**SOFTWARE ENGINEERING & AGILE DEVELOPMENT PROJECT**

On

**FAKE NEWS DETECTION**

Submitted in Partial Fulfilment of Award of

**BACHELOR OF TECHNOLOGY**

**In**

**Computer Science and Engineering**

By

K SAI LOKESH, 2022BCSE07AED273

N JAYAVARDHAN, 2022BCSE07AED280

G SREE HARSHA, 2022BCSE07AED324

T VISHNUVARDHAN, 2022BCSE07AED353

Y GANGA SAI REDDY, 2022BCSE07AED374

**Under the Supervision of**

DR. Vijayalakshmi, Associate professor, CSE



**ALLIANCE COLLEGE OF ENGINEERING AND DESIGN**

**ALLIANCE UNIVERSITY**

**BENGALURU**

**APRIL -2025**



**Computer Science and Engineering**

**ALLIANCE COLLEGE OF ENGINEERING AND DESIGN**

**CERTIFICATE**

This is to certify the project work entitiled “Fake News Detection” submitted by K.sai Lokesh[2022BCSE07AED273],N.jayavardhan[2022BCSE07AED280], G.Sreeharsha[2022BCSE07AED324],T.Vishnuvardhan[2022BCSE07AED353],and Y. ganga sai reddy [2022BCSE07AED374] in partial fulfilment for the award of the degree of Bachelor of Technology (Computer Science and Engineering) of Alliance University, is a Bonafide work accomplished under our supervision and guidance during the academic year 2024-2025. This thesis report embodies the results of original work and studies conducted by students and the contents do not form the basis for the award of any other degree to the candidate or anybody else.

**Dr. Vijayalakshmi N Dr. K. Ramalakshmi**

Associate Professor Professor & HOD

(Supervisor) (Head of Department)

**Examiner Signature: Date of Examination:**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ALLIANCE SCHOOL OF ADVANCED COMPUTING**

**Declaration**

This is to declare that the report titled **“Fake News Detection”** has been made for the partial fulfilment of the Course Bachelor of Technology in Computer Science and Engineering, under the Supervision of **Dr. Vijayalakshmi.** We confirm that this report truly represents our work undertaken as a part of our project work. This work is not a replication of work done previously by any other person. We also confirm that the report's contents and views have been discussed and deliberated with the faculty guide.

|  |  |  |
| --- | --- | --- |
| **NAME OF THE STUDENT** | **REG NO** | **SIGNATURE** |
| K SAI LOKESH | 2022BCSE07AED273 |  |
| N JAYAVARDHAN | 2022BCSE07AED280 |  |
| G SREEHARSHA | 2022BCSE07AED324 |  |
| T VISHNUVARDHAN | 2022BCSE07AED353 |  |
| Y GANGA SAI REDDY | 2022BCSE07AED374 |  |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **CHAPTER TITLE** | **PAGE NO** |
| 1 | Problem statement | **5-9** |
| 2 | Design Phase | **10-18** |
| 3 | Testing Phase | **19-20** |
| 4 | Agile Project Development | **21-49** |

**CHAPTER – 1**

**FAKE NEWS DETECTION**

**Introduction:**

In today's digital era, the rapid spread of misinformation and fake news has become a significant challenge. With the rise of social media and online platforms, false or misleading information can reach millions of people within minutes, influencing public opinion, shaping narratives, and even affecting decision-making in areas such as politics, health, and finance.

Fake news detection is a critical field that involves identifying and combating false information using various techniques, including natural language processing (NLP), machine learning, and fact-checking methodologies. Researchers and developers are leveraging AI-powered algorithms to analyse text patterns, verify sources, and differentiate between credible and deceptive content.

**Requirements:**

**Software Requirements:**

Operating System: Windows 7 / Windows 10

Front-End Technologies:

* J2EE (Java Enterprise Edition) or any equivalent framework
* HTML, CSS, JavaScript (for UI enhancements)

Back-End Technologies:

* MySQL Server, MongoDB, DBMS

CASE Tool (Modelling & Design):

* AGROUML or any equivalent (Rational Rose, Lucid chart, Microsoft Visio)

Programming Languages:

**.** Python, Node.js, JavaScript (MERN stack)

**Hardware Requirements:**

Processor:

* Intel Core i3 or higher

RAM:

* Minimum 4 GB (8 GB recommended for better performance)

Hard Disk Drive:

* Minimum 500 GB

Monitor & Display:

* Standard LED/LCD Monitor (1024x768 resolution or higher)

**Procedure/Step-by-step Instructions:**

**1.Problem Statement:**

Allow employees to gain jobs and receive applications online. Provide an automated store of ATS for each applicant based on resume analysis. Offer real-time resume analysis suggestions to applicants. He /She also tracks whether he/she shortlisted or not and receives feedback.

**2.Preparation of Software requirements Specification Document:**

**1. User Requirements (What users expect from the system)**

1. **Comprehensive Data Collection**

* Source Variety: Gather data from a wide range of trusted news outlets, social media platforms, and independent websites to cover different angles and domains.
* Balanced Dataset: Ensure the dataset contains a mixture of fake and legitimate news. It should include labeled examples that have been verified through credible sources or fact-checking organizations.

2. **Textual Analysis with Natural Language Processing (NLP)**

* Text Classification: Implement algorithms that classify news articles as real or fake based on linguistic features. This includes sentiment, word choice, and phrase structure.
* Entity Recognition: Use Named Entity Recognition (NER) to identify important entities (people, organizations, events) and check for consistency across the article and external sources.
* Contextual Analysis: Apply deep learning models, such as transformers (e.g., BERT), that can understand the context of statements and detect contradictions or misleading information.

**3. Fact-Checking Mechanisms**

* Cross-Verification: Implement automated tools that check the facts and figures mentioned in the news article against credible, authoritative databases and sources (e.g., fact-checking websites like PolitiFact or Snopes).
* Verification APIs: Leverage APIs that provide access to trusted fact-checking platforms or databases for real-time validation of news claims.

**4. Machine Learning Models for Classification**

* Supervised Learning: Utilize labeled data to train machine learning models (e.g., SVMs, Random Forests, or Neural Networks) to distinguish between fake and real news.

**2. System Requirements (Technical specifications needed to run the system)**

**Software Requirements:**

**Operating System:** Windows 7 / Windows 10

**Front-End Technologies:**

* J2EE (Java Enterprise Edition) or any equivalent framework
* HTML, CSS, JavaScript (for UI enhancements)

**Back-End Technologies:**

* MySQL Server, MongoDB, DBMS

**CASE Tool (Modelling & Design):**

* AGROUML or any equivalent (Rational Rose, Lucid chart, Microsoft Visio)

**Programming Languages:**

**.** Python, Node.js, JavaScript (MERN stack)

**Hardware Requirements:**

* **Processor:** Intel Core i3 or higher
* **RAM:** 4 GB (8 GB recommended for better performance)
* **Hard Disk:** 500 GB minimum
* **Internet Connection:** Required for cloud-based features

**3.Functional requirements:**

* 1. **User authentication and management**
* Users should be able to register and log in using email/password or social login.
* Admins should have access to manage users and their activities.

**News input & processing**

**.** Users should be able to submit news articles, headlines, or URLs for verification.

* The system should preprocess the input using text cleaning, tokenization, and stemming.

**Fake news classification**

* The system should analyse the input text using Machine Learning (ML) or Deep Learning models.
* **It should classify news into categories like:**

1. Real
2. Fake
3. Misleading
4. Satire or Opinion (Optional)

* Confidence scores should be displayed for classification results. Bulk resume processing and screening.

**Fact checking & external verification**

* integrate with fact-checking APIs (e.g., Google Fact Check API, PolitiFact).
* Compare the submitted news with trusted news sources.

**Database Management**

* Store submitted news articles and classification results in a database.
* Maintain a history of user queries for future reference.
* Allow admins to update and delete records.

**User feedback**

* Users should be able to report incorrect classifications.
* Implement a feedback system to improve model accuracy.

**4.Non-Functional Requirements:**

**4.1 Performance & Scalability**

* The system should classify news articles within 2-5 seconds.
* It should handle at least 100 concurrent users without performance degradation.
* The system should be scalable, allowing integration with cloud computing (AWS, Google Cloud, Azure).

**4.2 Accuracy & Reliability**

* The fake news detection model should have an accuracy of at least 85%.
* The system should cross-verify information using external sources before classifying news.
* The detection model should be updated periodically to improve accuracy.

**4.3 Security & Privacy**

* Implement SSL encryption for secure communication.
* User data privacy must comply with GDPR or other applicable laws.
* Prevent SQL injection, XSS, and CSRF attacks.

**4.4 Maintainability & Upgradability**

* The system should have modular architecture for easy updates and improvements.
* Machine learning models should be retrainable with new datasets.

