be as complex as requiring the system to handle 10,000 transactions per minute, while still being able to load a web page within eight seconds. The section below offers best practices for the execution of performance testing.

During research, one topic that was repeated was the importance that the performance- testing server is exactly like the production server; in fact, it ideally should be an exact replica in every way. It is very important to make sure every component (networks, firewalls, servers, mainframes) matches the production equipment.

Performance testing can be done through the "window" of the browser, or directly on the server. If done on the server, some of the performance time that the browser takes is not accounted for. Scripting GUI orientated transactions to drive the browser interface can be much more complicated and the synchronization between test tool and browser it not always reliable. Therefore, if testers decide to ignore the performance time taken by the browsers, it is important to get "buy in" from project team members and users to understand the compromise. If there are issues, testers should advise management that performance-testing using the GUI will introduce a time-intensive effort that may or may not impact the project timeline.

To assist with load and performance testing, testers should use the test scripts that have been created early in the project as a basis for initial load testing. By using the existing scripts, this avoids rework and allows the scripts to be used at different times by different virtual users when validating system performance

- *Scalability*

The term "scalability" can be defined as a web application's ability to sustain its required number of simultaneous users and/or transactions, while maintaining adequate response times to its end users.

When testing scalability, configuration of the server under test is critical. All logging levels, server timeouts, etc. need to be configured just like production. In an ideal situation, all of the configuration files should be simply copied from test

environment to the production environment, with only minor changes to the global variables.

In order to test scalability, the web traffic loads must be determined to know what the threshold requirement for scalability should be.

- *Security*

  Security is a critical part of an e-commerce website. Best practices to best test how secure the site is being in this section.

- *Data Collection*

  Web sites collect data in log files, as well as through forms in which users supply to the website information that is saved on the web server. The web server should be setup so that users cannot browse directories and obtain file names.

- *Cookies*

  A cookie is a text file that is placed on a website visitor's system that identifies the user's "identity." The cookie is retrieved when the user re-visits the site at a later time. Cookies can expire in a short period of time, such as minutes or hours (session cookie) or can last for months or years (persistent cookie). Cookies can be controlled by the user, regarding whether they want to allow them or not.

4.3. DEBUGGING

After I have created my TOOL and resolved the build errors. I must now correct those logic errors that keeps my application or stored procedures from running correctly. I have done this with the development environment's integrated debugging functions. These allowed me to stop at procedure locations, inspect memory and register values, change variables, observe message traffic, and get a close look at what my code does.
Visual Studio Code supports various types of debugging which helped me to debug my website.

- *Parallel Debugging*

  Two new windows have been added for debugging parallel applications:

The GPU Threads window displays the status and the details of the threads running on the GPU.

The Parallel Watch window displays values of a single expression across multiple threads at the same time.

You can sort, reorder, configure, and group on the columns in the GPU Threads, Threads, Parallel Tasks, and Parallel Watch windows.

- *IntelliTrace Debugging*

  You can record diagnostic events with the IntelliTrace collector for SharePoint 2010 applications running outside Visual Studio. This lets you save user profile events, Unified Logging System (ULS) events, and IntelliTrace events to an *iTrace* file. You can open this file in Visual Studio Ultimate to start diagnosing SharePoint 2010 applications in production or other environments.

5. SYSTEM SECURITY MEASURES

5.1. DATABASE SECURITY

Database security is addressed at several levels:

- Data base file protection. All files stored within the database are protected from reading by any account other than the Postgres superuser account.

- Connections from a client to the database server are, by default, allowed only via a local Unix socket, not via TCP/IP sockets. The backend must be started with the option to allow non-local clients o connect.

- Client connections can be restricted by IP address and/or user name via the pg_hba.conf file in PG_DATA.

- Client connections may be authenticated vi other external packages.

- Each user in Postgres is assigned a username and (optionally) a password. By default, users do not have write access to databases they did not create.

- Users may be assigned to groups, and table access may be restricted based on group privileges.

6. COST ESTIMATION WITH COST ESTIMATION MODEL

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e. **number of Lines of Code**. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality. It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models.

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily:

- **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.

- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, months.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm's definition of organic, semidetached, and embedded systems:

1. **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

2. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Example, Compilers or different Embedded Systems can be considered of Semi-Detached type.

3. **Embedded –** A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

All the above system types utilize different values of the constants used in Effort Calculations.

The Intermediate COCOMO

The intermediate model estimates software development effort in terms of size of the program and other related cost drivers parameters (product parameter, hardware parameter, resource parameter, and project parameter) of the project. The estimated effort and scheduled time are given by the relationship:

Effort (E) = a*(KLOC)b *EAF MM
Scheduled Time (D) = c*(E)d Months(M)

Where,

E = Total effort required for the project in Man-Months (MM).
D = Total time required for project development in Months (M).
KLOC = The size of the code for the project in Kilo lines of code.
a, b, c, d = The constant parameters for the software project.
EAF = It is an Effort Adjustment Factor, which is calculated by multiplying the parameter values of different cost driver parameters. For ideal, the value is 1.

The constant values a,b,c, and d for the Basic Model for the different categories of the system

Software Project	A	B	C	D
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Fig 10. Cocomo Model

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Product Parameter					
Required Software	0.75	0.88	1	1.15	1.4
Size of Project Database	NA	0.94		1.08	1.16
Complexity of The Project	0.7	0.85		1.15	1.3
Hardware Parameter					
Performance Restriction	NA	NA	1	1.11	1.3
Memory Restriction	NA	NA		1.06	1.21
virtual Machine Environment	NA	0.87		1.15	1.3
Required Turnabout Time	NA	0.94		1.07	1.15
Personnel Parameter					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17		0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming Experience	1.14	1.07		0.95	NA
Project Parameter					
Software Engineering Methods	1.24	1.1	1	0.91	0.82
Use of Software Tools	1.24	1.1		0.91	0.83
Development Time	1.23	1.08		1.04	1.1

Fig 11. Cost Estimation

Estimated cost of project:

- Project having 1100 line of code

- Estimated size of the project =1.1 KLOC

- Developer having highly application experience=1.13

- Developer having low experience in programming=1.07

- Software engineer capacity =1.17

- Required Software is normal=1

- Size of the database is Normal=1

- Size of project is normal=1

- Virtual machine environment is normal=1

EAF = 1.13*1.07*1.17*1*1*1*1

EAF=1.414647

Effort (E) = a*(KLOC)b *EAF MM

a =3.0

b =1.12

E=3.0*(1.1) 1.12*1.414647

E=4.72 MM

Scheduled Time (D) = c*(E)d Months(M)

c =2.5

d =0.35

Schedule Time(D)=2.5(4.72)0.35

=4.30

The value for project duration enables the planner to determine a recommended number of people, N, for the project:

