GENUINE INFORMATION FINDER

(G!F)

A Project Report

Submitted for the partial fulfilment for the award of the degree of

B.TECH CSE 3RD YEAR

Submitted by

HARSHITA ARORA : 2116166

HARSHITA BAJIYA : 2116167

ISHANI VERMA : 2116172

ISHIKA TRIPATHI : 2116175

Under the supervision of

DR. KUSUM GUPTA

Department of Computer Science
Banasthali Vidyapith

Session: 2023-24

Certificate

Certified that **Ms. HARSHITA ARORA, Ms. HARSHITA BAJIYA, Ms. ISHANI VERMA, Ms. ISHIKA TRIPATHI** has carried out the project work titled **"GENUINE INFORMATION FINDER (G!F)"** from 15.9.24 to 8.4.24 for the award of the B.TECH CSE 3RD YEAR from BANASTHALI VIDYAPITH under my supervision. The thesis embodies result of original work and studies carried out by Student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

DR. KUSUM GUPTA Kusum

Designation: PROFESSOR

Place: AIM & ACT

Date: 7.4.2024

Abstract

The proliferation of misinformation in today's digital age poses a significant challenge to society. In response, this project introduces "Genuine Information Finder," an AI-powered tool designed to discern the veracity of factual claims. Leveraging advanced algorithms and machine learning techniques, including natural language processing, logistic regression, and decision trees, our model analyses textual data to determine the authenticity of statements. Targeted towards youth, our tool aims to combat the spread of fake facts by empowering users to verify the accuracy of information they encounter. By providing a reliable mechanism for fact-checking, "Genuine Information Finder" contributes to promoting informed decision-making and mitigating the impact of misinformation in the digital landscape.

Acknowledgments

We extend our sincere gratitude to all those who have contributed to the successful completion of this project, "Genuine Information Finder."

First and foremost, we would like to express our deepest appreciation to our project mentor, Dr. Kusum Gupta, whose guidance, support, and invaluable insights were instrumental throughout every stage of the project. Their expertise and encouragement have been invaluable in shaping our approach and refining our methodologies.

We are also grateful to our faculty members Mr Deepak Kumar and Mrs Neelam Sharma for their encouragement and feedback, which greatly enriched our understanding of the subject matter and helped us navigate through challenges.

We would like to acknowledge the contribution of our peers and colleagues for their assistance, constructive criticism, and discussions, which contributed significantly to the development and improvement of our project.

Furthermore, we extend our thanks to the participants who generously volunteered their time and provided valuable feedback during the testing phase, enabling us to refine our tool and enhance its effectiveness.

Last but not least, we express our heartfelt appreciation to our families and friends for their unwavering support, understanding, and encouragement throughout this endeavour.

This project would not have been possible without the collective efforts and support of all those mentioned above. Thank you for your invaluable contributions.

Table of Contents

1. Introduction

The objective of Genuine Information Finder is to develop an artificial intelligence-based system capable of analysing facts and other textual content to identify and categorize potentially fake or misleading information.

1.1 Purpose

The purpose of this project is to create a reliable and efficient tool that can help combat the spread of misinformation and fake news across digital platforms. The system will assist users in making more informed judgments about the content they encounter online.

1.2 Scope

In Scope:

The following elements are included in the scope of the project:

- Training the proposed model with a diverse and extensive dataset.
- User interface for interacting with the proposed model.
- Regular model updates and improvements based on user feedback.

1.3 Definitions, Acronyms, and Abbreviations.

Definitions

- Software Requirements Specification (SRS): A comprehensive document that outlines the functional and non-functional requirements for a software system.
- System: The software product or application for which the requirements are being specified.
- Functional Requirement: Describes what the system should do, such as features, operations, and interactions.

• Non-Functional Requirement: Specifies constraints and quality attributes, like performance, security, and usability.

• Requirement Document: A collective term for all documents related to system requirements, including the SRS.

• Scope: Defines the boundaries of what the system is and is not intended to do.

• Constraint: A limitation or restriction that must be considered when designing and implementing the system.

• Dependency: A relationship between requirements where one requirement relies on another.

• Agile: An iterative and flexible approach to software development, emphasizing collaboration and customer feedback.

• Waterfall: A traditional, linear approach to software development, with phases like requirements, design, implementation, testing, and maintenance.

• Fake News Detection: The process of identifying news articles, stories, or information that are intentionally fabricated, misleading, or inaccurate.

• Fact Verification: The process of confirming the accuracy and authenticity of news articles and claims through thorough investigation and credible sources.

• Web Interface: The graphical user interface accessible through web browsers that allows users to interact with the AI tool.

• User Authentication: The process of confirming the identity of users who access the system to ensure only authorized individuals can use the tool.

• User Authorization: Permissions and controls that dictate what actions and features users can access within the application.

• Machine Learning (ML): The subset of artificial intelligence that uses algorithms to enable the AI tool to learn from data and improve its fake news detection abilities over time.

• Natural Language Processing (NLP): A field of AI that focuses on the interaction between computers and human language, enabling the tool to understand and process text data.

• Algorithm: The set of rules and processes used by the AI tool to assess the credibility of news articles.

Acronyms:

• SRS: Software Requirements Specification

• AI: Artificial Intelligence

• ML: Machine Learning

• NLP: Natural Language Processing

• UI: User Interface

• API: Application Programming Interface

• CPU: Central Processing Unit

• GPU: Graphics Processing Unit

• TLS: Transport Layer Security

• GDPR: General Data Protection Regulation

• JSON: JavaScript Object Notation

• CSV: Comma-Separated Values

• REST API: Representational State Transfer Application Programming Interface

Abbreviations:

• UAT: User Acceptance Testing - The process of testing a software application with endusers to ensure it meets their needs and requirements.

• QA: Quality Assurance - The process of ensuring that the software meets the specified quality standards and requirements.

• DB: Database - A structured collection of data used to store and retrieve information.

• URL: Uniform Resource Locator - A web address that specifies the location of a resource on the internet.

• HTML: Hypertext Markup Language - The standard markup language used to create web pages.

• CSS: Cascading Style Sheets - A stylesheet language used to describe the presentation of a document written in HTML.

• JS: JavaScript - A programming language commonly used to add interactivity and dynamic behaviour to web pages.

• API: Application Programming Interface - A set of rules and protocols that allows different software applications to communicate with each other.

• SQL: Structured Query Language - A domain-specific language used for managing and querying relational databases.

• HTTPS: Hypertext Transfer Protocol Secure - The secure version of the HTTP protocol used for secure communication over the internet.

• GUI: Graphical User Interface - A visual way for users to interact with software, including buttons, menus, and windows.

• OS: Operating System - Software that manages computer hardware and provides services for computer programs.

• RAM: Random Access Memory - A type of computer memory that is used to store data that is being actively used or processed.

• CPU: Central Processing Unit - The primary component of a computer that executes instructions and performs calculations.

• API: Application Programming Interface - A set of rules and protocols that allow different software applications to communicate with each other.

• URL: Uniform Resource Locator - A web address that specifies the location of a resource on the internet.

1.4 Overview

This section provides a brief guide to the contents and organization of the Software Requirements Specification (SRS).

1.4.1 What to Expect

1.4.1.1 Customer Perspective

For customers and potential users, the SRS primarily delves into the envisioned features and functionalities of the AI-based news detection and verification tool. The crucial aspects for your attention are:

Section 2 (System Overview): Understand the high-level description of the system, its goals, and the overall features that will be available to end-users.

Section 2.2 (Features): Explore the detailed functionalities, such as news detection, verification, and the user interface. This section provides a comprehensive understanding of what the tool is designed to accomplish.

Section 3 (Functional Requirements): Focus on the specific requirements from a user's perspective. This includes user registration, news submission, and the presentation of analysis results.

1.4.1.2 Developer Focus

For developers and the technical team involved in the implementation, the SRS is a roadmap that guides the development process. Key areas of interest include:

Section 3 (Functional Requirements): Delve into the specific functionalities and operations that the system must perform. Developers should pay close attention to the details outlined here.

Section 4 (Non-Functional Requirements): Understand the performance, security, and scalability aspects of the system. These requirements shape the technical architecture and implementation strategy.

Section 5 (User Interface Design): Developers involved in designing and implementing the user interface should refer to this section for insights into the expected user experience.

1.4.2 Organization of the Document

The SRS is organized to provide a clear structure for different stakeholders:

Section 1 (Introduction): Gives an overview of the project's purpose and scope.

Section 2 (System Overview): Provides a high-level description of the system, including its features.

Section 3 (Functional Requirements): Details the specific functionalities expected from the system.

Section 4 (Non-Functional Requirements): Outlines performance, security, and scalability aspects.

Section 5 (User Interface Design): Focuses on the design and user experience.

Section 6 (Conclusion): Summarizes the key points and serves as a closing statement.

This organization is designed to facilitate efficient navigation, allowing each stakeholder to access the information most relevant to their role and interests.

2. Requirement Analysis

2.1 Requirement Specification

2.1.1. Stakeholder Identification:

- **End Users:** The primary stakeholders are the users of the "Genuine Information Finder" tool, primarily youth who seek to verify the accuracy of information they encounter online.
- **Developers:** The team responsible for designing, developing, and maintaining the AI-powered tool.
- **Supervisor and Faculty Members:** The project supervisor and academic faculty members provide guidance, feedback, and evaluation throughout the project.

2.1.2. Functional Requirements:

- **Input Interface:** The tool should provide a user-friendly interface for users to input textual claims or statements for verification.

- **Processing Algorithm:** Implementation of AI algorithms, including natural language processing, logistic regression, and decision trees, to analyse the input data and determine the veracity of the claims.

- **Output Presentation:** The tool should present the results of the analysis clearly and intuitively, indicating whether the claim is determined to be true or false.

- **Scalability:** The tool should be scalable to accommodate a growing user base and increasing demand for fact-checking services.

2.1.3. Non-Functional Requirements:

- **Accuracy:** The tool should strive for high accuracy in its assessments to build trust and credibility among users.

- **Speed:** The processing time for analysing claims should be optimized to provide prompt responses to users.

- **Security:** Robust security measures should be implemented to safeguard user data and prevent unauthorized access.

- **Reliability:** The tool should demonstrate consistent performance and reliability in verifying factual claims.

- **Accessibility:** The interface should be accessible to users with diverse backgrounds and abilities, adhering to accessibility standards.

2.1.4. Constraints:

- **Technological Constraints:** The development of the tool may be limited by the availability of resources, expertise, and technology.

- **Data Availability:** The effectiveness of the tool may depend on the availability and quality of data for training and testing the AI algorithms.

- **Regulatory Compliance:** The tool should comply with relevant laws and regulations governing data privacy, security, and ethical use of AI technologies.

2.1.5. Assumptions:

- **User Proficiency:** It is assumed that users have basic proficiency in using digital tools and understanding the results provided by the "Genuine Information Finder."

- **Internet Access:** Users are assumed to have access to the internet to use the tool effectively.

2.2 Hardware and Software Requirements

2.2.1 For Users:

Hardware Requirements:

1. **Device:**
   - Any computer, laptop, tablet, or smartphone with internet connectivity.

2. **Internet Connection:**
   - A stable internet connection for accessing the "Genuine Information Finder" tool online.

Software Requirements:

1. **Operating System:**
   - Compatible with major operating systems such as Windows, macOS, Linux, iOS, and Android.

2. **Web Browser:**
   - Compatibility with popular web browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, etc.

2.2.2 For Developers:

Hardware Requirements:.

1. **Internet Connection:**
   - Stable internet connection for accessing online resources, code repositories, and collaboration tools.

2. **Processor:**
   - Quad-ore CPU (e.g., mid – range Intel core i5)
   - RAM – 2 GB
   - Hard disk – 25 GB

Software Requirements:

1. **Operating System:**
   - Windows, macOS, or Linux, depending on personal preference and development tools.

2. **Integrated Development Environment (IDE):**
   - PyCharm, Visual Studio Code, Sublime Text, or any preferred IDE for Python development.

3. **Version Control System:**
   - Git for version control and collaboration with team members.

4. **Python Development Tools:**
   - Python interpreter (latest version) and relevant libraries and frameworks such as NLTK, scikit-learn, TensorFlow, etc.

5. **Web Development Tools (Optional):**
   - HTML, CSS, JavaScript, and relevant frameworks for frontend development if working on web interfaces.

6. **Database Management System:**
   - SQLite, MySQL, PostgreSQL, or other databases as per project requirements for local development and testing.

2.3 Feasibility Study

A feasibility study is a critical step in assessing the viability of a project before committing significant resources to its development. It evaluates various aspects of the project, including technical, economic, legal, and operational factors, to determine whether the project is feasible and worth pursuing. Here is an outline of what a feasibility study for the "Genuine Information Finder" project might entail:

2.3.1. Technical Feasibility:

- **Assessment of Technology:** Evaluate the availability of AI algorithms, machine learning models, and natural language processing tools necessary for building GIF.