**3. Study and usage of any Design phase CASE tool**

CASE Tool: AGROUML or Any Equivalent

**Step-by-Step Installation:**

**Step 1: Download Java 8**

1. Open your browser and go to Oracle's Java 8 download page:  
   <https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html>
2. Download the Windows x64 Installer (jdk-8uXXX-windows-x64.exe).
3. Run the installer and follow the on-screen instructions.
4. After installation, verify Java by opening Command Prompt (cmd) and typing:

java -version

It should display Java version 1.8.0\_xxx.

**Step 2: Download ARGOUML**

1. Go to the official AGROUML website:  
   http://argouml.tigris.org/ (or search for a mirror).
2. Download the latest ARGOUML JAR file (argouml-0.34.jar).

**Step 3: Run ARGOUML**

1. Open Command Prompt (cmd).
2. Navigate to the folder where you downloaded argouml-0.34.jar.
3. Run the following command:

java -jar argouml-0.34.jar

1. ARGOUML should now launch.

**Step 4: Create a Desktop Shortcut (Optional)**

1. Right-click argouml-0.34.jar → Open With → Choose Java Runtime Environment.
2. Check "Always use this app" and click OK.
3. Now, double-clicking the JAR file will launch ARGOUML.

**CHAPTER - 2**

**Diagram types in ARGOUML:**

1.Use Case Diagram

2.Class Diagram

3.Sequence Diagram

4.State State Diagram

5.Activity Diagram

6.Collaboration Diagram

7.Component Diagram

8.Deployment Diagram

9.Composite Structure Diagram

**Features of ARGOUML:**

* ARGOUML is an open-source UML diagramming tool.
* It supports various UML 1.4 diagram types.
* It includes design critics for design improvement suggestions.
* It can generate code from UML diagrams.
* It supports reverse engineering of code to UML diagrams.
* It uses XMI for UML model exchange.
* It is platform-independent due to its Java implementation.
* It provides a graphical user interface for diagram creation.
* It offers different explorer perspectives for model navigation.

**Performing the Design by using any Design phase CASE tools:**

CASE Tool: AGROUML

**Use Case Diagram:**

A Use Case Diagram is used in software engineering to visually represent the interactions between users (actors) and a system. It helps in understanding system functionality from a user's perspective.

**Actors:**

* user
* Admin

**Use Cases:**

* Submit the news article
* View result
* Receive notification
* Monitor system performance
* Generate report
* Store results

**Visual representation:**

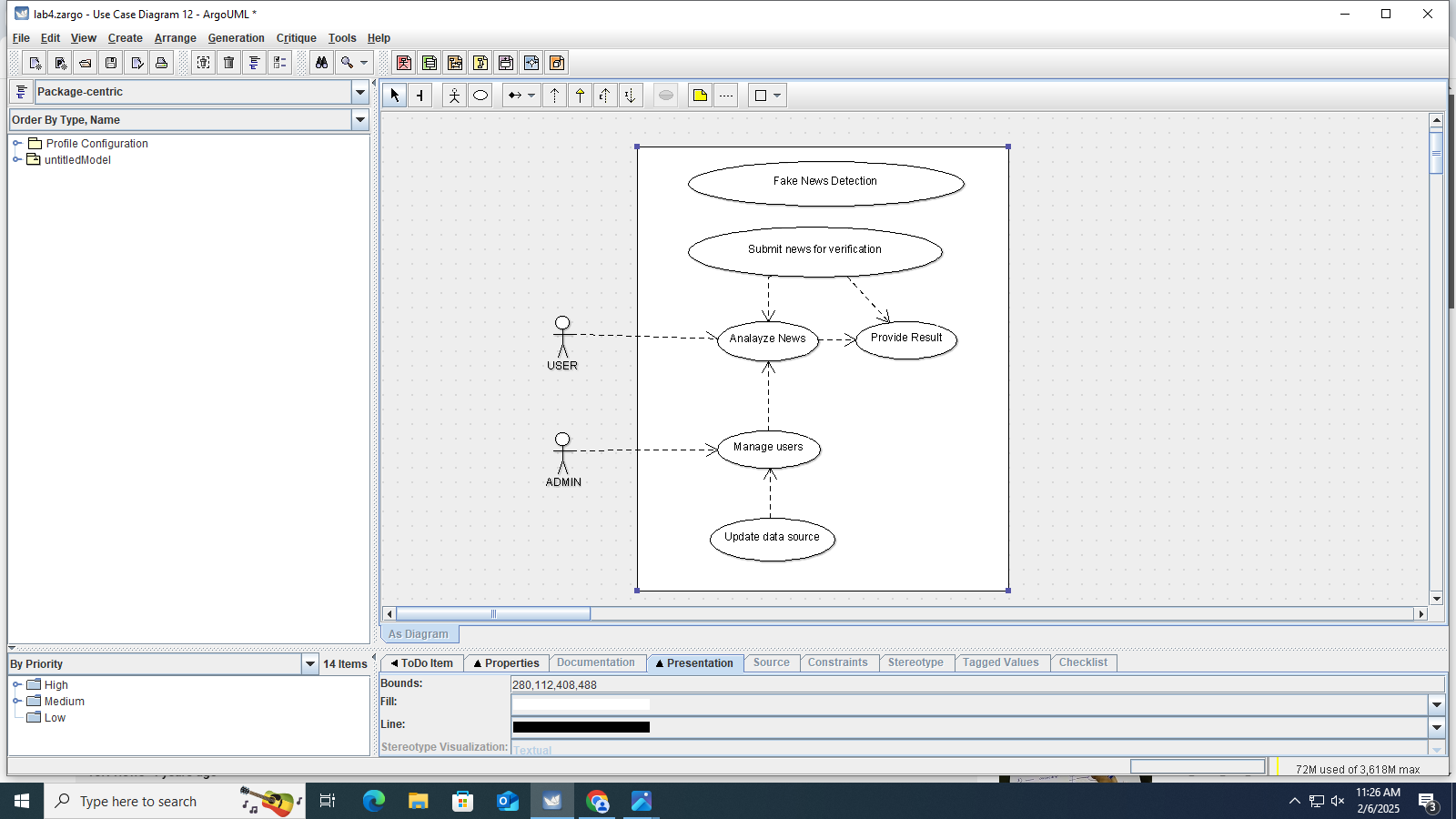


Figure no. 1: use case diagram

**1. Actors and Their Use**

**Admin (Fake News Creator & Distributor)**

**Role:**

Admins are the masterminds behind fake news. They generate and control the spread of false information, often with a specific agenda.

**What They Do:**

1. Create Fake Stories – Write misleading articles, posts, or videos.
2. Spread Through Networks – Use bots, fake accounts, and influencers to make the news go viral.
3. Target Emotions – Design content to trigger anger, fear, or excitement to make people engage.
4. Profit from Clicks – Earn money through ad revenue by attracting traffic to misleading websites.
5. Shape Public Opinion – Influence elections, social movements, or financial markets.

**User (Consumer & Sharer of Fake News)**

**Role:**

Users are the audience that interacts with fake news, either believing it, questioning it, or spreading it further.

**What They Do:**

1. Read & Believe – Some users accept fake news as truth without questioning.
2. Share & Spread – They repost, retweet, or forward misinformation, sometimes unintentionally.
3. Engage in Discussions – Some debate the news, while others defend it passionately.
4. Reinforce Biases – People tend to believe news that aligns with their views, making fake news more effective.

**Use Cases in Fake news detection**

1. **Submit News for Verification**

Actor: User  
Description: The user submits an article, image, or video for fact-checking.  
Flow:

1. User uploads a news link, text, or media file.
2. The system extracts key details such as the source, keywords, and context.
3. The request is sent to the AI model for analysis.

2. **Analyze News Content**

Actor: System  
Description: The system processes the submitted news to determine authenticity.  
Flow:

1. The AI model checks the credibility of the source.
2. NLP algorithms analyze the text for misleading language.
3. Cross-referencing with verified databases helps detect false claims.
4. Image/video forensics check for deepfakes or manipulated content.

3. **View Verification Results**

Actor: User  
Description: The user receives an analysis report on the submitted news.  
Flow:

1. The system provides a truthfulness score (e.g., True, False, Misleading).
2. The result includes fact-checked sources for reference.
3. If the news is false, a warning message explains why.

**Class Diagram:**

A Class Diagram is a type of UML (Unified Modelling Language) diagram that visually represents the structure of a system by showing its classes, attributes, methods, and relationships between them. It is widely used in object-oriented design to define the blueprint of a software system.

**1. User Class**

* Represents individuals who submit news for verification.
* **Attributes:**
  + userID – Unique identifier for the user.
  + name – User's full name.
  + email – Contact email of the user.
* **Methods:**
  + submit News (content: string) – Allows the user to submit news for fact-checking.