N = E/D

N=1.097

7. SCREENSHOTS

HOME

Fig 12. Home Page
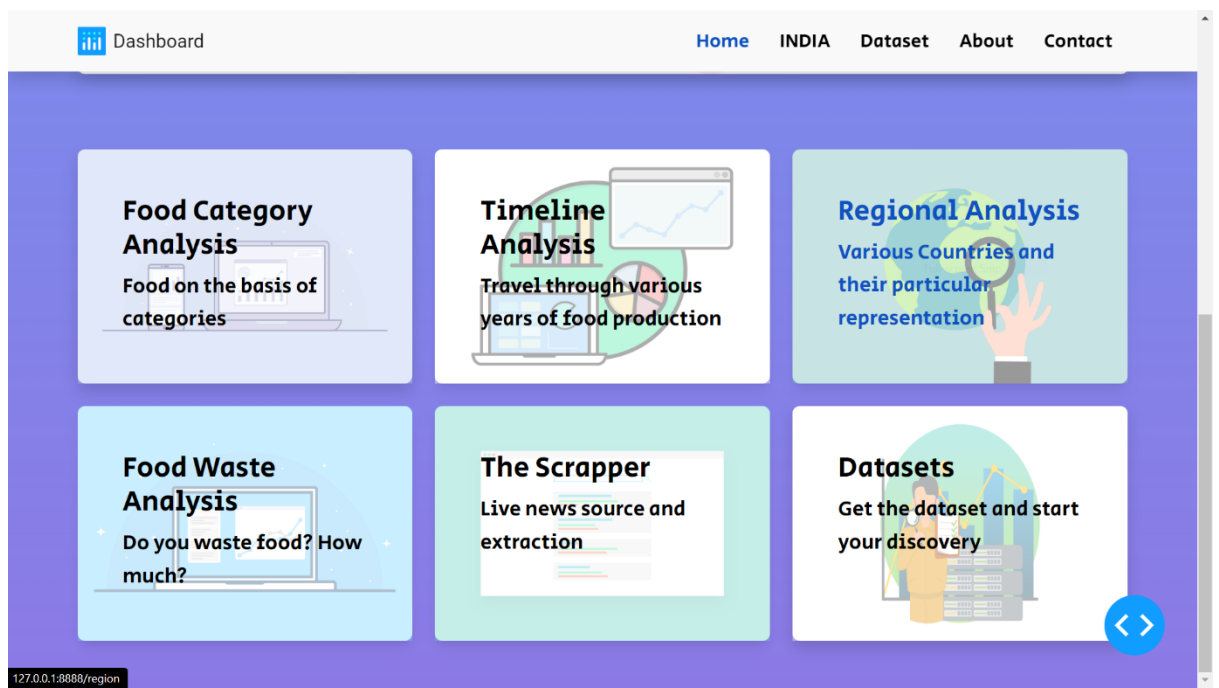
Fig 13. Home Page 2

<u>INDIA</u>

Fig 14. India
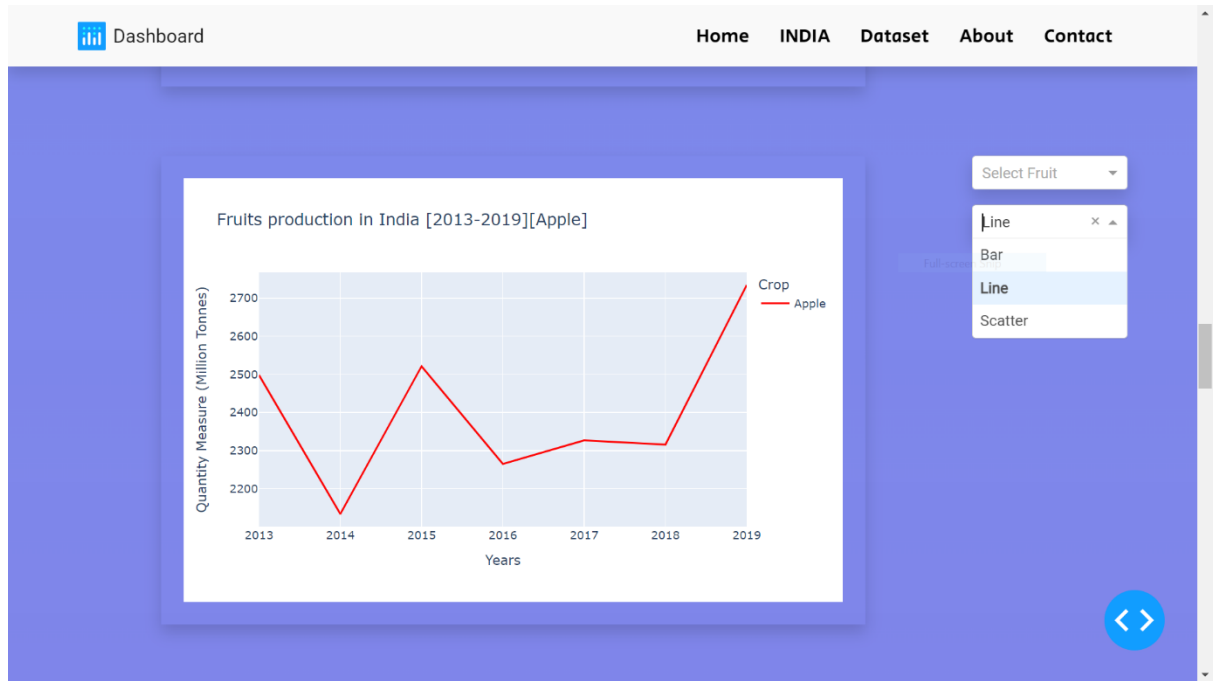