- **Technical Expertise:** Determine if the development team possesses the required skills and expertise to implement and maintain the technology stack.

- **Infrastructure Requirements:** Assess the technical infrastructure needed for hosting and running GIF, including servers, databases, and development environments.

2.3.2. Economic Feasibility:

- **Cost Estimation:** Estimate the development, deployment, and maintenance costs of GIF, including software development, infrastructure setup, and ongoing support.

- **Revenue Projection:** Analyse potential revenue streams, such as subscription fees, premium features, or partnerships, to determine the financial viability of the project.

- **Return on Investment (ROI):** Calculate the expected ROI based on cost estimates and revenue projections to assess the project's economic feasibility.

2.3.3. Market Feasibility:

- **Market Analysis:** Conduct market research to identify the demand for fact-checking tools and the competitive landscape.

- **User Needs Assessment:** Determine the target audience for GIF, their preferences, and willingness to pay for such a service.

- **Competitor Analysis:** Identify existing fact-checking tools and assess their strengths, weaknesses, and market positioning to identify opportunities and threats.

2.3.4. Legal and Regulatory Feasibility:

- **Compliance Assessment:** Evaluate legal and regulatory requirements related to data privacy, intellectual property rights, and ethical use of AI technologies.

- **Risk Analysis:** Identify potential legal risks and liabilities associated with GIF, such as data breaches, copyright infringement, or regulatory non-compliance, and develop mitigation strategies.

2.3.5. Operational Feasibility:

- **Operational Processes:** Define the operational processes and workflows required for developing, deploying, and maintaining GIF.

- **Organizational Readiness:** Assess the readiness of the organization to support the ongoing operation and support of GIF, including staffing, training, and infrastructure.

- **Risk Assessment:** Identify potential operational risks, such as scalability issues, user adoption challenges, or technical support requirements, and develop contingency plans.

2.3.6. Schedule Feasibility:

- **Project Timeline:** Develop a project timeline and schedule outlining key milestones, tasks, and dependencies.
- **Resource Allocation:** Evaluate the availability of resources, including personnel, funding, and technology, to determine the feasibility of meeting project deadlines.

- **Risk Management:** Identify potential schedule risks and develop mitigation strategies to minimize delays and ensure timely project delivery.

2.3.7. Risk Analysis:

- **Risk Identification:** Identify potential risks and uncertainties that may impact the success of GIF, including technical, economic, market, legal, and operational risks.

- **Risk Assessment:** Assess the likelihood and potential impact of each identified risk on the project's objectives and develop risk response strategies.

- **Risk Monitoring and Control:** Implement mechanisms to monitor and control project risks throughout the project lifecycle, including regular risk assessments and updates to risk management plans.

2.4 Product Functions

2.4.1 Fact Verification:

- Analyse textual claims or statements provided by users to determine their accuracy and veracity.

- Utilize AI algorithms and machine learning models to assess the credibility of information based on various factors such as source reliability, context, and supporting evidence.

2.4.2 User Interface:

- Provide a user-friendly interface for users to interact with the tool easily.

- Design intuitive input forms and result displays to facilitate seamless user experience.

2.4.3 Search and Input Mechanism:

- Enable users to input textual claims or statements through a search bar or text entry field.

- Support various input formats, including plain text, URLs, or snippets from social media platforms.

2.4.4 Analysis and Processing:

- Process the input data using natural language processing (NLP) techniques to extract key features and analyse the content.

- Apply machine learning algorithms such as logistic regression, decision trees, or neural networks to classify the veracity of the claims.

2.4.4 Result Presentation:

- Present the results of the analysis in a clear and understandable format, indicating whether the claim is determined to be true, false, or unverified.

- Provide additional context or evidence to support the classification decision and help users understand the reasoning behind it.

2.4.5 Integration with External Sources:

- Integrate with external databases, APIs, or fact-checking platforms to access additional sources of information and enhance the accuracy of the analysis.

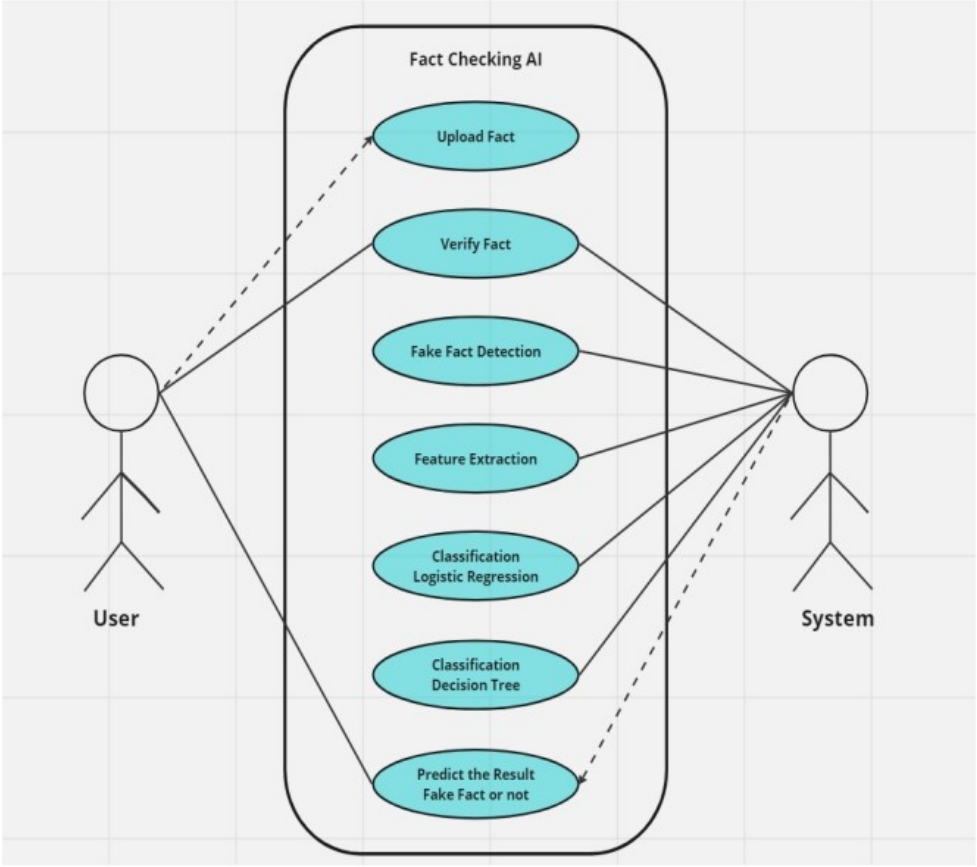
- Provide links or references to external sources to allow users to verify information independently.

2.4.6 Accessibility and Multi-platform Support:

- Ensure accessibility features to accommodate users with diverse needs and preferences.

- Support multi-platform access, including web browsers, mobile devices, and desktop applications.

2.5 Use -Case Diagram



Actors:

- **User:** This represents anyone who wants to use the system to check the validity of information.

System:

- **Fact Checking Al:** This represents the artificial intelligence engine behind the system that analyses information for factual accuracy.

Use Cases:

- **Upload Fact:** The user submits a piece of text or a file (presumably containing factual claims) for fact-checking.
- **Verify Fact:** The system analyses the uploaded content to determine its factual accuracy.

3. System Design

The System Architectural Design section is the cornerstone of the high-level conceptual design for the AI News Verification System. It provides a comprehensive view of the organization and 30 structure of the solution, allowing the reader to grasp the fundamental framework that underlies the entire system. The primary goals of this section are to elucidate the architectural components, their interactions, and the overall flow of data and processes within the system.

3.1 High-Level Design

The High-level Design Overview presents a conceptual view of the AI News Verification System, introducing key components and systems at a high level. This section provides a pictorial representation of the system architecture, illustrating the interactions between various tiers and components.

3.1.1 System Architecture

The AI News Verification System adopts a three-tier architecture, where each tier plays a distinct role in the overall functionality of the system.

3.1.1.1 Presentation Tier (Web Server)

At the topmost tier is the Presentation Tier, represented by the Web Server. This tier encompasses the user interface and the web server environment. The web server utilizes Internet Information Services (IIS) with the ASP.NET runtime, providing a platform for users to interact with the system through a web interface. Users submit news articles, request verification results, and interact with the system's functionalities through this tier.

3.1.1.2 Application Tier (Verification Engine)

The middle tier is the Application Tier, where the Verification Engine resides. This tier is responsible for processing user requests, managing the core verification algorithms, and orchestrating the flow of information between the Presentation Tier and the Data Tier. The

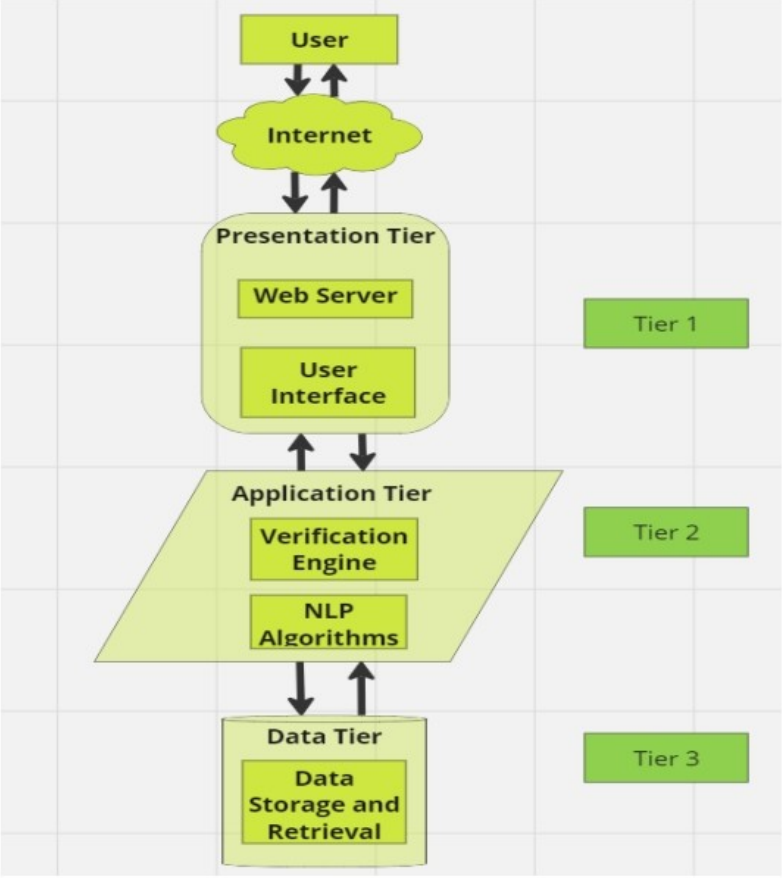
Verification Engine incorporates Natural Language Processing (NLP) and image processing algorithms for analysing and verifying news articles.