**2. News Class**

* Stores details of news articles submitted for verification.
* **Attributes:**
  + newsID – Unique identifier for each news item.
  + content – The textual or multimedia content of the news.
  + source – The original source or publisher of the news.
  + submitted by – The user who submitted the news.
  + status – The verification status (Pending, Verified, Fake, Misleading).
* **Methods:**
  + getNewsDetails () – Fetches detailed information about the news.
  + submitForAnalysis () – Sends the news content for verification.

**3. FakeNewsDetectionSystem Class**

* The core system that uses AI and algorithms to detect fake news.
* **Attributes:**
  + systemID – Unique identifier for the detection system.
  + algorithmVersion – The version of the AI model used for analysis.
* **Methods:**
  + analyzeNews (news: News) – Processes the submitted news and evaluates authenticity.
  + checkSourceReliability (news: News) – Verifies whether the news source is credible.
  + detectManipulation (news: News) – Checks for deepfake images, misleading text, or doctored videos.
  + assignCredibilityScore (news: News) – Assigns a reliability percentage to the news item.

**4. Report Class**

* Stores and presents the results of fake news analysis.
* **Attributes:**
  + reportID – Unique identifier for each report.
  + newsID – Associated news item being analyzed.
  + credibilityScore – A percentage indicating the reliability of the news (0-100%).
* **Methods:**
  + generateReport (news: News) – Generates a report based on the AI’s analysis.
  + getReportDetails () – Retrieves the summary and credibility assessment.

**5. Admin Class**

* Moderators who review flagged content and take corrective actions.
* **Attributes:**
  + adminID – Unique identifier for each admin.
  + name – Admin’s full name.
* **Methods:**
  + reviewFlaggedNews (newsID: int) – Manually checks news flagged by the AI system.
  + takeAction (newsID: int, action: string) – Takes action such as warning, removal, or restricting content.

**6. Database Class**

* Stores all news, reports, and user-related data.
* **Attributes:**
  + databaseID – Unique identifier for the database.
  + newsRecords – A collection of all submitted news items.
  + userRecords – A collection of registered users.
  + reportRecords – A collection of fact-checking reports.
* **Methods:**
  + storeNews (news: News) – Saves a new news item in the database.
  + fetchNews (newsID: int) – Retrieves a specific news item.
  + storeReport (report: Report) – Saves a generated report.

**Visual Representation:**

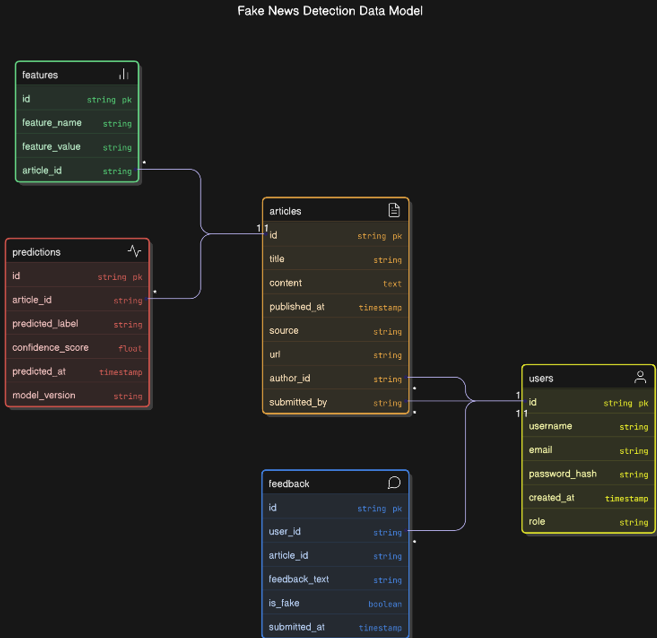
****

Figure no. 2: class diagram

**Interaction Diagram:**

An Interaction Diagram is a type of UML (Unified Modelling Language) diagram used to visualize how objects in a system interact with each other over time. It focuses on the flow of messages between objects to complete a specific task or process. Interaction diagrams help developers understand system behaviour and design efficient communication between components.

**1. Key Actors in the Fake News Detection System**

1. User – Submits news for verification and views the results.
2. Admin – Reviews flagged news and takes necessary actions (e.g., warn, remove, restrict).
3. Fact-Checker – Manually verifies flagged news and updates its status.
4. Fake News Detection System (AI Model) – Analyzes and classifies news authenticity.

**2. Key Objects in the System**

1. News – Represents the submitted news content.
2. Fake News Detection System – AI-powered tool that evaluates news credibility.
3. Report – Stores credibility scores and fact-checking results.
4. Database – Stores news, reports, and verification details.

3. **Sequence of Interactions (Step-by-Step Process)**

A. User Submits News for Verification

1. User → News: The user submits a news article or media for verification.
2. News → Fake News Detection System: The system receives the news and processes it.
3. Fake News Detection System → Database: The system checks past fact-checking records for similar content.
4. Fake News Detection System → Report: The system assigns a credibility score and generates a report.
5. Report → Database: The verification report is stored for future reference.
6. User → Report: The user views the credibility score and fact-checking details.

**Visual Representation:**

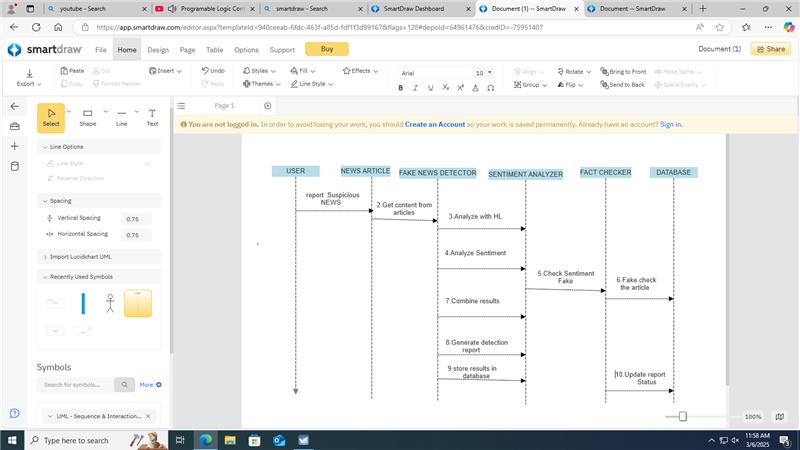


Figure no. 3:interaction Diagram

**State Chart Diagram:**

A State Chart Diagram (also called a State Machine Diagram) is a UML diagram that represents the different states of an object and how it transitions from one state to another based-on events.

It is useful for modelling dynamic behaviour in systems where an object changes states in response to inputs, actions, or conditions.

**1. States Involved in Report Generation**

1. Report Not Created – No report exists yet, waiting for news submission.
2. News Submitted – The user submits a news article for fact-checking.
3. Under Analysis – The Fake News Detection System processes the news.
4. Initial Report Generated – The system assigns a preliminary credibility score.
5. Flagged for Review (If needed) – If the AI detects high-risk content, the report is flagged for manual review.
6. Fact-Checker Verification (Optional) – A human expert reviews the flagged report.
7. Final Report Generated – The final status is determined (True, False, Misleading, Inconclusive).
8. Report Stored in Database – The final verified report is saved.
9. Report Available to Users – Users can view the report and verification details.

**2. Representation of the State Transitions**

1. Start (Initial State) → Report Not Created
2. User submits news → News Submitted
3. AI processes the news → Under Analysis
4. AI assigns credibility score → Initial Report Generated
5. If news is suspicious → Flagged for Review
6. Fact-checker manually verifies (if required) → Fact-Checker Verification
7. Final decision made → Final Report Generated