Fig 15. India

DEMO DATASET

Fig 16. Dataset

Fig 17. Dataset 2

ABOUT

Fig 18. About

CONTACT

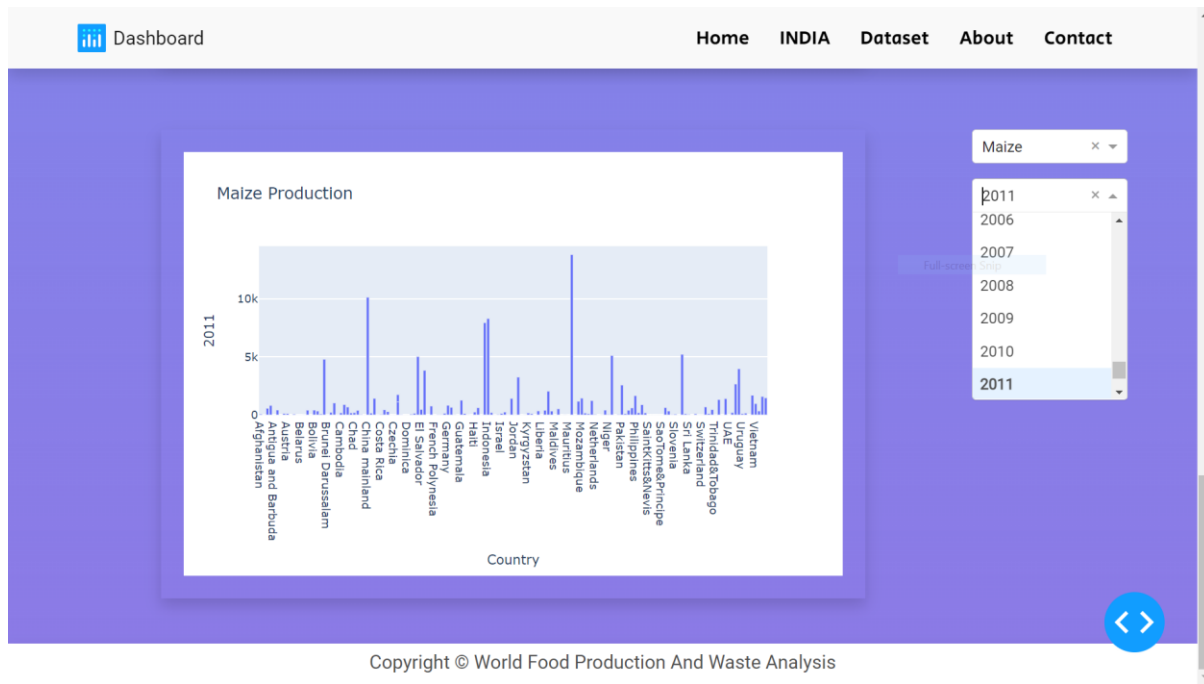

Fig 19. Contact