3.1.1.3 Data Tier (SQL Server)

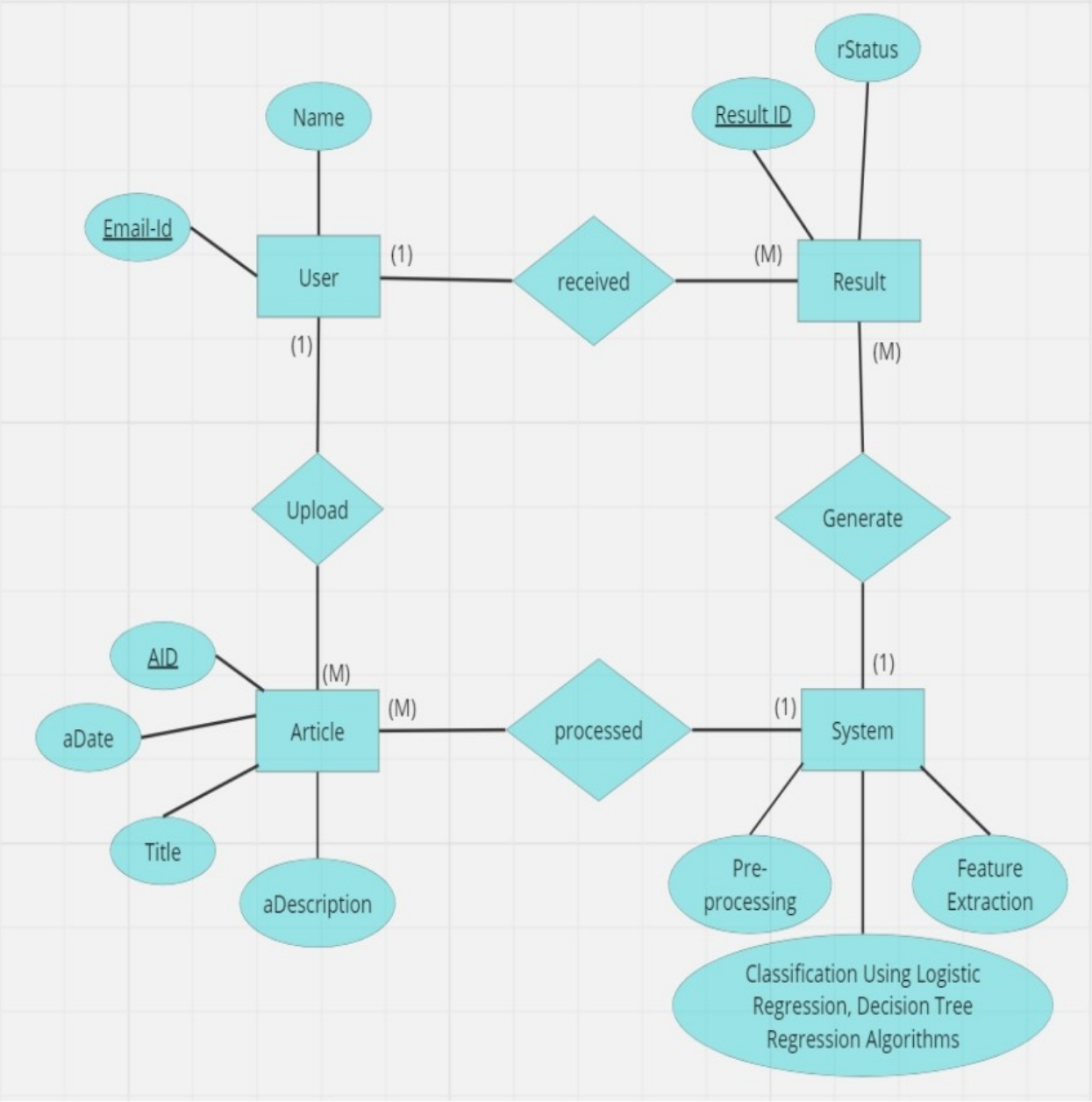
At the bottommost tier is the Data Tier, represented by SQL Server. This tier is dedicated to the storage and retrieval of data essential to the system. It includes user information, news articles, and verification results. The SQL Server ensures data integrity, persistence, and efficient retrieval, supporting the overall functionality of the AI News Verification System.

3.1.2 Pictorial Representation

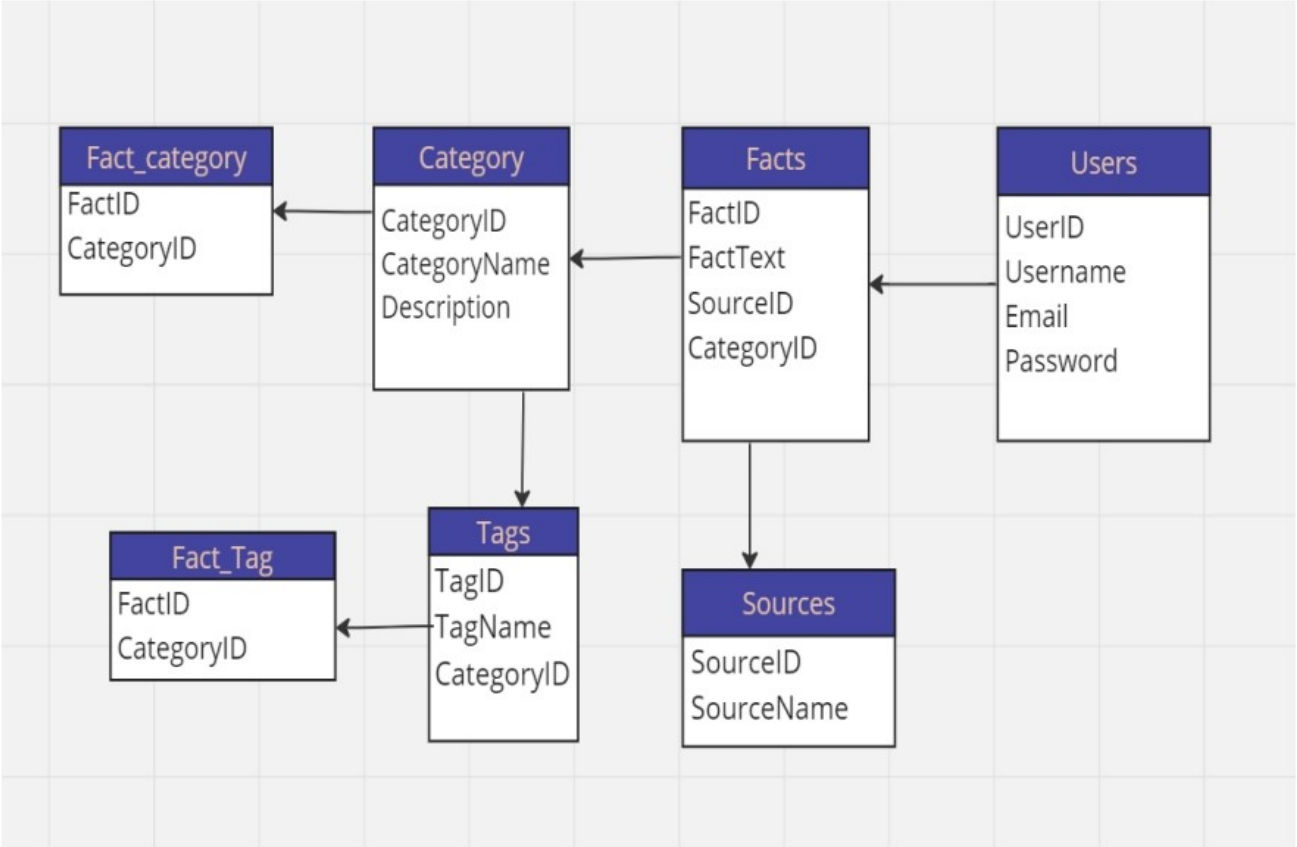
The pictorial representation illustrates the connectivity and interaction patterns between the three tiers of the system. Users connect to the system through the Web Server, which forwards user requests and data to the Verification Engine in the Application Tier. The Verification Engine processes the requests using sophisticated algorithms and interacts with the SQL Server in the Data Tier for data storage and retrieval.