**Visual Representation:**

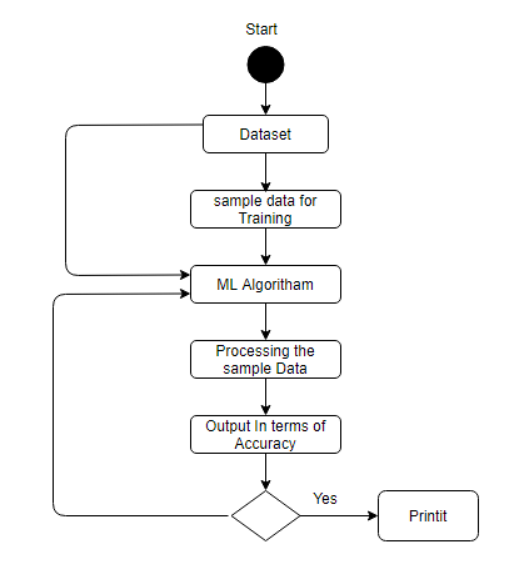


Figure no. 4: state chart diagram

CHAPTER - 3

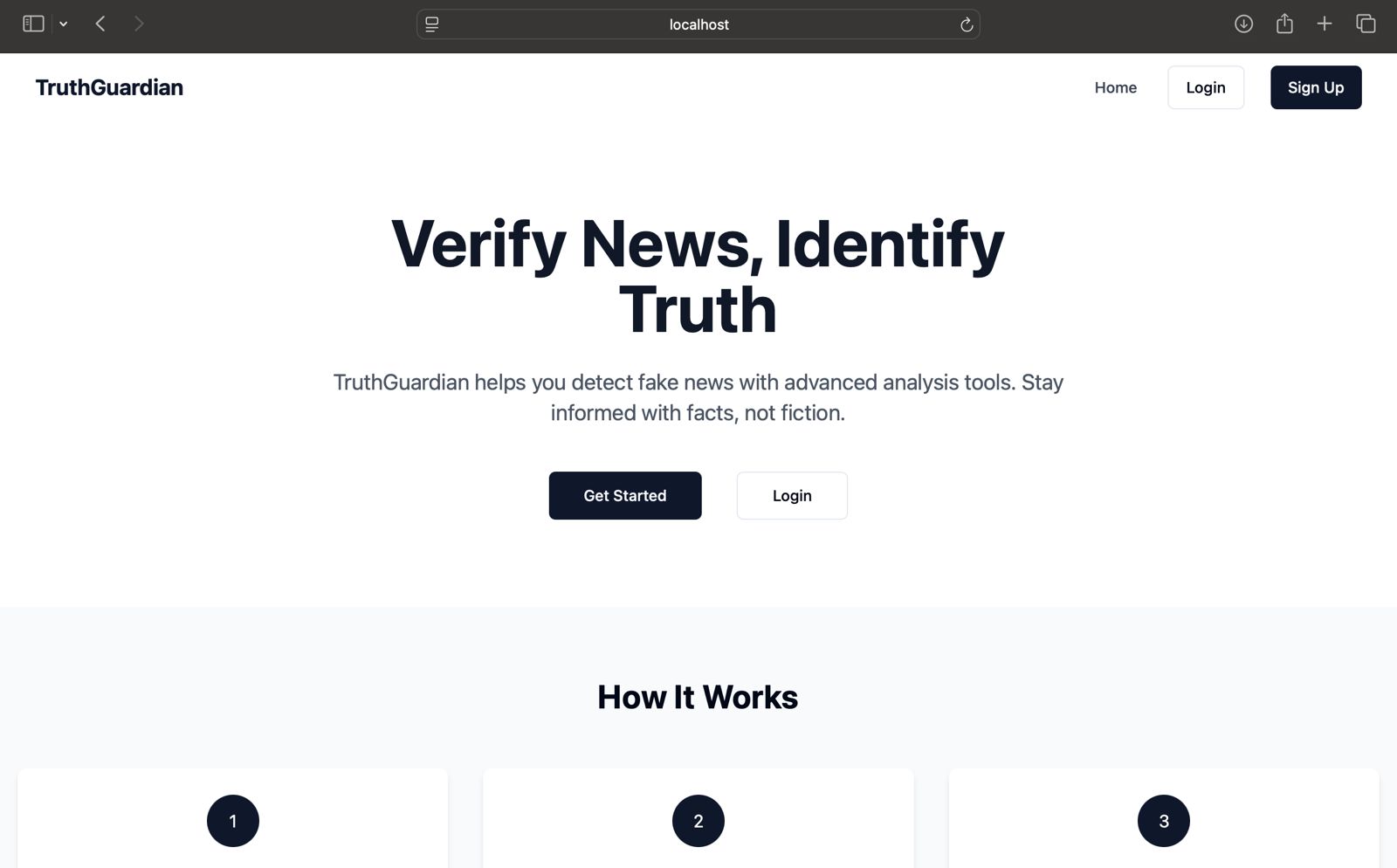
**Sample output:**

Figure no. 5: Login Page

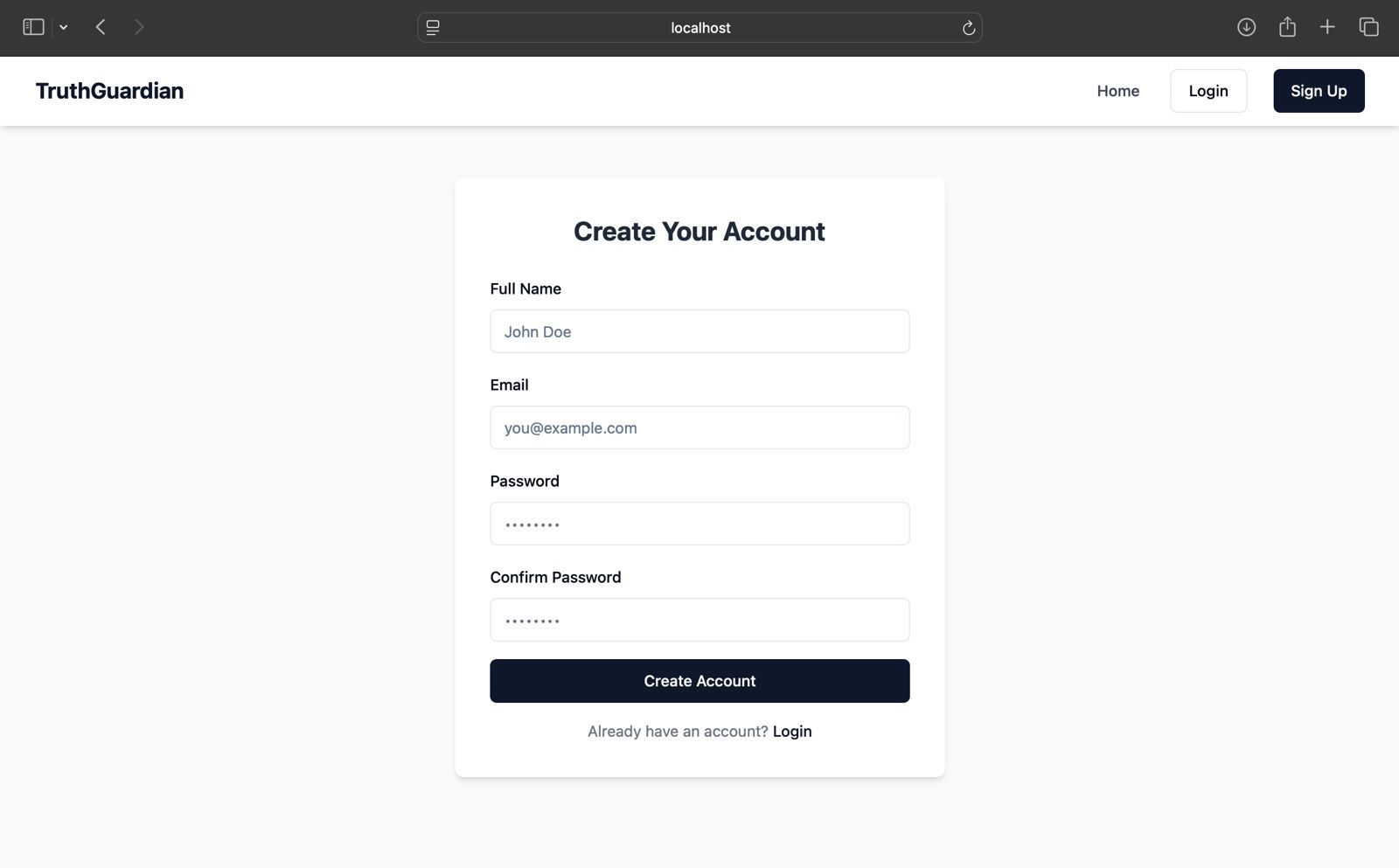


Figure no.6 Account Creation

A screenshot of a computer

AI-generated content may be incorrect.

Figure no. 7: Sample output

INFERENCE

The Fake News Detection System developed in this project effectively addresses the growing issue of misinformation across digital platforms. By using machine learning and natural language processing, the system is capable of analyzing news content and classifying it as real or fake based on learned patterns. The inclusion of features such as result explanation and user feedback enhances the transparency and reliability of the predictions, making it user-friendly and trustworthy.

Through Agile development and iterative sprint cycles, each core component—from data preprocessing and model training to UI integration and testing—was successfully implemented and refined. The final outcome is a functional, responsive, and intelligent application that can assist users in verifying the authenticity of online news. This project not only demonstrates the practical application of AI in real-world problems but also contributes toward creating a more informed and responsible digital society

CHAPTER - 4

**AGILE DEVELOPMENT**

**Practical No. 1**

Understand the background and driving forces for taking an Agile Approach to  
Software Development

**Objective**  
1. Difference between agile software development model and waterfall model.  
2. Why Agile is better?  
3. Understanding the Agile Manifesto  
4. Discussing Important Characteristics that make agile approach best suited for  
Software Development.