FOOD CATEGORY ANALYSIS

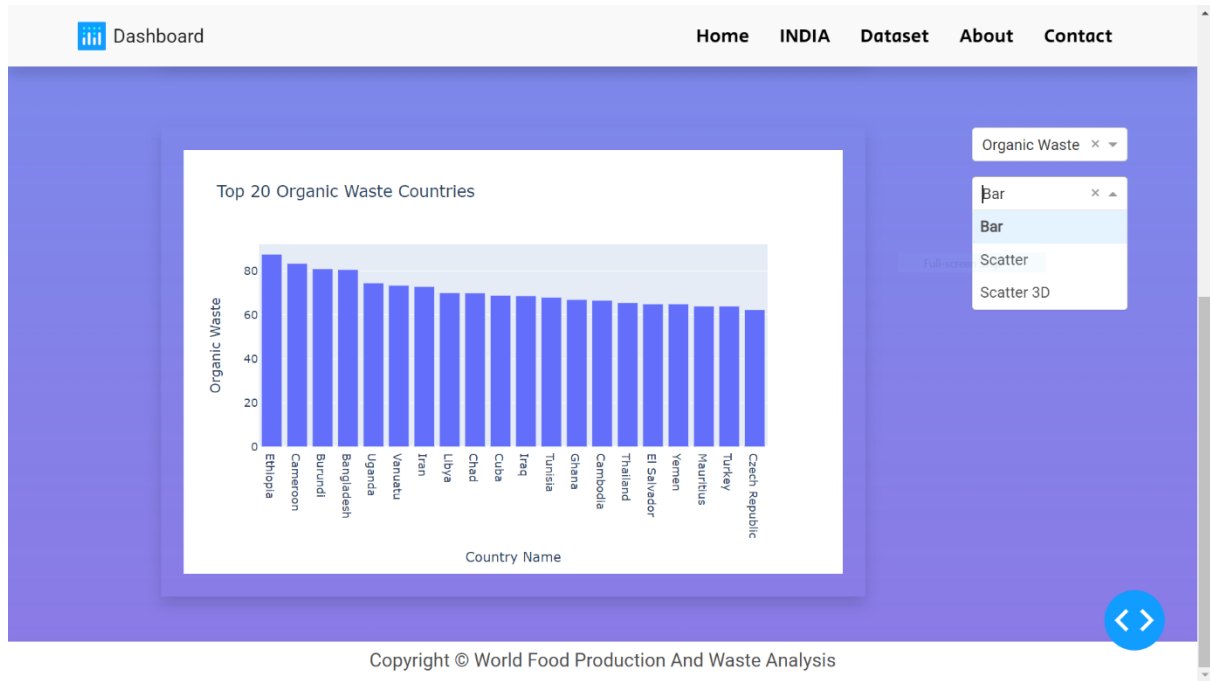Fig 20. Food Category Analysis

FOOD WASTE ANALYSIS
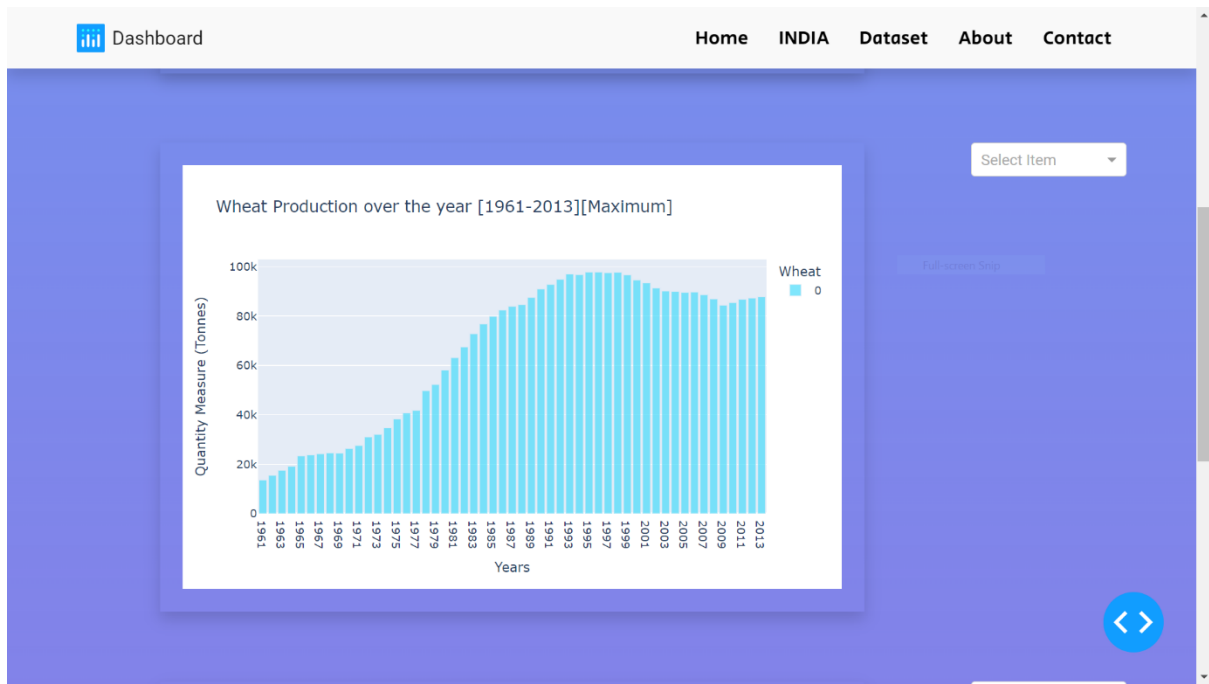
Fig 21. Food Waste Analysis