3.2 E-R Diagram

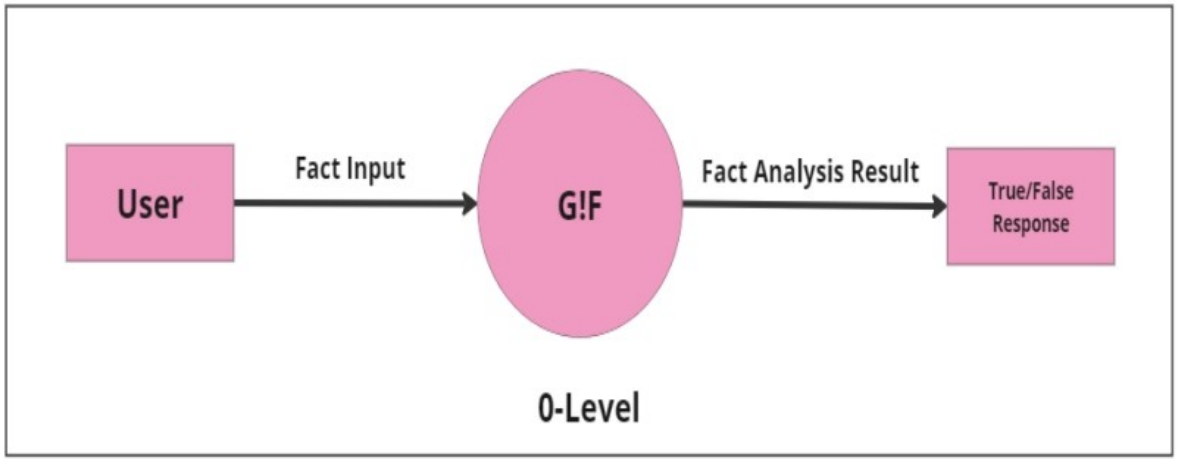


3.3 Database Design

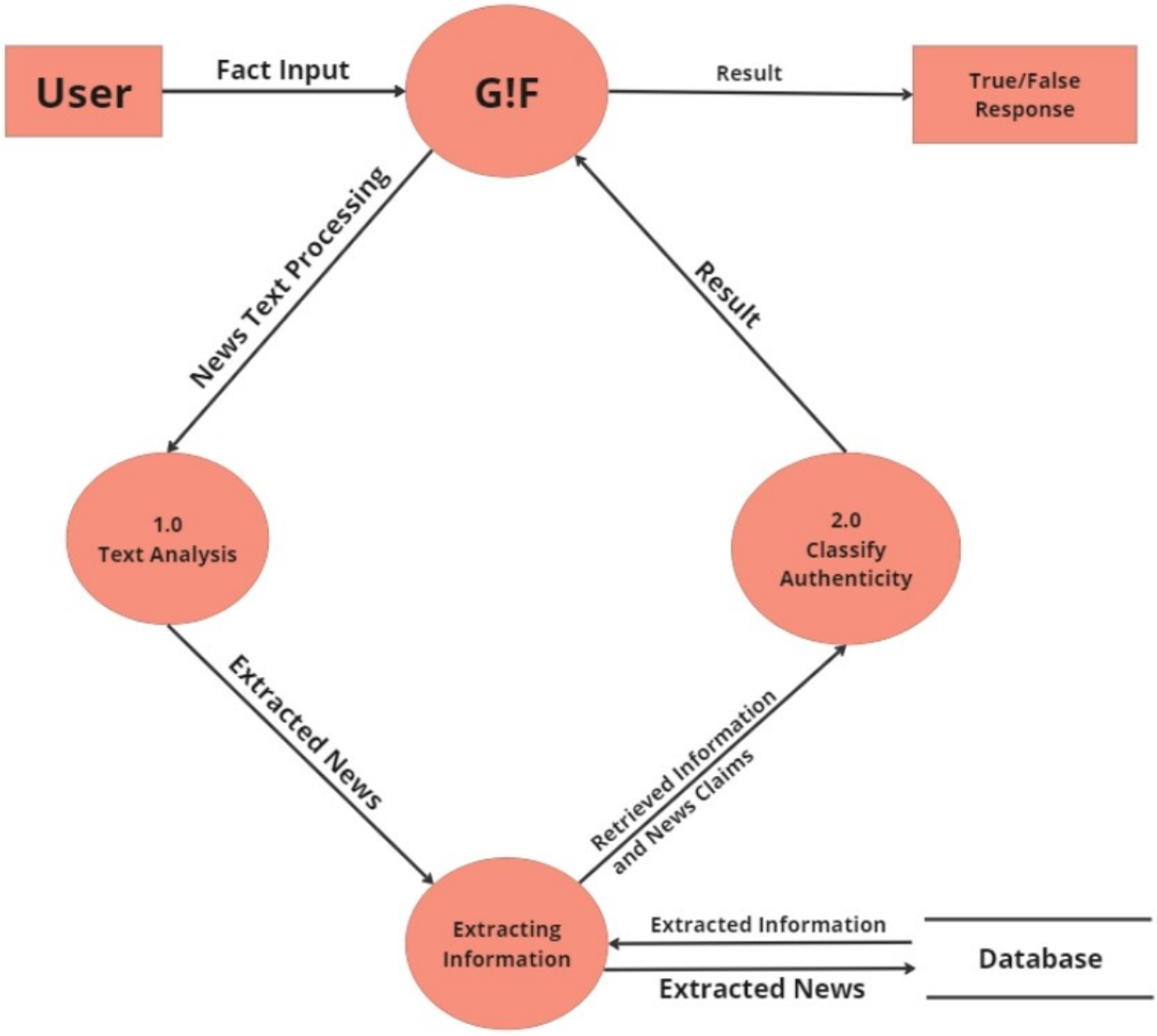


3.4 DFD

3.4.1 Level – 0

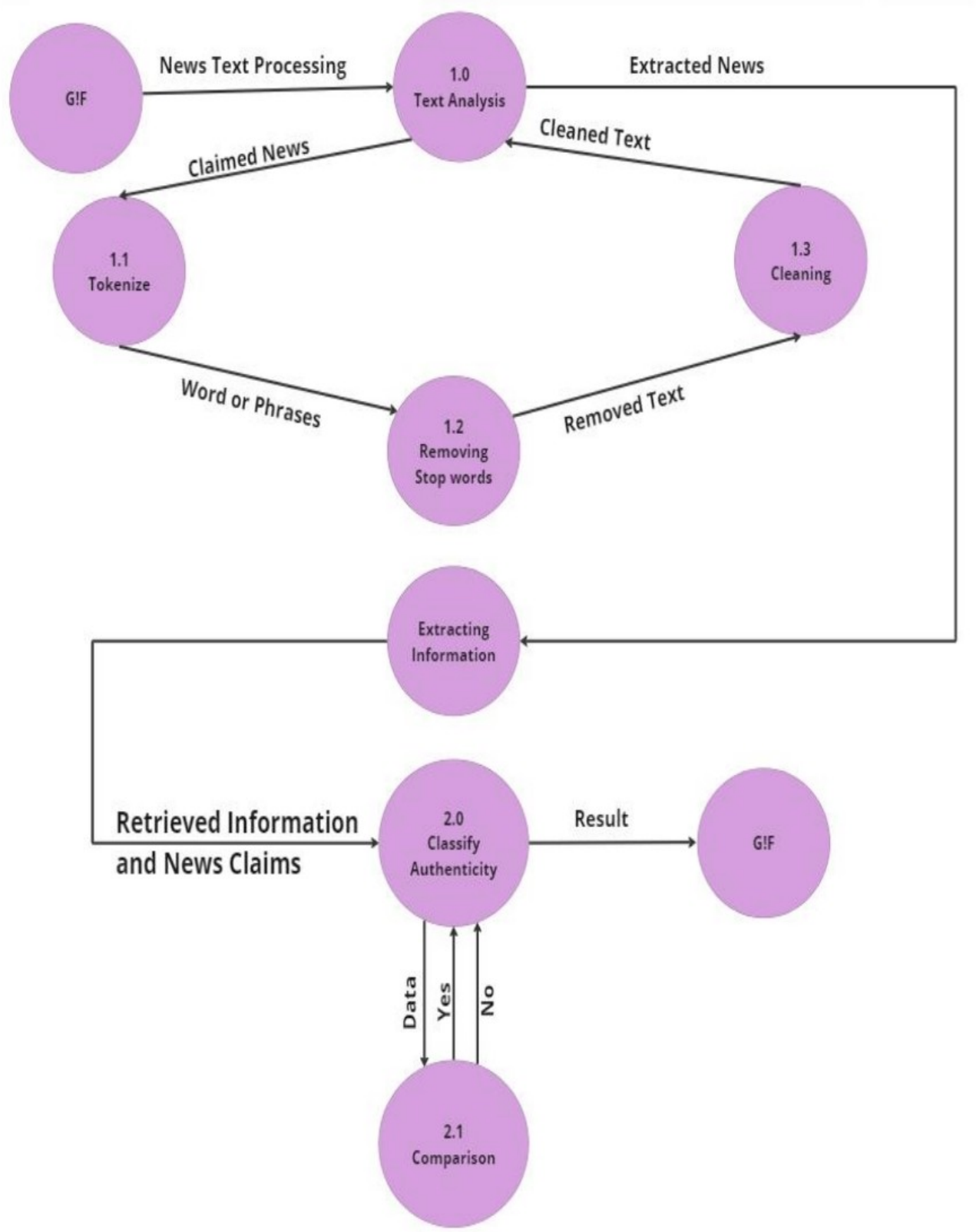


3.4.2 Level – 1



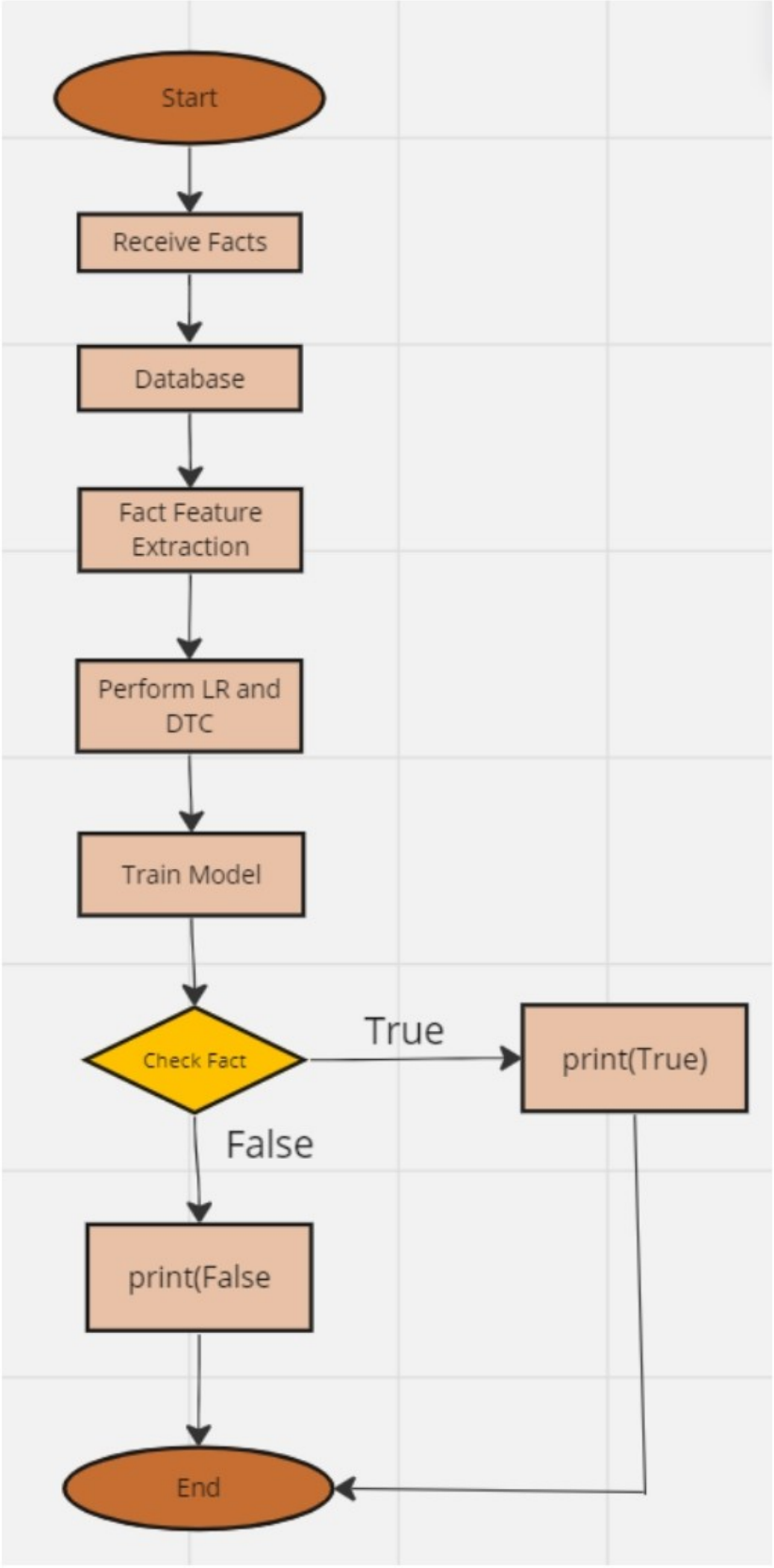
1-Level

3.4.3 Level – 2



2-Level

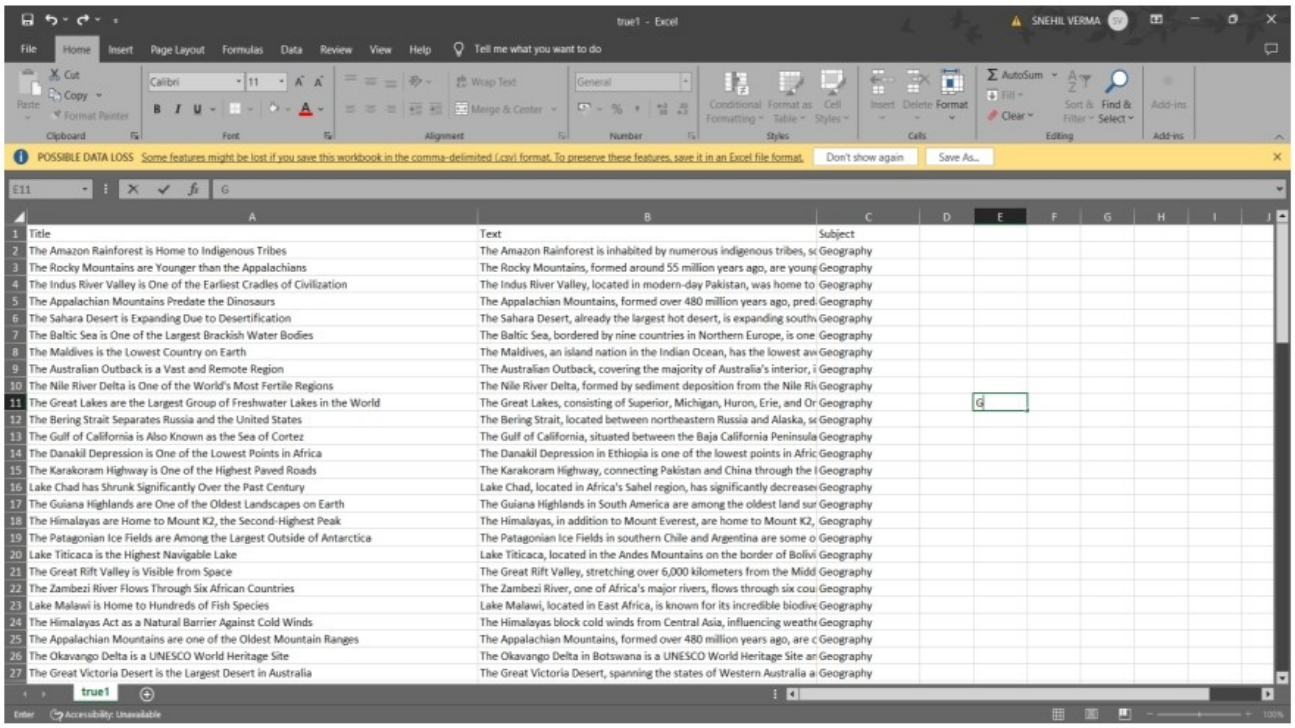
3.5 Flowchart



4. Coding

4.1 Methodology

4.1.1 The Dataset



	Title	Text	Subject
2	The Amazon Rainforest is Home to Indigenous Tribes	The Amazon Rainforest is inhabited by numerous indigenous tribes, s	Geography
3	The Rocky Mountains are Younger than the Appalachians	The Rocky Mountains, formed around 55 million years ago, are young	Geography
4	The Indus River Valley is One of the Earliest Cradles of Civilization	The Indus River Valley, located in modern-day Pakistan, was home to	Geography
5	The Appalachian Mountains Predate the Dinosaurs	The Appalachian Mountains, formed over 480 million years ago, pred	Geography
6	The Sahara Desert is Expanding Due to Desertification	The Sahara Desert, already the largest hot desert, is expanding south	Geography
7	The Baltic Sea is One of the Largest Brackish Water Bodies	The Baltic Sea, bordered by nine countries in Northern Europe, is one	Geography
8	The Maldives is the Lowest Country on Earth	The Maldives, an island nation in the Indian Ocean, has the lowest av	Geography
9	The Australian Outback is a Vast and Remote Region	The Australian Outback, covering the majority of Australia's interior, i	Geography
10	The Nile River Delta is One of the World's Most Fertile Regions	The Nile River Delta, formed by sediment deposition from the Nile Ri	Geography
11	The Great Lakes are the Largest Group of Freshwater Lakes in the World	The Great Lakes, consisting of Superior, Michigan, Huron, Erie, and On	Geography
12	The Bering Strait Separates Russia and the United States	The Bering Strait, located between northeastern Russia and Alaska, s	Geography
13	The Gulf of California is Also Known as the Sea of Cortez	The Gulf of California, situated between the Baja California Peninsula	Geography
14	The Danakil Depression is One of the Lowest Points in Africa	The Danakil Depression in Ethiopia is one of the lowest points in Afric	Geography
15	The Karakoram Highway is One of the Highest Paved Roads	The Karakoram Highway, connecting Pakistan and China through the I	Geography
16	Lake Chad has Shrunk Significantly Over the Past Century	Lake Chad, located in Africa's Sahel region, has significantly decreases	Geography
17	The Guiana Highlands are One of the Oldest Landscapes on Earth	The Guiana Highlands in South America are among the oldest land sur	Geography
18	The Himalayas are Home to Mount K2, the Second-Highest Peak	The Himalayas, in addition to Mount Everest, are home to Mount K2,	Geography
19	The Patagonian Ice Fields are Among the Largest Outside of Antarctica	The Patagonian Ice Fields in southern Chile and Argentina are some o	Geography
20	Lake Titicaca is the Highest Navigable Lake	Lake Titicaca, located in the Andes Mountains on the border of Bolivi	Geography
21	The Great Rift Valley is Visible from Space	The Great Rift Valley, stretching over 6,000 kilometers from the Midd	Geography
22	The Zambezi River Flows Through Six African Countries	The Zambezi River, one of Africa's major rivers, flows through six cou	Geography
23	Lake Malawi is Home to Hundreds of Fish Species	Lake Malawi, located in East Africa, is known for its incredible biodiv	Geography
24	The Himalayas Act as a Natural Barrier Against Cold Winds	The Himalayas block cold winds from Central Asia, influencing weathe	Geography
25	The Appalachian Mountains are one of the Oldest Mountain Ranges	The Appalachian Mountains, formed over 480 million years ago, are c	Geography
26	The Okavango Delta is a UNESCO World Heritage Site	The Okavango Delta in Botswana is a UNESCO World Heritage Site an	Geography
27	The Great Victoria Desert is the Largest Desert in Australia	The Great Victoria Desert, spanning the states of Western Australia a	Geography