**Theory**  
Agile software development is a group of software development methods in which  
requirements and solutions evolve through collaboration between self-organizing,  
cross-functional teams. It promotes adaptive planning, evolutionary development,  
early delivery, continuous improvement, and encourages rapid and flexible response to  
change.The Manifesto for Agile Software Development, also known as the Agile Manifesto, firstintroduced the term agile in the context of software development in 2001.Manifesto for Agile Software DevelopmentWe are uncovering better ways of developing software by doing it and helping others do it.Through this work we have come to value Individuals and interactions over processes and toolsWorking software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan That is, while there is value in the items on the right, we value the items on the left  
more.

**Agile principles**  
The Agile Manifesto is based on 12 principles:  
1. Customer satisfaction by rapid delivery of useful software  
2. Welcome changing requirements, even late in development  
3. Working software is delivered frequently (weeks rather than months)  
4. Close, daily cooperation between business people and developers  
5. Projects are built around motivated individuals, who should be trusted  
6. Face-to-face conversation is the best form of communication (co-location)  
7. Working software is the principal measure of progress  
8. Sustainable development, able to maintain a constant pace  
9. Continuous attention to technical excellence and good design  
10.Simplicity—the art of maximizing the amount of work not done—is essential

11. Self-organizing teams  
12. Regular adaptation to changing circumstance

**What’s wrong with Traditional Approaches?**  
In 1970, Dr. Winston Royce presented a paper entitled “Managing the Development of  
Large Software Systems,” which criticized sequential development. He asserted that  
software should not be developed like an automobile on an assembly line, in which  
each piece is added in sequential phases, each phase depending on the previous. Dr.Royce recommended against the phase based approach in which developers first gather all of a project’s requirements, then complete all of its architecture and design,then write all of the code, and so on. Royce specifically objected to the lack of communication between the specialized groups that complete each phase of work.It’s easy to see the problems with the waterfall method. It assumes that every requirement can be identified before any design or coding occurs. Could you tell a team of developers everything that needed to be in a software product before any of it was up and running? Or would it be easier to describe your vision to the team if you  
could react to functional software? Many software developers have learned the answer  
to that question the hard way: At the end of a project, a team might have built the  
software it was asked to build, but, in the time it took to create, business realities have  
changed so dramatically that the product is irrelevant. Your company has spent time  
and money to create software that no one wants. Couldn’t it have been possible to  
ensure the end product would still be relevant before it was actually finished?  
Today very few organizations openly admit to doing waterfall or traditional command  
and control. But those habits persist.

**Why Agile?**  
Agile development provides opportunities to assess the direction throughout the  
development lifecycle. This is achieved through regular cadences of work, known as  
Sprints or iterations, at the end of which teams must present a potentially shippable  
product increment. By focusing on the repetition of abbreviated work cycles as well as  
the functional product they yield, agile methodology is described as “iterative” and  
“incremental.” In waterfall, development teams only have one chance to get each  
aspect of a project right. In an agile paradigm, every aspect of development —  
requirements, design, etc. — is continually revisited. When a team stops and re-  
evaluates the direction of a project every two weeks, there’s time to steer it in another  
direction.  
This “inspect-and-adapt” approach to development reduces development costs and  
time to market. Because teams can develop software at the same time they’re  
gathering requirements, “analysis paralysis” is less likely to impede a team from  
making progress. And because a team’s work cycle is limited to two weeks,  
stakeholders have recurring opportunities to calibrate releases for success in the real  
world. Agile development helps companies build the right product. Instead of  
committing to market a piece of software that hasn’t been written yet, agile empowers  
teams to continuously replan their release to optimize its value throughout  
development, allowing them to be as competitive as possible in the marketplace. Agile  
development preserves a product’s critical market relevance and ensures a team’s  
work doesn’t wind up on a shelf, never released.

Difference between agile software development model and waterfall model  
It is worth mentioning here that the Waterfall model is the primitive model type and  
has been implemented in the development phase time after time. Hence in the due  
course if time developers found many drawbacks in this model which were later  
rectified to form various other development models.

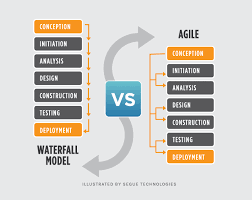


Figure no. 8: Waterfall model V/S Agile

**Advantages of the Agile Methodology**  
1. The Agile methodology allows for changes to be made after the initial planning. Re-writes to the program, as the client decides to make changes, are expected.  
2. Because the Agile methodology allows you to make changes, it’s easier to add features that will keep you up to date with the latest developments in your industry.  
3. At the end of each sprint, project priorities are evaluated. This allows clients to add their feedback so that they ultimately get the product they desire.  
4. The testing at the end of each sprint ensures that the bugs are caught and taken care of in the development cycle. They won’t be found at the end.  
5. Because the products are tested so thoroughly with Agile, the product could be launched at the end of any cycle. As a result, it’s more likely to reach its launch date.  
**Conclusion**  
Both the Agile and waterfall methodologies have their strengths and weaknesses. The  
key to deciding which is right for you comes down to the context of the project. Is it  
going to be changing rapidly? If so, choose Agile. Do you know exactly what you need?  
Good. Then maybe waterfall is the better option. Or better yet?

**Practical No. 2**   
Understand the business value of adopting Agile  
Approaches

**Objective**

1. Understanding the business values  
2. Adopting Agile Software Development: Issues and Challenges  
3. Overview of Scrum, Extreme Programming, Feature Driven development, Lean  
Software Development, Agile project management

**Theory**  
1. When it comes to creating custom applications, too many of us live in denial. We  
want to believe that it’s possible to predict accurately how long a group of  
developers will take to build software that meets our requirements.  
2. We also want to believe that we can know what those requirements are up  
front, before we’ve seen any part of the application, and that the requirements  
won’t change during development. Sadly, none of these things are true for most  
projects. We can’t predict how long development will take, largely because we  
can’t get the requirements right up front and we can’t stop them from changing.  
Because we deny these realities, many organizations still use software  
development processes that don’t work well. Fortunately, this is changing.  
3. Agile development processes get more popular every day, primarily because  
they’re rooted in reality: They’re designed to accommodate change.  
Doing software development in this way can be scary at first, especially for  
the business leaders who are footing the bill. This needn’t be the case, however.  
The truth is that agile processes are usually better both for development teams  
and for the business people who pay them. To understand why this is true,  
we need to start by understanding what an agile process really is. What Agile  
Development Means The challenge is always the same: We need to create  
software in the face of uncertainty. At the start of a development project,  
we don’t know whether we’ve defined the project’s requirements correctly.  
4. We also don’t know how those requirements will change while we’re building the  
software. A traditional development process does its best to pretend this  
uncertainty doesn’t exist. In the classic waterfall approach, for example,  
an organization creates detailed plans and precise schedules before writing any  
code. But real development projects rarely comply with these plans and  
schedules—they’re notoriously unruly. The core reason for this is that even  
though we use the term “software engineering”, writing code isn’t like other  
kinds of engineering. In traditional engineering projects—building a bridge,  
say, or constructing a factory—it’s usually possible to define stable  
requirements up front.  
5. Once you’ve done this, creating plans and schedules based on previous  
experience is straightforward. Software development just isn’t like this1 .  
Creating stable requirements up front is usually impossible, in part because  
people don’t know what they want until they see it. And since every  
development project involves some innovation—if it doesn’t, you should be  
buying rather than building the software—uncertainty is unavoidable.  
Traditional development processes work against these realities. Agile  
processes, however, are designed for this situation. Because requirements  
change, an agile process provides a way to add, remove, and modify  
requirements during the development process. Rather than resisting change, an

agile process embraces it. Just as important, the process recognizes that  
short-term plans are more stable than long-term plans. You might not know  
exactly what you want to be doing three months from now, but you probably do

know what you want to do in the next three weeks. To accomplish this, an agile  
development process relies on iteration. Rather than the traditional approach of  
defining all of the requirements, writing all of the code, then testing that code, an agile  
process does these things over and over in smaller iterations. Each iteration creates  
more of the finished product, with the requirements updated at the start of each one.  
Challenges in adopting agile Methodology

Challenge 1: Missing the Agile Master Role Agile master or Agile coach is an essential  
role during Agile adopting process in any organization. Agile coach is considered a  
consultant for the team in every step of a project using any Agile method, such as  
Scrum, that is responsible of providing guidance and helps to succeed in adopting  
Agile. Entity's” management recognized the need to hire a contractor as an agile  
master. However, the position was not filled due to financial constraints.

Challenge 2: The overzealous teams after attending a course on Agile methods, many  
of entity “S” teams wanted to adopt Agile methods as soon as possible hoping it will  
solve all their previous development challenges known for traditional methods. This  
overzealous team fast adoption of Agile resulted in a decrease in productivity because  
the development cycle took longer time due to many mistakes in implementation. This  
decrease in productivity led many team members to be less optimistic and started to  
lose interest in agile methods.