TIMELINE ANALYSIS

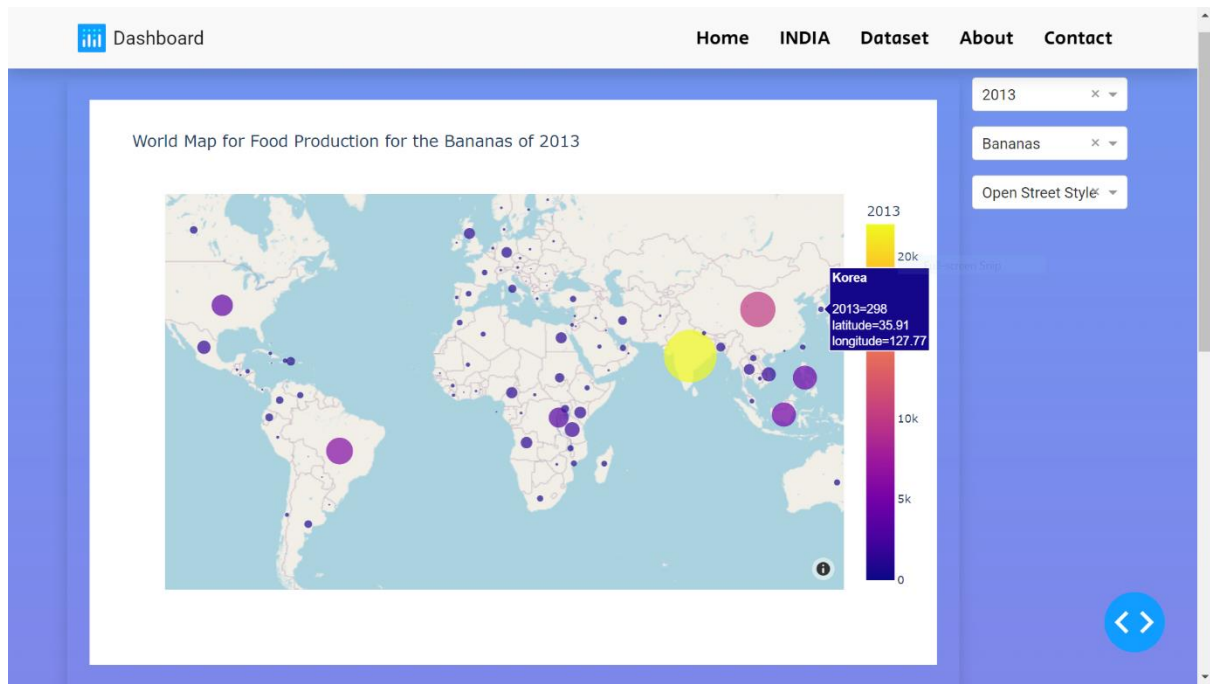Fig 22. Timeline Analysis

REGIONAL ANASLYSIS
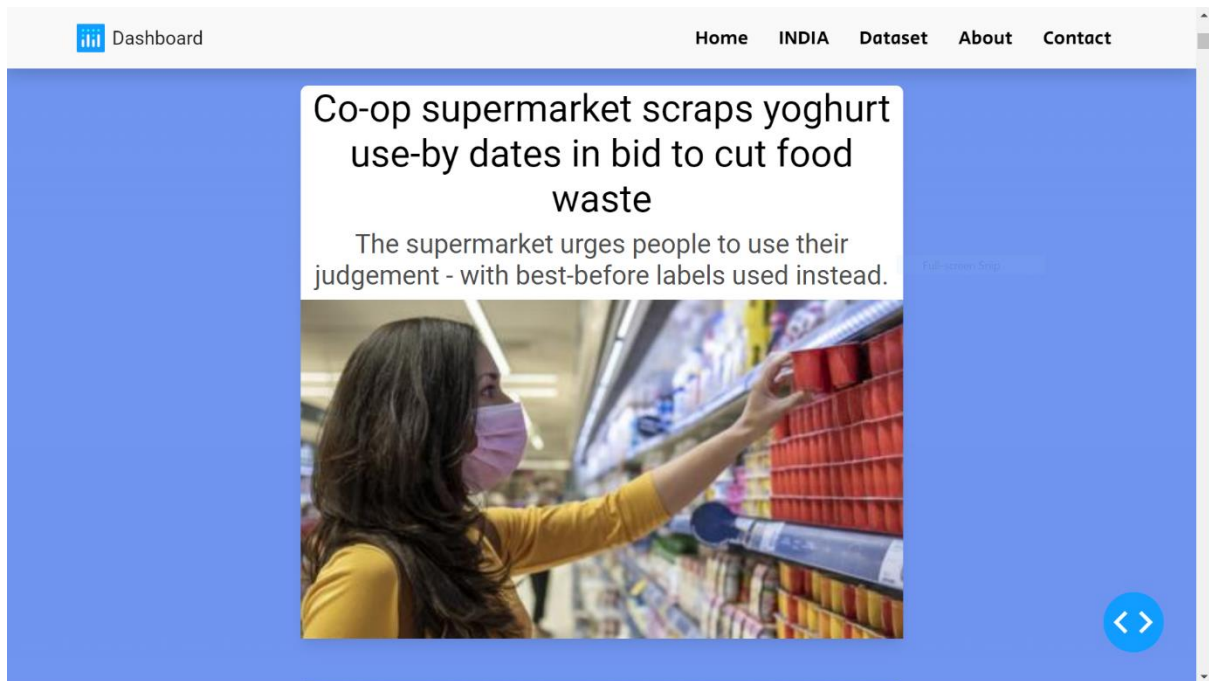

Fig 23. Regional Analysis

<u>SCRAPPER</u>

Fig 24. Scrapper

DATASETS FOR DOWNLOADING

Fig 25. Datasets

8. **CONCLUSION**

This project aims to stretch a helping hand to humans in need. As said enough, 'Humanity isn't dead yet!'.

The project states that the upcoming years will be the fruitful reward of all the hard work and the intelligence of analytic techniques. It will be a bridge to fill the gap between the need and the fulfilment. We will also be sending a strong message of social responsibility among the youth, eventually responsible for the dynamic growth.

9. FUTURE SCOPE

The project in current status is the minimum viable attempt.

As per the trends in technology and concentrating on the future. This project holds a great scope. As it provides a great visualization dashboard which can be presented in front of people with full confidence and therefore is highly useful.

The new and further components techniques and technologies that can be added to this project are:

    i.   Machine Learning.

    ii.   Automated Data Wrangling using python libraries.

    iii.   Report Saving.

    iv.   Varied number of Datasets.

    v.   More Classifications.

10. BIBLIOGRAPHY

- https://en.wikipedia.org/
- https://stackoverflow.com/
- https://www.github.com
- https://www.python.org
- https://dash.plotly.com/
- https://www.fao.org/faostat/en/

11. GLOSSARY

- **Field:** An attribute on a model; a given field usually maps directly to a single database column.
- **Model:** Models store your application's data.
- **Template:** A chunk of text that acts as formatting for representing data. A template helps to abstract the presentation of data from the data itself.
- **View:** A function responsible for rendering a page.
- **Dash:** A Python package – i.e., a directory of code – that contains all the settings for an instance of Dash. This would include database configuration, Dash-specific options and application-specific settings.
- **Database:** Collection of relations or table.
- **Authentication:** Process of identifying authentic user.
- **Foreign Key:** Key that helps to connect two table.
- **Primary Key:** Unique key that identify particular row items in a table.