The dataset is simple. It contains the titles of the news, the body text and a subject field. There are 3 main segments of the methodology:

- The core Machine Learning model.
- The web interfaces.
- The common platform that brings the model and the interface together

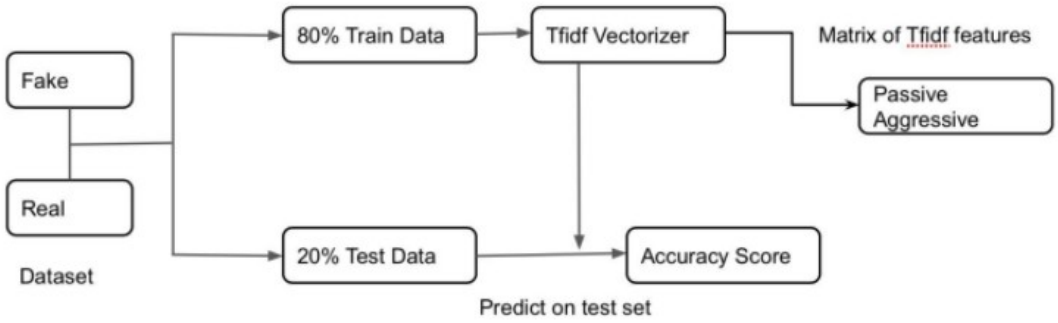
4.1.2 The Machine Learning Model

There are two parts to the ML Model building. Machine Learning is a part of our life that can help us in predicting. We are using two types of models in this case. For the first part, we used passive-aggressive classifiers. And the steps include:

1. **Data Loading:** We are loading a CSV file for the data sorting and training-testing part of the model. The CSV file is turned into an array for easier work purpose.

2. **Vectorization:** Vectorization is needed for determining the frequency of the words present in a passage. This is needed to determine which words are used often.

3. **Classifier:** Passive-aggressive algorithms are a family of great learning algorithms. They are like Perceptron because it does not require a reading scale. However, unlike Perceptron, they include parameter correction. Passive is used when the prediction is correct and there is no change in the model. But if there is any kind of change in the model, that is if the prediction is not correct then the aggressive part is called, which changes the model accordingly. The aggressive part of the model changes the model according to its wish on the backend.



4. **Model Building:** The model is built through the train and test of the dataset, by ensuring that the training is done for 80% of the dataset and testing is done in the rest of the 20% of the dataset.

In the second part, we used is LSTM. Here are the steps:

1. **Loading the data:** For this step, it is the same as the passive-aggressive one.

2. **Scanning and parsing:** Data is loaded from a CSV file. This consists of the body of selected news articles. In this code block, we scan the CSV and clean the titles to filter out stop words and punctuation.

3. **Tokenization:** The tokenizer is used to assign indices to words, and filter out infrequent words. This allows us to generate sequences for our training and testing data.

4. **Embedding matrix:** Apply the embedding matrix. An embedding matrix is used to extract the semantic information from the words in each title.

5. **Model Building:** Building the model and finding out the accuracy via confusion matrix. The model is created using an Embedding layer, LSTM, Dropout, and Dense layers. We are going to run the data on 20 epochs. We observed that the LSTM model is vastly inaccurate in predicting the authenticity of the news. So, we decided to show the output by running it through the Logistic Regression and Decision Tree Classifier model

4.1.3 The Web Interface

1. **HTML for building the basic skeleton:** HTML makes the structure of the web application and there are some of the functions that can be achieved best with HTML only.
2. **CSS for design:** The CSS part is for designing only. Because it will give a more beautiful aspect to the website.

4.1.4 Common Platform: Flask

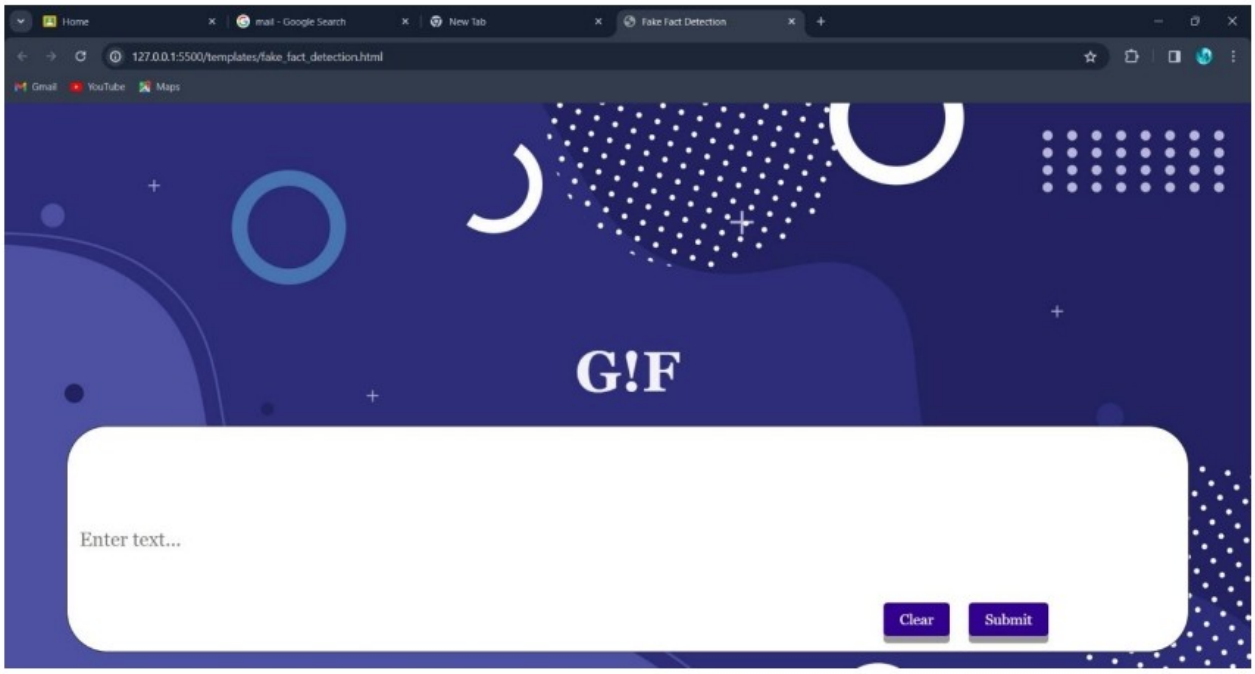
This acts as a common platform and takes the input with the pickle module and passes it to the machine learning model afterwards the prediction is shown on the screen with the HTML and CSS website.

1. Building functions for taking input.
2. Passing input values through the ML model.
3. Using the Pickle module for serializing and de-serializing the dataset.
4. Providing output.

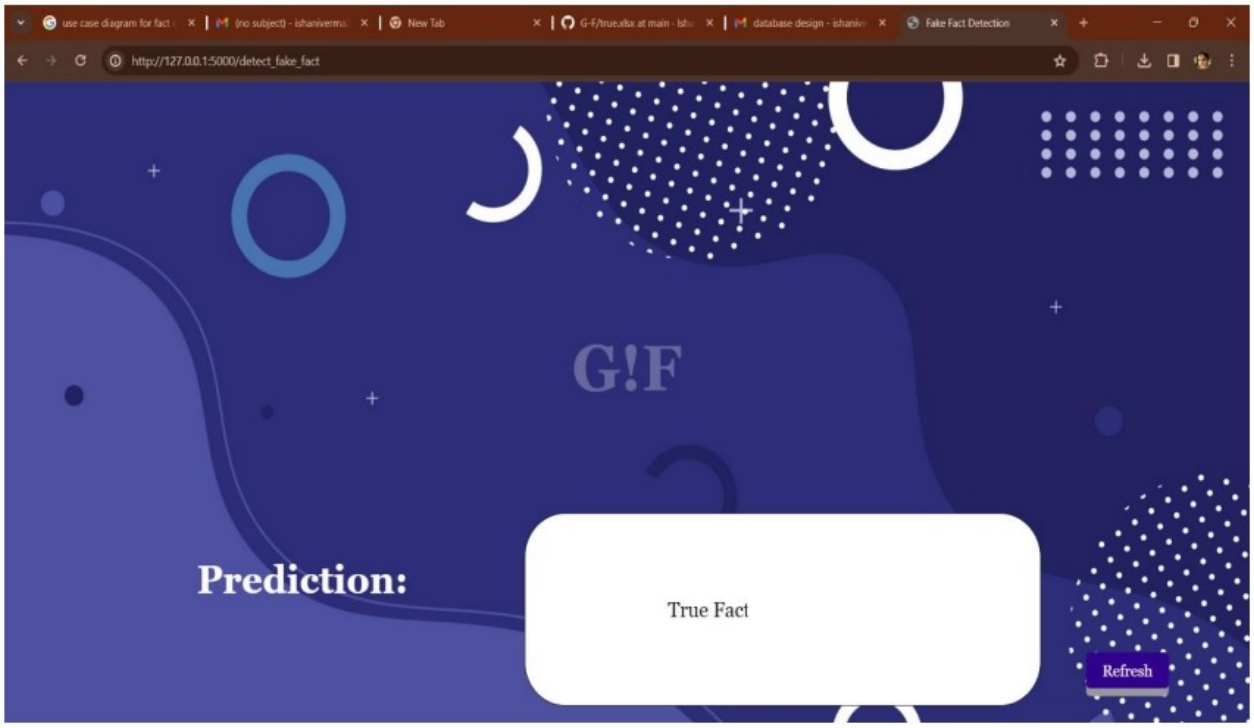
4.2 Implementation

4.2.1 The Interface

This is what you see when you go to the web interface. You are supposed to copy the news and paste it into the input box.



When you paste the fact on the input box and click 'Submit' the model will give you the result. If the fact seems authentic, the output will be 'True Fact'. Otherwise, it will show 'Fake Fact'. That's how you can detect fake or real fact via the interface



4.2.2 The ML Model

The code for the ML model building is as follows:

TF-IDF stands for Term Frequency-Inverse Document Frequency. Term frequency is basically a ratio of the number of times a particular word appears with respect to the total number of words. And Inverse Document Frequency is basically the weight of a rare word.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

Implementing Logistic Regression and Decision Tree Regression Classifier

```
import pandas as pd
import numpy as np
# import seaborn as sns
# import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

```
import re
import string

data_fake = pd.read_csv('D:/BANASTHALI VIDYAPITH/Inhouse project/fake1.csv')
data_true = pd.read_csv('D:/BANASTHALI VIDYAPITH/Inhouse project/true1.csv')
data_fake.head()
data_true.head()

data_fake['class'] = 0
data_true['class'] = 1

data_fake_manual_testing = data_fake.tail(10)
for i in range(98,88,-1):
    data_fake.drop([i], axis = 0, inplace = True)

data_true_manual_testing = data_true.tail(10)
for i in range(98, 88,-1):
    data_true.drop([i], axis = 0, inplace = True)

data_fake_manual_testing['class'] = 0
data_true_manual_testing['class'] = 1

data_merge = pd.concat([data_fake, data_true], axis = 0)
data_merge.head(10)

data = data_merge.drop(['Title', 'Subject'], axis = 1)

def wordopt(Text):
    Text=Text.lower()
    Text=re.sub('\[.*?\]',",Text)
    Text=re.sub("\\W","",Text)
    Text=re.sub('https?://\S+|www\.\S+',",Text)
    Text=re.sub('<.*?>+',",Text)
    Text=re.sub('[%s]'%re.escape(string.punctuation),",Text)
    Text=re.sub('\n',",Text)
```

```
    Text=re.sub('\w*\d\w*','',Text)
    return Text

data['Text'] = data['Text'].apply(wordopt)

x = data['Text']
y = data['class']

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.25)

from sklearn.feature_extraction.text import TfidfVectorizer
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)

from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(xv_train, y_train)

pred_lr = LR.predict(xv_test)
LR.score(xv_test, y_test)
print(classification_report(y_test, pred_lr))

from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)

pred_dt = DT.predict(xv_test)
DT.score(xv_test, y_test)
print(classification_report(y_test, pred_dt))
def output_lable(n):
    if n == 0:
        return "Fake Fact"
    elif n == 1:
        return "True Fact"
```

```
def manual_testing(news):
    testing_news = {"Text":[Fact]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["Text"] = new_def_test["Text"].apply(wordopt)
    new_x_test= new_def_test["Text"]
    new_xv_test =vectorization.transform(new_x_test)
    pred_LR =LR.predict(new_xv_test)
    pred_DT= DT.predict(new_xv_test)
    return print("\n\nLR Prediction: {} \nDT Prediction:".format(output_lable(pred_LR[0]),
output_lable(pred_DT[0])))
Fact=str(input())
manual_testing(Fact)
```

4.2.3 Flask Code