Challenge 3: The absent of a Pilot Project Another challenge is the absent of a pilot  
project in the transition from the previous traditional method to the scrum method.  
Conducting a pilot project was a recommended step in the adoption of agile  
development for the first time. As a part of the plan to adopt agile method, the pilot  
project is essential to evaluate how ”S” environment will be able to move from the  
previous heavy-weight method to a new light method. Many organizations went  
through the same experience of running a pilot project especially those companies that  
have large projects such as Amazon, Yahoo, Microsoft and Intel. After investing the  
needed time and resources they have reached to a successful adoption of Agile.

Challenge 4: Scrum Implementation International Journal of Managing Value and  
Supply Chains (IJMVSC) Vol. 2, No. 3, September 2011 7 Although the employees in  
”S” were very experienced but yet none of them had any previous experience with agile  
development methods or Scrum implementation in particular. This is in addition of  
the absence of the agile master. For the team members, scrum implementation was  
not easy as it appeared to be during the training session. The team members find  
themselves, suddenly, in a completely new set up. The experience of traditional  
methods is completely different than committing to daily meeting, working with time  
boxes, finishing tasks in small period iteration and documenting the stories (or  
backlogs) in a different way.

Challenge 5: Current Work Pressure Although ”S” software development team serves a  
very large organization of over 30 departments and developed numerous projects  
through the years, the development projects require continues maintenance and  
support. In addition, the team was working on a new project with firm deadlines. The  
work environment was very demanding and team worked under pressure to produce  
products according to the planned schedule. Scrum adoption process started while  
every member of the team was engaged in his/her everyday tasks. With such work  
pressure the daily Scrum meetings were considered waste of time and added extra  
pressure to the employees. They used to meet weekly and later twice a month and  
then only when required and usually after working hours. As teams started to skip

daily meetings it also affected the learning process of scrum between the team  
members. That eventually leads to the failure of learning and implementing agile  
method correctly.

Challenge 6: Upper Management Concerns The upper management of ”S” had many  
concerns about the effectiveness and success of the transition to a new method. They  
were not easily convinced to invest in a new method.

Challenge 7: Governmental bureaucratic System The traditional method currently in  
“S” was customized to comply with the governmental system of other department. The  
new Agile method being introduced, Scrum, is developed in such highly bureaucratic  
environment. The Agile team has to secure approvals and signatures before moving  
from one step to another. This was perceived by the team members as unnecessary  
and more time was taken into account to develop a new project. The scrum method  
requires much less correspondence, less time in communication between the customer  
and the team and requires significantly less paper work and approvals as the  
customer is supposed to be involved in every step.

Challenge 8: Documentation requirements after years and years of extensive  
documentation of every step in the traditional method, moving to a new method with  
minimum documentation requirements was one of the greatest challenges. Every  
project used to end up with dozens of document such as project charter, project plan,  
testing plan, SRS, STS, technical documents, user manual, etc. Each of these  
document contained large number of pages written by every member of the team and  
consumed hours of the valuable development time. The documentation requirements  
were driven basically from the previous challenge (the governmental system), upper  
management, ISO certificate requirements and the traditional development method  
that is currently used. Although agile development promises sufficient documentation  
of the projects, it didn’t seem very convincing to the upper management when they  
end up receiving few documents in comparison with the previous model of  
documentation. Many attempts were made to try to balance between the upper  
management requirement regarding documentation and between adopting Scrum  
method. Agile teams started to increase the number of documents required for  
documentation and started to customize Scrum as much as possible to conform to all  
the upper management requirements of documentation norms. This did not work very  
well and it created extra burden on the agile teams.

Agile Process Examples  
1. SCRUM  
2. FDD  
3. Lean software development  
4. XP

1. SCRUM  
Scrum is an iterative and incremental agile software development methodology for  
managing product development. It defines "a flexible, holistic product development  
strategy where a development team works as a unit to reach a common goal",  
challenges assumptions of the "traditional, sequential approach" to product  
development, and enables teams to self-organize by encouraging physical co-location  
or close online collaboration of all team members, as well as daily face-to-face  
communication among all team members and disciplines in the project.

1. FDD

Feature-driven development (FDD) is an iterative and incremental software  
development process. It is one of a number of lightweight or agile methods for  
developing software. FDD blends a number of industry-recognized best practices into a  
cohesive whole. These practices are all driven from a client-valued functionality  
(feature) perspective. Its main purpose is to deliver tangible, working software  
repeatedly in a timely manner.

3. Lean software development  
Lean software development (LSD) is a translation of lean manufacturing and lean  
IT principles and practices to the software development domain. Adapted from the  
Toyota Production System,] a pro-lean subculture is emerging from within the Agile  
community. Lean is most popular with startups that want to penetrate the market, or  
  
**Conclusion**  
Each agile methodology has a slightly different approach for implementing the core  
values from the Agile Manifesto, just as many computer languages manifest the core  
features of object-oriented programming in different ways. A recent survey shows that  
about 50 percent of agile practitioners say that their team is doing Scrum. Another 20  
percent say that they are doing Scrum with XP components. An additional 12 percent  
say that they are doing XP alone. Because more than 80 percent of agile  
implementations worldwide are Scrum or XP, MSF for Agile Software Development  
v5.0 focuses on the core processes and practices of Scrum and XP

**Practical No. 3**  
Understand the Agile development practices

**Objective**  
Understanding SCRUM

**Theory**  
Scrum is an iterative and incremental agile software development methodology for  
managing product development. It defines "a flexible, holistic product development  
strategy where a development team works as a unit to reach a common goal",  
challenges assumptions of the "traditional, sequential approach" to product  
development, and enables teams to self-organize by encouraging physical co-location  
or close online collaboration of all team members, as well as daily face-to-face  
communication among all team members and disciplines in the project.  
A key principle of scrum is its recognition that during a project the customers can  
change their minds about what they want and need (often called "requirements  
churn"), and that unpredicted challenges cannot be easily addressed in a traditional  
predictive or planned manner. As such, scrum adopts an empirical approach—  
accepting that the problem cannot be fully understood or defined, focusing instead on  
maximizing the team's ability to deliver quickly and respond to emerging requirements.

History  
Scrum was first defined as "a flexible, holistic product development strategy where a  
development team works as a unit to reach a common goal" as opposed to a  
"traditional, sequential approach" in 1986 by Hirotaka Takeuchi and Iki Jiro Nonaka in  
the New Product Development Game. Takeuchi and Nonaka later argued in The  
Knowledge Creating Company that it is a form of "organizational knowledge creation,  
[...] especially good at bringing about innovation continuously, incrementally and  
spirally".  
The authors described a new approach to commercial product development that would  
increase speed and flexibility, based on case studies from manufacturing firms in the  
automotive, photocopier and printer industries. They called this the holistic or rugby  
approach, as the whole process is performed by one cross-functional team across  
multiple overlapping phases, where the team "tries to go the distance as a unit,  
passing the ball back and forth". (In rugby football, a scrum refers to a tight -packed  
formation of players with their heads down who attempt to gain possession of the ball.)  
In the early 1990s, Ken Schwaber used what would become scrum at his company,  
Advanced Development Methods, and Jeff Sutherland, with John Scumniotales and  
Jeff McKenna, developed a similar approach at Easel Corporation, and were the first to  
refer to it using the single word scrum. In 1995, Sutherland and Schwaber jointly  
presented a paper describing the scrum methodology at the Business Object Design  
and Implementation Workshop held as part of Object-Oriented Programming, Systems,  
Languages & Applications '95 (OOPSLA '95) in Austin, Texas, its first public  
presentation. Schwaber and Sutherland collaborated during the following years to  
merge the above writings, their experiences, and industry best practices into what is  
now known as scrum.

n 2001, Schwaber worked with Mike Beedle to describe the method in the book Agile  
Software Development with Scrum. Its approach to planning and managing projects is  
to bring decision-making authority to the level of operation properties and  
certainties. Although the word is not an acronym, some companies implementing the  
process have been known to spell it with capital letters as SCRUM. This may be due to  
one of Ken Schwaber's early papers, which capitalized SCRUM in the title.  
Later, Schwaber with others founded the Scrum Alliance and created the Certified  
Scrum Master programs and its derivatives. Schwaber left the Scrum Alliance in the  
fall of 2009, and founded Scrum.org to further improve the quality and effectiveness of  
scrum.