```
from flask import Flask, render_template, request, redirect, url_for, jsonify
app = Flask(__name__)

# Define routes
@app.route('/')
def index():
    return render_template("qqq.html")

@app.route('/explore')
def explore():
    return render_template("explore.html")

@app.route('/fake_fact_detection')
def fake_fact_detection():
    return render_template("fake_fact_detection.html")

@app.route('/detect_fake_fact', methods=['POST'])
def detect_fake_fact():
    if request.method == 'POST':
        fact = request.form['fact_input']
        # Perform your fake fact detection process here
```

```
        result = manual_testing(fact)  # Replace this with your actual result
        return render_template("output.html", result=result)
    return render_template("output.html")

@app.route('/get_new_facts')
def get_new_facts():
    data = pd.read_excel('D:/BANASTHALI VIDYAPITH/Inhouse project/true1.xlsx',
engine='openpyxl')
    # # Convert the data to a list of dictionaries
    # new_facts = data.sample(n=10)['Text'].tolist()
    # return jsonify({'facts': new_facts})
    new_facts = data.sample(n=10)['Text'].tolist()
    return jsonify({'facts': new_facts})

if __name__ == '__main__':
    app.run(debug=True)
```

4.2.5 Web Interface

```
//fake_fact_detection.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fake Fact Detection</title>
    <link rel="stylesheet" type="text/css" href="/static/fake_fact_detection.css">
</head>
<body>

    <h1 id="gif-text">G!F</h1>

  <div class="container">
    <form id="fake-fact-form" action="/detect_fake_fact" method="POST">
      <input type="hidden" id="hidden-fact-input" name="fact_input" value="">
```

```
    <input type="text" id="fact-input" placeholder="Enter text...">
    <br>
    <button id="submit-btn" type="submit">Submit</button>
    <button id="clear-btn">
      <a href="/fake_fact_detection">Clear</a>
    </button>
  </form>
  <button class="home-button" type="submit">
    <a href="/">Home</a>
  </button>




  </div>

  <script src="/scripts/app.js"></script>
</body>
</html>
//fake fact detection.css
body {

    margin: 0;
    padding: 0;

    position: relative;
    display: flex;

    background-size: cover;
    background-position: center top;
    background-color: #000000;
    background-image: url('bg2.jpg');
    min-height: 100vh;
    align-items: center;
    justify-content: center;
```

```
}


* {
position: relative;
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: Georgia;
}

.container {
  display: grid;
  place-items: center;
  text-align: center;
  height: 100vh
}

.title {
  font-family: "Montserrat", sans-serif;
  font-weight: 800;
  font-size: 8.5vw;
  text-transform: uppercase;
  font-size: 60px;
  text-align: center;
  position: absolute;
  top: 10px;
  left: 50%;
  transform: translateX(-50%);
}

h1 {
  font-size: 100px;
  margin-bottom: 100px;
```

```
}

#fact-input {
  width: 90%;
  position: fixed;
  bottom: 0;
  height: 40%;
  padding: 15px;
  margin-top: 20px;
  margin-bottom: 20px;
  font-size: 24px;
  border: 1px solid #060606;
  border-radius: 50px;
  box-sizing: border-box;
  background-color: white;
  text-align: left;
  resize: none;
  transition: width 0.4s ease-in-out;
}

#fact-input2 {
width: 60%;
position: absolute;
bottom: 0;
margin-left:700;
height: 30%;
padding: 15px;
margin-top: 20px;
margin-bottom: 20px;
font-size: 24px;
border: 1px solid #060606;
border-radius: 50px;
box-sizing: border-box;
background-color: white;
text-align: left;
```

```
resize: none;
transition: width 0.4s ease-in-out;
right: 0 !important;
margin-left: 20px;
margin-right: 20px;
}

#clear-btn,#refresh-btn,
#submit-btn
{

  margin-bottom: 20px;
  position: fixed;
  padding: 10px 20px;
  font-size: 18px;
  background-color: rgb(144, 132, 165);
  color: #fff;
  border: 0ch;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
  bottom: 20px;
  box-shadow: 0 9px #999;
}

#clear-btn {
  margin-top: 0px;
  margin-left: 20px;
  margin-right: 20px;
  right: calc(200px + 4cm);
}

#submit-btn {
  right: 250px;
}
```

```
#refresh-btn {
  right: 100px;
  margin-top: 20px;
}

#submit-btn:hover,
#clear-btn:hover, #refresh-btn:hover {
  background-color: rgb(113, 101, 171);
  box-shadow: 0 12px 16px 0 rgba(0, 0, 0, 0.24), 0 17px 50px 0 rgba(0, 0, 0, 0.19);
}

#gif-text {
font-size: 60px;
text-align: center;
position: absolute;
top: 250px;
margin-top: 40px;
transform: translateX(-50%);
animation: slideIn 1s ease-in-out infinite alternate;
color: white;
font-size: 10vh;

}
@keyframes slideIn {
  0% {
     transform: translateY(-10px);
     opacity: 0;
  }
  100% {
     transform: translateY(0);
     opacity: 1;
  }
}
.pred-container {
```

```
justify-content: flex-end;
align-items: flex-start;
height: 60vh;
position: absolute;

top: 300px;
}

#pred {
font-size: 45px;
color: rgb(55, 31, 190);
text-shadow: 2px 2px 4px white(144, 135, 135, 0.5);
margin-right: 800px;
margin-top:500px;
}

.home-button {
  background-color: rgb(144, 132, 165);
  border: none;
  color: #130533;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  font-size: 16px;
  cursor: pointer;
  border-radius: 5px;
  box-shadow: 0 9px #999;
  color: rgb(1, 1, 1);
  position: absolute;
  right:    100px;
 bottom: 40px;


}
```

```
 .home-button:hover
{
 background-color: rgb(113, 101, 171);
 box-shadow: 0 12px 16px 0 rgba(0, 0, 0, 0.24), 0 17px 50px 0 rgba(0, 0, 0, 0.19);
}

//app.js
document.getElementById('clear-btn').addEventListener('click', function() {
   document.getElementById('fact-input').value = '';
});



document.getElementById('submit-btn').addEventListener('click', function () {
   // Redirect to another page
   window.location.href = 'output.html';
});

const homeButton = document.querySelector('.home-button');

// Attach a click event listener to the button
homeButton.addEventListener('click', function() {
 // This function will be executed when the button is clicked
 window.location.href = 'qqq.html';
 // You can add your desired functionality here
});

//output.html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Fake Fact Detection</title>
   <link rel="stylesheet" type="text/css" href="/static/fake_fact_detection.css">
```

```
</head>
<body>

    <h1 id="gif-text">G!F</h1>
    <div class="container">


        <h2 id="pred">Prediction:</h2>
        <input type="text" id="fact-input2" placeholder="output...." value="{{ result }}"
readonly>
        <br>
        <button id="refresh-btn">
            <a href="/fake_fact_detection">Refresh</a>
        </button>
    </div>


    <script src="/scripts/output.js"></script>
</body>
</html>

//output.css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;

    position: relative;
    display: flex;
    background-image: url("fact.jpg");
    background-size: cover;
    background-position: center top;
    background-color: rgba(70, 63, 63, 0.5);

}
```

```
h1 {
    font-size: 1000px;
    margin-bottom: 20px;
}

#fact-input {
    width: 60%;
    position: fixed;
    bottom:0;
    height: 30%;
    padding: 15px;
    margin-top: 20px;
    margin-bottom: 20px;
    font-size: 24px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-sizing: border-box;
    background-color: rgba(0, 0, 0, 0.5);
    text-align: left;
}
#fact-input::placeholder {
    text-align: left;

}

 /* #clear-btn {
    margin-top: 20px;
    margin-bottom: 20px;
    position: fixed;
    padding: 10px 20px;
    font-size: 18px;
    background-color: black;
    color: #fff;
```

```
    border:0ch;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    bottom: 0;
    right: 200px;
    padding :15px;
} */
#refresh-btn
{
    margin-top: 20px;
    margin-bottom: 20px;
    position: fixed;
    padding: 10px 20px;
    font-size: 18px;
    background-color: black;
    color: #fff;
    border:0ch;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    bottom: 0;
    left:100ch;
    padding :15px;
    transition-duration: 0.4s;
    box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0 rgba(0,0,0,0.19);
}
#refresh-btn:hover {
    background-color: black; /* Green */
    box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
    color: white;
  }

#submit-btn:hover, #clear-btn:hover {
    background-color:black;
```

```
}

#gif-text {
    font-size: 60px;
  text-align: center; /* Align the heading text to the center */
  position: absolute; /* Position the heading */
  top: 10px; /* Set the distance from the top of the viewport */
  left: 50%; /* Align the heading horizontally in the middle */
  transform: translateX(-50%);
}
/* #plus-btn {
    padding: 10px 20px;
    font-size: 18px;
    background-color: black;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    position: absolute;
    top: 20px;
    right: 80px;
}

#equals-btn {
    padding: 10px 20px;
    font-size: 18px;
    background-color:black;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    position: absolute;
    top: 20px;
```

```
    right: 20px; /* Adjusted position */
 */

#user-profile-icon {
    padding: 10px 20px;
    margin-top: 20px;
    margin-bottom: 20px;
    position: fixed; /* Position the image relative to the viewport */
    bottom: 0; /* Align the image to the bottom of the viewport */
     right: 0;
    width: 80px; /* Adjust icon size as needed */
    height: 40px; /* Adjust icon size as needed */
}

#pred {
    font-family: Arial, sans-serif; /* Change the font family as desired */
    font-size: 60px; /* Change the font size as desired */
    color: rgb(249, 244, 244); /* Change the text color as desired */
    margin-top: 300px; /* Adjust the top margin as desired */
}
#output-box::placeholder {
    text-align: left;

}

//output.js
document.addEventListener('DOMContentLoaded', function() {
    // Retrieve the result from the input box and display it in the fact-input2 box
    var result = document.getElementById('fact-input2').value;
    // Set the value of fact-input2
    document.getElementById('fact-input2').value = result;

    // Event listener for refreshing the page
    document.getElementById('refresh-btn').addEventListener('click', function () {
        // Redirect to the first page
```

```
        window.location.href = '/';
    });
});

//qqq.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fact or Fiction</title>

      <link rel="stylesheet" type="text/css" href="/static/qqq.css">

</head>
<body>
    <h1>G!F</h1>
    <div class="container">

        <div class="left">
            <div>

                <h2>Fact or Fiction?</h2>
                <p>Let's Play Truth or Dare with Reality! Join the adventure where facts are put to
the test.</p>
                <p>Are you ready to uncover the real story?</p>
            </div>
        </div>
        <img src="/static/images/fact.jpg" alt="An example image">
        <div class="right">
            <button id="check-btn" type="submit">
                <a href="/fake_fact_detection">Check</a>
            </button>
            <!-- <a href="/fake_fact_detection">Fake Fact Detection</a> -->
            <button id="Explore-btn">
```

```
            <a href="/explore">Explore</a>
        </button> <!-- Added About button -->
        <!-- <a href="/explore">Fake Fact Detection</a> -->
      </div>
  </div>

  <script src="/static/script.js"></script>
</body>
</html>

//qqq,css
body {
  margin: 0;
  padding: 0;
  font-family: Georgia;

  background-size: cover;
  background-position: center;
}
.container {
  display: flex;
  height: 100vh;
  background-color:#d5def5; /* Light blue background with opacity */
}
.left {
  flex: 1;
  display: flex;
  justify-content: center;
  align-items: center;
  text-align: center;
  padding: 20px;
  color: rgb(11, 7, 85);
}
.right {
  flex: 1;
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    text-align: center;
    padding: 20px;
}

h1{
    font-size: 10vh; /* Using viewport height for responsive sizing */
    color: rgb(68, 37, 121);
    text-align: center;
    position: absolute;
    top: 16px; /* Adjust the top position as needed */
    left: 45%;
    transform: translateX(-50%);
    animation: slideIn 1s ease-in-out infinite alternate;
  }

  @keyframes slideIn {
    0% {
      transform: translateY(-10px);
      opacity: 0;
    }
    100% {
      transform: translateY(0);
      opacity: 1;
    }
  }


  @keyframes slideIn {
    0% {
        transform: translateY(-10px);
        opacity: 0;
    }
```

```
    100% {
        transform: translateY(0);
        opacity: 1;
    }
  }

//qqq.js
document.getElementById('check-btn').addEventListener('click', function () {
    // Redirect to another page
    window.location.href = 'fake_fact_detection.html';
});
document.getElementById('Explore-btn').addEventListener('click', function () {
    // Redirect to another page
    window.location.href = 'explore.html';
});