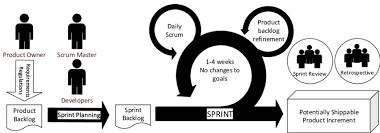


Figure no.9: Process of agile

How Does Scrum Fit With Agile?  
The Agile Manifesto doesn’t provide concrete steps. Organizations usually seek more  
specific methods within the Agile movement. These include Crystal Clear, Extreme  
Programming, Feature Driven Development, Dynamic Systems Development Method  
(DSDM), Scrum, and others. While I like all the Agile approaches, for my own team  
Scrum was the one that enabled our initial breakthroughs. Scrum’s simple definitions  
gave our team the autonomy we needed to do our best work while helping our boss  
(who became our Product Owner) get the business results he wanted. Scrum opened  
our door to other useful Agile practices such as test-driven development (TDD). Since  
then we’ve helped businesses around the world use Scrum to become more agile. A  
truly agile enterprise would not have a “business side” and a “technical side.” It would  
have teams working directly on delivering business value. We get the best results  
when we involve the whole business in this, so those are the types of engagements I’m  
personally the most interested in.

What’s The Philosophy behind Scrum?  
Scrum’s early advocates were inspired by empirical inspect and adapt feedback loops  
to cope with complexity and risk. Scrum emphasizes decision making from real-world  
results rather than speculation. Time is divided into short work cadences, known as  
sprints, typically one week or two weeks long. The product is kept in a potentially  
shippable (properly integrated and tested) state at all times. At the end of each sprint,  
stakeholders and team members meet to see a demonstrated potentially shippable  
product increment and plan its next steps.  
Scrum is a simple set of roles, responsibilities, and meetings that never change. By  
removing unnecessary unpredictability, we’re better able to cope with the necessary  
unpredictability of continuous discovery and learning.

**Scrum Roles**  
**Product Owner**: The Product Owner should be a person with vision,  
authority, and availability. The Product Owner is responsible for  
continuously communicating the vision and priorities to the development  
team. It’s sometimes hard for Product Owners to strike the right balance of  
involvement. Because Scrum values self-organization among teams, a  
Product Owner must fight the urge to micro-manage. At the same time,  
Product Owners must be available to answer questions from the team.

**Scrum Master**: The Scrum Master acts as a facilitator for the Product  
Owner and the team. The Scrum Master does not manage the team. The  
Scrum Master works to remove any impediments that are obstructing the  
team from achieving its sprint goals. This helps the team remain creative  
and productive while making sure its successes are visible to the Product  
Owner. The Scrum Master also works to advise the Product Owner about  
how to maximize ROI for the team.  
**Team:** According to Scrum’s founder, “the team is utterly self-managing.”  
The development team is responsible for self-organizing to complete work. A  
Scrum development team contains about seven fully dedicated members  
(officially 3-9), ideally in one team room protected from outside distractions.  
For software projects, a typical team includes a mix of software engineers,  
architects, programmers, analysts, QA experts, testers, and UI designers.  
Each sprint, the team is responsible for determining how it will accomplish  
the work to be completed. The team has autonomy and responsibility to  
meet the goals of the sprint.

**Scrum Meetings**  
There are number of regular meetings take place among Agile Team members. Let’s go  
through different type of meetings.

1. Sprint Planning Meeting  
During a sprint team works on selected features. Those features are planned and  
selected in sprint planning meeting which holds before every sprint. Product owner  
and scrum team participate this planning session which usually last for 4 hours.

Product owner come up with a set of features to discuss with intent to add into sprint  
backlog so team could work on it during the sprint. Usually those features have a  
specific theme attach to them, that theme is also called sprint goal.  
Following are a few example of the sprint goal.  
● Improve the performance of the current system.  
● Build the UI of certain component.  
● Create API of the product so other application could access some specific  
information.  
Based on input from the team, features are finalized for the sprint. Team input helps  
to decide if proposed features can be completed in the sprint or there are any  
impediments involve in completing those features. Features story points should be fall  
within sprint velocity. Once features are decided scrum team breaks those features  
into tasks. Team may assign hours or story point to each task. The output of the  
meeting is a sprint backlog, sprint theme/goal and tasks related to each feature.

Timings: Before every Sprint  
Duration: 4-8 hours  
Participants: Scrum Owner, Scrum master, Scrum Team  
Artifacts: Sprint Tag/Goal, sprint backlog, tasks

**2. Release Planning**  
Meeting Purpose of the meetingAn Agile project may consist of number of iterations (sprints). A release consists of a sub set of such iterations. The length of release could be between 2 to 6 months.Release planning meeting is held at the beginning of each release. The purpose of the release planning meeting is to go through the backlog and give an estimate the  
number and set the priority of features that can be completed in a release. At the start of the meeting Product owner usually come up with the backlog with selected features  
he wants to get completed during the release time frame. Scrum team gives its opinion  
regarding the features and based on the discussion certain features are added or  
removed from the release, or their priority and time to complete may change. The  
technical knowledge of the team gets very helpful in this planning session. The scrum  
team may prefer to complete certain issues before than others because of architecture  
design or some other technical issues that product owner does not have much  
knowledge about. An important measure which plays a vital role in release planning  
and estimation meeting is Velocity. If it’s not the first release, velocity of previous  
releases can be helpful in finding out number of sprint may take to complete a release.  
If the velocity is not known then team have to estimate the work that can be completed  
in a release.  
Initial estimate of the team can go wrong. In those situations team can hold another  
planning meeting to adjust the estimates based on real progress. With time velocity  
gets more predictable the release planning meeting start giving more accurate  
estimates based on velocity.

Timings: Beginning of each release or revised during the release

Duration: 4 – 16 hours  
Participants: product owner, scrum master, scrum team  
Best Practices:  
● Project owner come with prioritized and estimated features, so that team could  
create a schedule for the release.  
● Use the velocity to determine the number of sprints.  
● Don’t dictate technical team with your technical choices.  
● Don’t try to push features against the team recommendations.  
● If teams are very big than one person representing each area should attend the  
meeting.  
Artifacts: Release backlog.

**3. Review Meetings**  
a. Sprint Review (Demo) Meeting  
Purpose of the meeting  
Once sprint is completed Sprint team present a demonstration of the features  
completed during the sprint. The Product owner and customer review the features and  
give their comments on the completed features. Product owner may accept or reject  
the feature or point out the deficiency or bug in the work completed during the sprint.  
Features that fulfil the definition of done, provided by the product owner, are  
accepted. The fate of those features that are not completed or partially completed  
decided during next sprint planning meeting.

Timings: After sprint completion.  
Duration: 2 hours  
Participants: Product owner, customer, scrum team  
Artifacts: completed features.  
b. Sprint Retrospective  
Meeting Purpose of the meeting  
The purpose of Retrospective meeting is to discuss with team what processes or  
practices went well during the sprint? and what process and practices need to be  
improved? In this meeting only scrum team and scrum team participate. This is the  
last meeting of the sprint.  
Following are some of the examples.  
● Smaller task work better than larger ones  
● Definition of done need to be more clear  
● Information radiators need to be placed where everyone could see them  
● Daily meeting should not exceed 15 minutes.  
Timings: After Sprint Review meeting

Duration: 2-4 hours  
Participants: Scrum Team, Scrum Master  
Best Practices: Following are some of the best practices we can follow.  
● Use time line of the sprint to get feedback  
● Make team comfortable and relax  
● Try to let team to reach a consensus without scrum master interruptions  
● Don’t try to cover events outside of the sprint  
Artifacts: better processes and practices.

**4. Daily Stand-up**  
Meeting Purpose of the  
meeting  
The Daily stand-up or daily scrum meeting is the most important scrum meeting. The  
purpose of the meeting is to understand what team members are working on and  
propose them a solution for any impediment they may be facing. Each scrum team  
member answer following three questions in each of the meeting.  
What you have done since last meeting?  
What, if anything, is preventing you to perform your task?  
What you will do between now and next meeting?  
Timings: Daily  
Duration: 15 minutes  
Participants: Scrum master, scrum team (other can participate as a silent  
participants)

Best Practices: Following are some of the best practices we can follow.  
● Don’t exceed more than 15 minutes.  
● Only scrum team speaks, product owner can attend the meeting as silence  
participant  
● Fix the time for the meeting.  
● Don’t discuss anything out of three questions mentioned above.

**Artifacts**

**Product backlog**  
The product backlog comprises an ordered list of requirements that a scrum team  
maintains for a product. It consists of features, bug fixes, non-functional  
requirements, etc.—whatever needs doing in order to successfully deliver a viable  
product. The product owner orders the product backlog items (PBIs) based on  
considerations such as risk, business value, dependencies, and date needed.  
Items added to a backlog are commonly written in story format. The product backlog is  
what will be delivered, ordered into the sequence in which it should be delivered. It is  
open and editable by anyone, but the product owner is ultimately responsible for  
ordering the items on the backlog for the development team to choose.