//explore.html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Explore</title>
<link rel="stylesheet" type="text/css" href="/static/explore.css">
</head>
<body>
<div class="container">
    <h1>Do you Know?</h1>
    <textarea id="facts-textarea" readonly>

    </textarea>
    <button class="new-button" type="submit">
      <a href="/explore">New</a>
    </button></div>
    <script>
```

```
    document.addEventListener('DOMContentLoaded', function() {
        // Function to fetch new facts
        function fetchNewFacts() {
            fetch('/get_new_facts')
                .then(response => response.json())
                .then(data => {
                    // Extract the facts from the response
                    const facts = data.facts;

                    // Update the textarea with new facts
                    const textarea = document.getElementById('facts-textarea');
                    textarea.value = facts.map(fact => '- ' + fact).join('\n');
                })
                .catch(error => console.error('Error fetching new facts:', error));
        }

        // Fetch new facts when the page loads
        fetchNewFacts();

        // Add event listener for the "New" button
        const newFactsBtn = document.getElementById('new-button');
        newFactsBtn.addEventListener('click', fetchNewFacts);
    });
  </script>
    <button class="home-button" type="submit">
     <a href="/">Home</a>
    </button>
</body>
</html>

//explore.css
body {
    font-family: Georgia;
    margin: 0;
    padding: 0;
```

```
    background-color:#d5def5; /* Set background color to lavender */
}
.container {
    height: 80vh; /* Set container height to 80% of the viewport height */
    max-width: 800px;
    margin: 10vh auto; /* Center vertically within the viewport */
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    background-color: #fff; /* Set background color to white */
    display: flex;
    flex-direction: column;
    align-items: center;
}
h1 {
    text-align: center;
    font-size: 40px;
    margin-bottom: 30px;
    color: rgba(12, 12, 79, 0.758); /* Set heading color to blue */
}
textarea {
    width: 90%;
    max-width: 800px;
    height: 60%;
    padding: 8px;
    border-radius: 5px;
    border: 1px solid #ccc;
    resize: none;
    font-size: 20px;
    line-height: 1.6;
    margin-bottom: 20px; /* Add some margin at the bottom */
    font-family: Georgia;
}
.new-button {
```

```
    background-color: rgb(144, 132, 165);
    border: none;
    color: #130533;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    font-size: 16px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 9px #999;
    color: white;
    display: flex;
}

.new-button:hover
 {
  background-color: rgb(113, 101, 171);
  box-shadow: 0 12px 16px 0 rgba(0, 0, 0, 0.24), 0 17px 50px 0 rgba(0, 0, 0, 0.19);
}

.home-button {
    background-color: rgb(144, 132, 165);
    border: none;
    color: #130533;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    font-size: 16px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 9px #999;
    color: white;
    position: absolute;
    right:    100px;
   bottom: 40px;
```

```
 }
 .home-button:hover
{
 background-color: rgb(113, 101, 171);
 box-shadow: 0 12px 16px 0 rgba(0, 0, 0, 0.24), 0 17px 50px 0 rgba(0, 0, 0, 0.19);
}
```

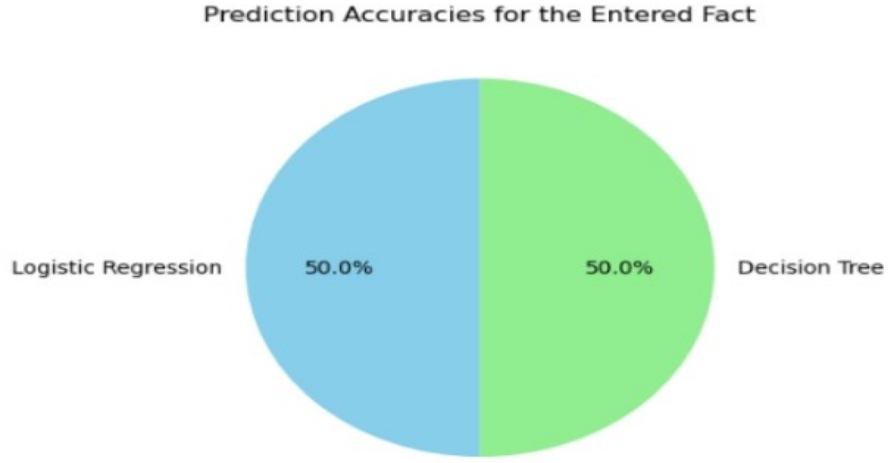
5. Testing

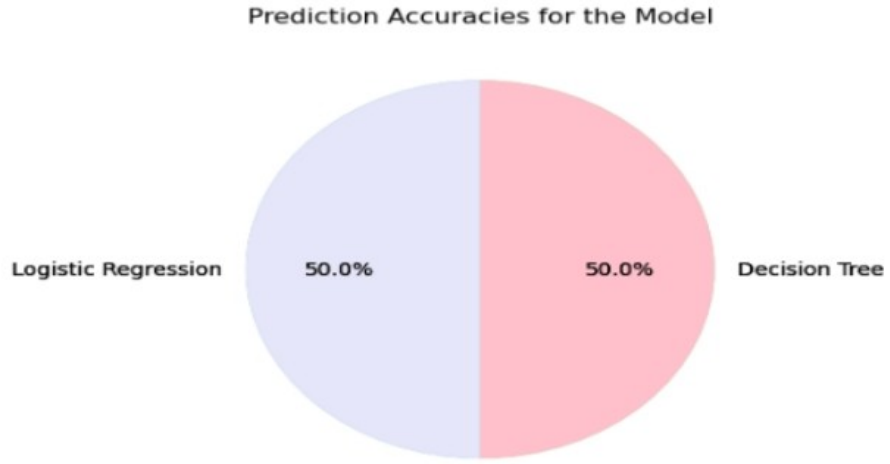
```
def manual_testing(news):
    testing_news = {"Text": [news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["Text"] = new_def_test["Text"].apply(wordopt)
    new_x_test = new_def_test["Text"]
    new_xv_test = vectorization.transform(new_x_test)
    pred_LR = LR.predict(new_xv_test)
    pred_DT = DT.predict(new_xv_test)
    # True label for the entered fact
    true_label = data['class'].iloc[-1]

    # Check if the predictions are correct
    lr_correct = 1 if pred_LR[0] == true_label else 0
    dt_correct = 1 if pred_DT[0] == true_label else 0
    # Output the predictions and accuracy
    print("LR Prediction: {} \nDT Prediction: {}".format(output_lable(pred_LR[0]),
output_lable(pred_DT[0])))
```

```
    print("Correct LR Prediction: {} \nCorrect DT Prediction: {}".format(lr_correct, dt_correct))
    print("Accuracy for the entered fact:")
    print("LR Accuracy: {:.2%}".format(lr_correct))
    print("DT Accuracy: {:.2%}".format(dt_correct))
    # Calculate model accuracies
    lr_accuracy = accuracy_score([true_label], pred_LR)
    dt_accuracy = accuracy_score([true_label], pred_DT)
    print("Model accuracies:")
    print("LR Model Accuracy: {:.2%}".format(lr_accuracy))
    print("DT Model Accuracy: {:.2%}".format(dt_accuracy))
    # Plotting the pie chart
    flabels = ['Logistic Regression', 'Decision Tree']
    faccuracies = [lr_correct, dt_correct]
    plt.pie(faccuracies, labels=flabels, autopct='%1.1f%%', startangle=90, colors=['skyblue',
'lightgreen'])
    plt.title('Prediction Accuracies for the Entered Fact')
    display(plt.gcf())
    mlabels = ['Logistic Regression', 'Decision Tree']
    maccuracies = [lr_accuracy, dt_accuracy]

    plt.pie(maccuracies, labels=mlabels, autopct='%1.1f%%', startangle=90, colors=['lavender',
'pink'])
    plt.title('Prediction Accuracies for the Model')

# Get user input for manual testing
Fact = str(input("Enter a news text for manual testing: "))
```

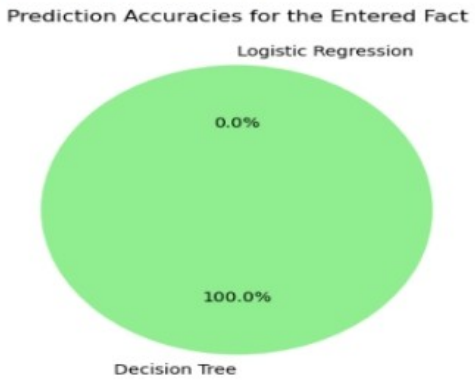
manual_testing(Fact)

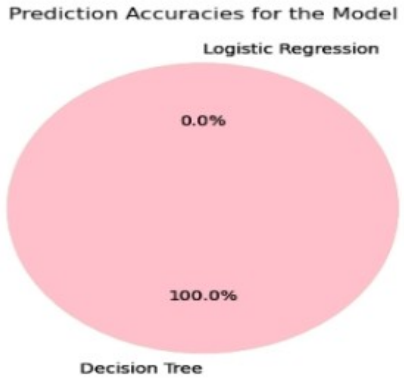
```
Enter a news text for manual testing: display(plt.gcf())
LR Prediction: True Fact
DT Prediction: True Fact
Correct LR Prediction: 1
Correct DT Prediction: 1
Accuracy for the entered fact:
LR Accuracy: 100.00%
DT Accuracy: 100.00%
Model accuracies:
LR Model Accuracy: 100.00%
DT Model Accuracy: 100.00%
```



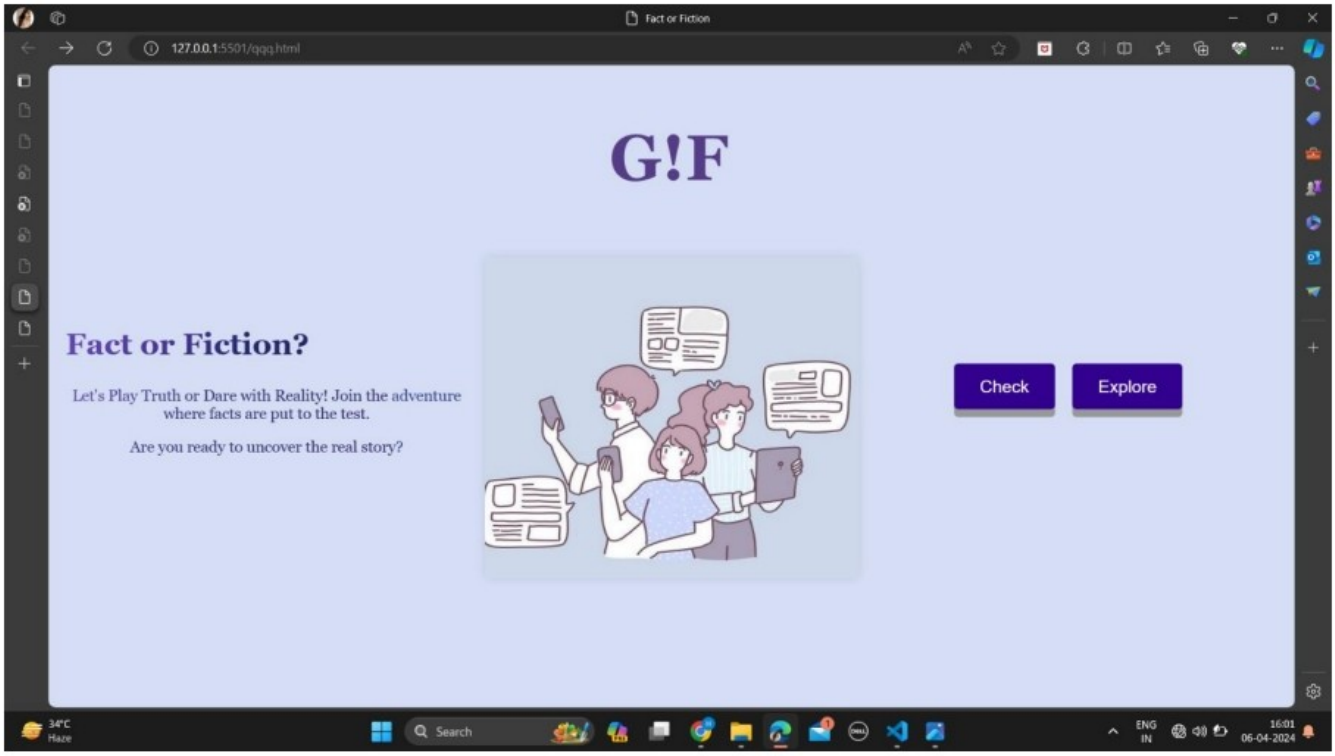


```
Enter a news text for manual testing: The Great Wall of China is Visible from the Moon
LR Prediction: Fake Fact
DT Prediction: True Fact
Correct LR Prediction: 0
Correct DT Prediction: 1
Accuracy for the entered fact:
LR Accuracy: 0.00%
DT Accuracy: 100.00%
Model accuracies:
LR Model Accuracy: 0.00%
DT Model Accuracy: 100.00%
```





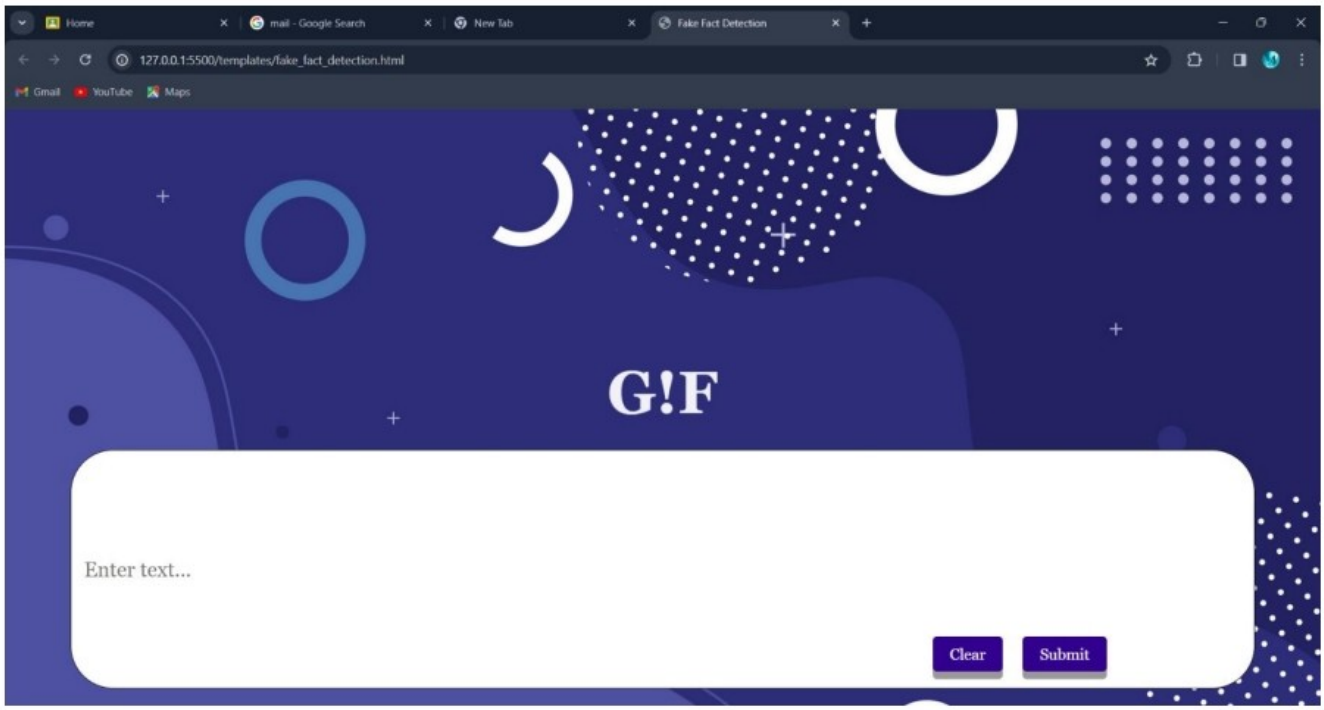
6. User Interface



The First page interacts with the user, telling it about the app and its use.

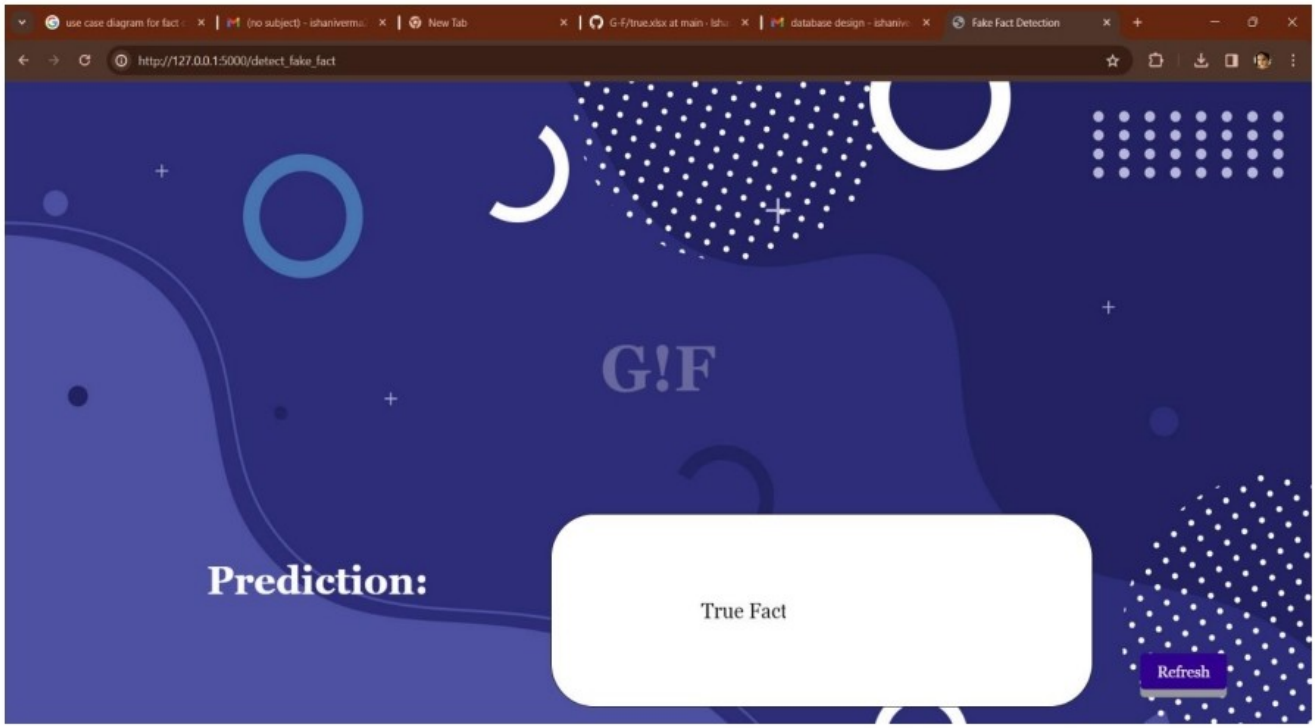
The Check button follows on the second page which is input page.

s



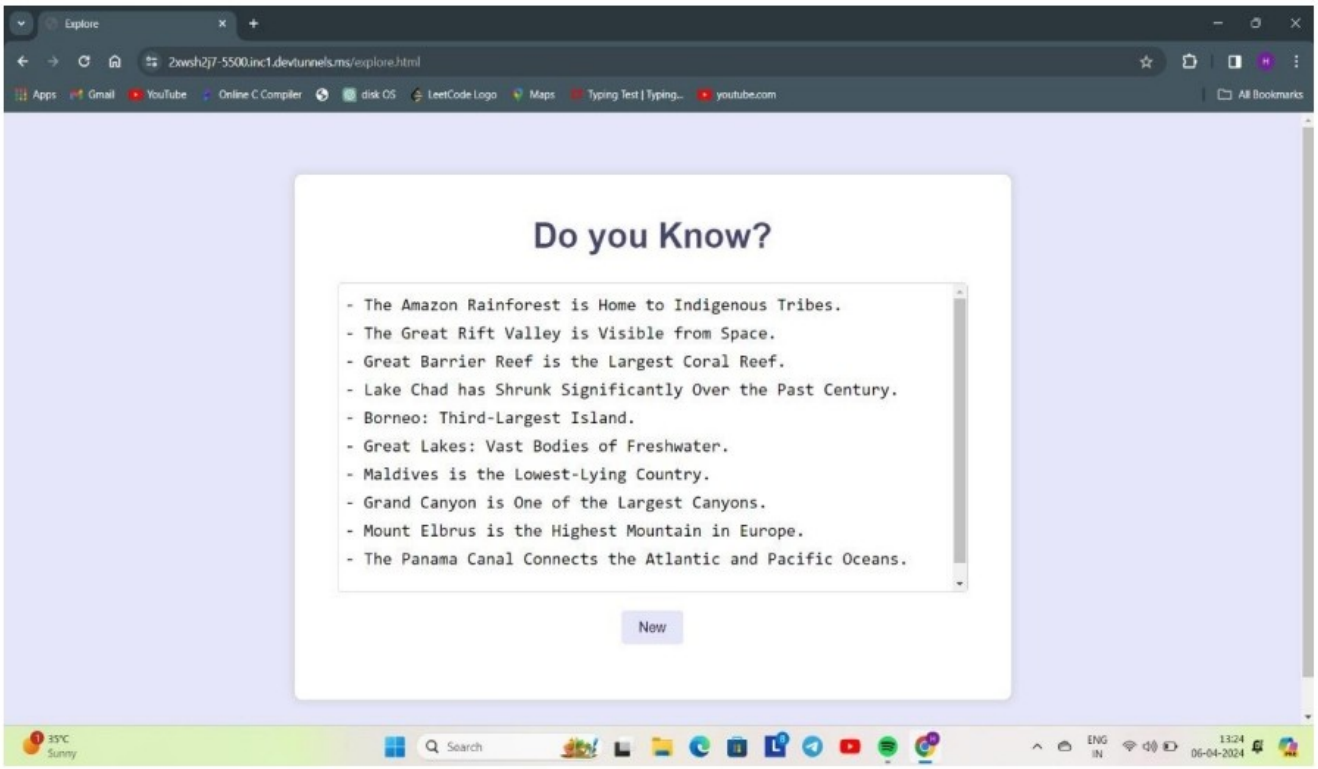
The text box takes the input.

The clear button is used to clear any text and the submit button is used to submit the input fact. It redirects it to the second page.



The Prediction box shows the output using LR and DT prediction. The refresh button redirects back to the input page.

The Explore button on the first page redirects to the page where the user an read and know facts.



The new button refreshes and shows new facts to the user.

7. Appendices

7.1 Test Environments Overview

7.1.1 Development Environment

Description: The development environment is where individual developers work on code and conduct unit tests.

Characteristics: Volatile environment with frequent code changes.

7.1.2 Integration or Regression Test Environment

Description: This environment allows multiple modules to be connected and tested as part of a use-case scenario.

Characteristics: Controlled environment with a focus on validating interactions between integrated components.

7.1.3 Stress Test Environment

Description: Stress tests are conducted in an environment identical to the target hosting environment, simulating high-stress conditions.

Characteristics: Identical to the target environment, may include production site usage during the first development cycle.

7.1.4 Acceptance Test and Staging Environment

Description: Acceptance tests and staging are performed in the same environment as stress tests.

Characteristics: Efficiency is gained by using the same environment for both acceptance tests and stress tests.

7.2 Testing Phases Summary

7.2.1 Unit Tests

Scope: Individual components or functions are tested.

Environment: Developer machines or a common server.

Implementation: Developers perform tests on their machines or a shared server.

7.2.2 System Tests

Scope: Multiple modules are connected and executed as in a typical use-case scenario.

Environment: Can be the same as unit testing or a separate machine.

Implementation: Testing focuses on validating system behaviour under normal conditions.

7.2.3 Integration or Regression Tests

Scope: Integration of multiple components or systems, or regression testing after changes.

Environment: Separate environment, like the target environment.

Implementation: Managed environment with controlled software changes.

7.2.4 Stress Tests

Scope: Simulates high-stress conditions to evaluate system performance.

Environment: Identical to the target hosting environment.

Implementation: Testing includes conditions that may lead to database deadlocks.

7.2.5 Acceptance Tests and Staging

Scope: Validates that the system meets specified requirements and prepares for deployment.

Environment: Same as stress tests, with potential use of the production environment during the initial development cycle.

Implementation: Simultaneous testing with stress tests, enhancing efficiency and resource optimization.

7.3 Additional Information

7.3.1 Test Bed Replication

Purpose: Replicate the deployment environment as part of the test bed for at least the second development cycle.

Considerations: Plan for scalability and flexibility in the test bed environment.

7.3.2 Resource Optimization

Efficiency: Sharing environments between tests optimizes resources and ensures consistency.
Planning: Adequate planning for hardware needs and potential expansion is crucial.

8. References

IEEE Std 830-1998: "IEEE Recommended Practice for Software Requirements Specifications," Institute of Electrical and Electronics Engineers, Inc., 1998.

ISO/IEC/IEEE 29148:2018: "Systems and software engineering - Life cycle processes - Requirements engineering," International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), and Institute of Electrical and Electronics Engineers (IEEE), 2018.

Additional References:

OWASP Application Security Verification Standard (ASVS): "OWASP Application Security Verification Standard," Open Web Application Security Project (OWASP), Version 4.0, ASVS Documentation.

SQL Standard: "ISO/IEC 9075:2016 - Information technology - Database languages - SQL," International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), 2016.

RESTful API Design Guide: "RESTful API Design Guide," by Microsoft, API Design Guide