The product backlog contains the product owner's assessment of business value and  
the development team's assessment of development effort, which are often, but not  
always, stated in story points using a rounded Fibonacci sequence. These estimates  
help the product owner to gauge the timeline and may influence ordering of backlog  
items; for example, if the "add spellcheck" and "add table support" features have the  
same business value, the product owner may schedule earlier delivery of the one with  
the lower development effort (because the return on investment is higher) or the one  
with higher development effort (because it is more complex or riskier, and they want to  
retire that risk earlier).  
The product backlog and the business value of each backlog item is the responsibility  
of the product owner. The size (i.e. estimated complexity or effort) of each backlog item is,  
however, determined by the development team, who contributes by sizing items,  
either in story points or in estimated hours.  
There is a common misunderstanding that only user stories are allowed in a product  
backlog. By contrast, scrum is neutral on requirement techniques. As the Scrum  
Primer states,  
Product Backlog items are articulated in any way that is clear and sustainable.  
Contrary to popular misunderstanding, the Product Backlog does not contain "user  
stories"; it simply contains items. Those items can be expressed as user stories, use  
cases, or any other requirements approach that the group finds useful. But whatever  
the approach, most items should focus on delivering value to customers.  
Scrum advocates that the role of product owner be assigned. The product owner is  
responsible for maximizing the value of the product. The product owner gathers input  
and takes feedback from, and is lobbied by, many people, but ultimately makes the  
call on what gets built.  
The product backlog is used to:  
● capture requests for modifying a product. This can include adding new features,  
replacing old features, removing features and fixing issues  
● ensure the development team is given work which maximizes the business benefit  
to the owner of the product  
Typically, the product owner and the scrum team come together and write down  
everything that needs to be prioritized and this becomes content for the first sprint,  
which is a block of time meant for focused work on selected items that can be  
accommodated within a timeframe. The product backlog can evolve as new  
information surfaces about the product and about its customers, and so later sprints  
may address new work.  
The following items typically comprise a scrum backlog: features, bugs, technical  
work, and knowledge acquisition. Web development can entail confusion as to the  
difference between a feature and a bug: technically a feature is "wanted", while a bug  
is a feature that is "unintended" or "unwanted" (but may not be necessarily a defective  
thing). An example of technical work would be: "run virus check on all developers'  
workstations". An example of knowledge acquisition could be a scrum backlog item  
about researching WordPress plugin libraries and making a selection.  
Managing the product backlog between product owner and scrum team  
A backlog, in its simplest form, is merely a list of items to be worked on. Having well-  
established rules about how work is added, removed and ordered helps the whole  
team make better decisions about how to change the product.

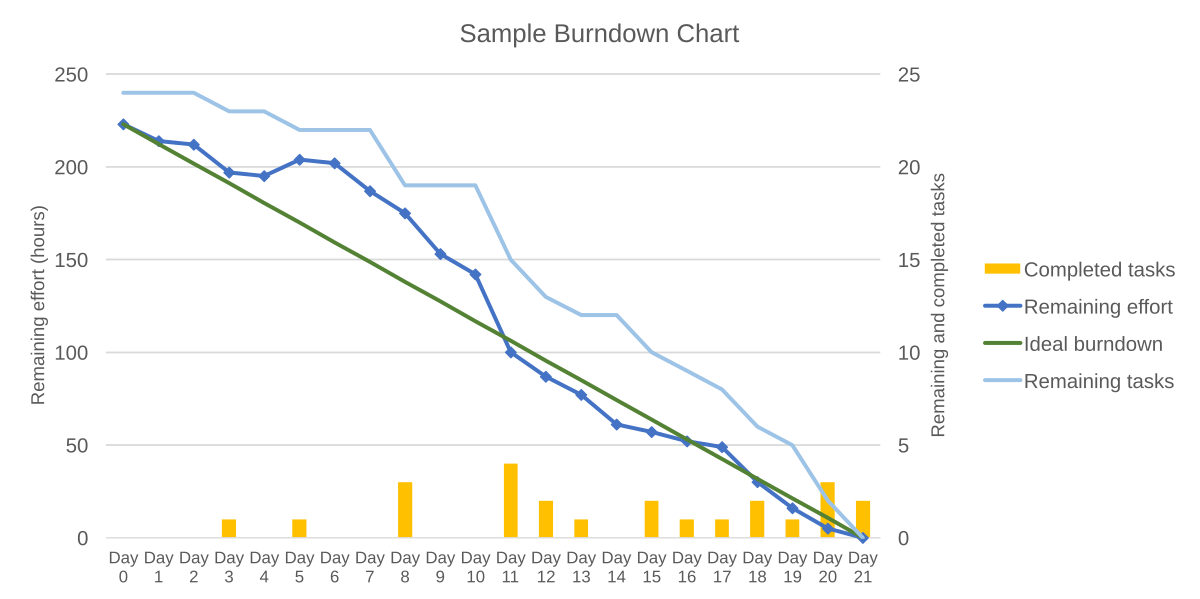


Figure no. 10: Sample Burndown Chart

The sprint burndown chart is a public displayed chart showing remaining work in the  
sprint backlog. Updated every day, it gives a simple view of the sprint progress. It also  
provides quick visualizations for reference. During sprint planning the ideal burndown  
chart is plotted. Then, during the sprint, each member picks up tasks from the sprint  
backlog and works on them. At the end of the day, they update the remaining hours  
for tasks to be completed. In such way the actual burndown chart is updated day by  
day.  
The following terms are often used in a scrum process.  
Sprint burn-down chart  
Daily progress for a sprint over the sprint's length.  
Release burn-down chart  
Feature level progress of completed product backlog items in the product backlog.  
Product backlog (PBL) list  
A prioritized list of high-level requirements.  
Sprint backlog (SBL) list  
A prioritized list of tasks to be completed during the sprint.

**Sprint**  
A time period (typically 1–4 weeks) in which development occurs on a set of backlog  
items that the team has committed to. Also commonly referred to as a time-box or  
iteration.

**Spike**  
A time boxed period used to research a concept and/or create a simple prototype.  
Spikes can either be planned to take place in between sprints or, for larger teams, a  
spike might be accepted as one of many sprint delivery objectives. Spikes are often  
introduced before the delivery of large or complex product backlog items in order to  
secure budget, expand knowledge, and/or produce a proof of concept. The duration  
and objective(s) of a spike will be agreed between the product owner and development  
team before the start. Unlike sprint commitments, spikes may or may not deliver  
tangible, shippable, valuable functionality. For example, the objective of a spike might  
be to successfully reach a decision on a course of action. The spike is over when the  
time is up, not necessarily when the objective has been delivered.  
Tasks  
Work items added to the sprint backlog at the beginning of a sprint and broken down  
into hours. Each task should not exceed 12 hours (or two days), but it's common for  
teams to insist that a task take no more than a day to finish.

**Definition of Done (DoD)**

The exit-criteria to determine whether a product backlog item is complete. In many  
cases the DoD requires that all regression tests should be successful. The definition of  
"done" may vary from one scrum team to another, but must be consistent within one  
team.

**Velocity**  
The total effort a team is capable of in a sprint. The number is derived by evaluating  
the work (typically in user story points) completed from the last sprint's backlog items.  
The collection of historical velocity data is a guideline for assisting the team in  
understanding how much work they can do in a future sprint.  
**Impediment**  
Anything that prevents a team member from performing work as efficiently as  
possible.  
**Conclusion**  
Scrum is an agile process most commonly used for product development, especially  
software development. Scrum is a project management framework that is applicable to  
any project with aggressive deadlines, complex requirements and a degree of  
uniqueness. In Scrum, projects move forward via a series of iterations called sprints.  
Each sprint is typically two to four weeks long.

**PRACTICAL - 4**

Fake News Detection Recommendation

This project presents a web-based solution designed to identify and classify misleading or false news articles. By utilizing Natural Language Processing (NLP) and Machine Learning (ML) techniques, the system enables users to input news content, which is then analyzed to determine its authenticity. The aim is to promote digital truth and minimize the spread of misinformation.

**Objective**

To build a smart and reliable application that leverages machine learning algorithms for detecting and flagging fake news content, thereby encouraging information accuracy and improving the trustworthiness of online platforms.

**Vision Statement**

To foster a digitally literate society by offering AI-based tools that evaluate the credibility of online information, helping individuals make informed decisions and counteract the negative impact of false news.

**Use Case Diagram:**

A Use Case Diagram in UML (Unified Modeling Language) visually represents how users interact with the Fake News Detection System. It outlines the primary functions of the application by mapping user actions (such as submitting news or viewing results) to system responses. These diagrams serve as a simplified model showing the relationship between the user (or actor) and the system’s features.

This high-level view helps project stakeholders, developers, and analysts understand how the system is expected to behave from the user’s perspective. It also clarifies which operations are available, how the system handles different requests, and how each process fits into the overall structure. This approach is especially helpful for identifying system scope and ensuring all essential interactions are accounted for during development.

**Actors:**

* User

**Use Cases:**

* Sign Up/Sign In
* Submit news Article or link
* Analyze submitted link
* Generate Authenticity Verdict
* Display Results to User

**Visual representation:**

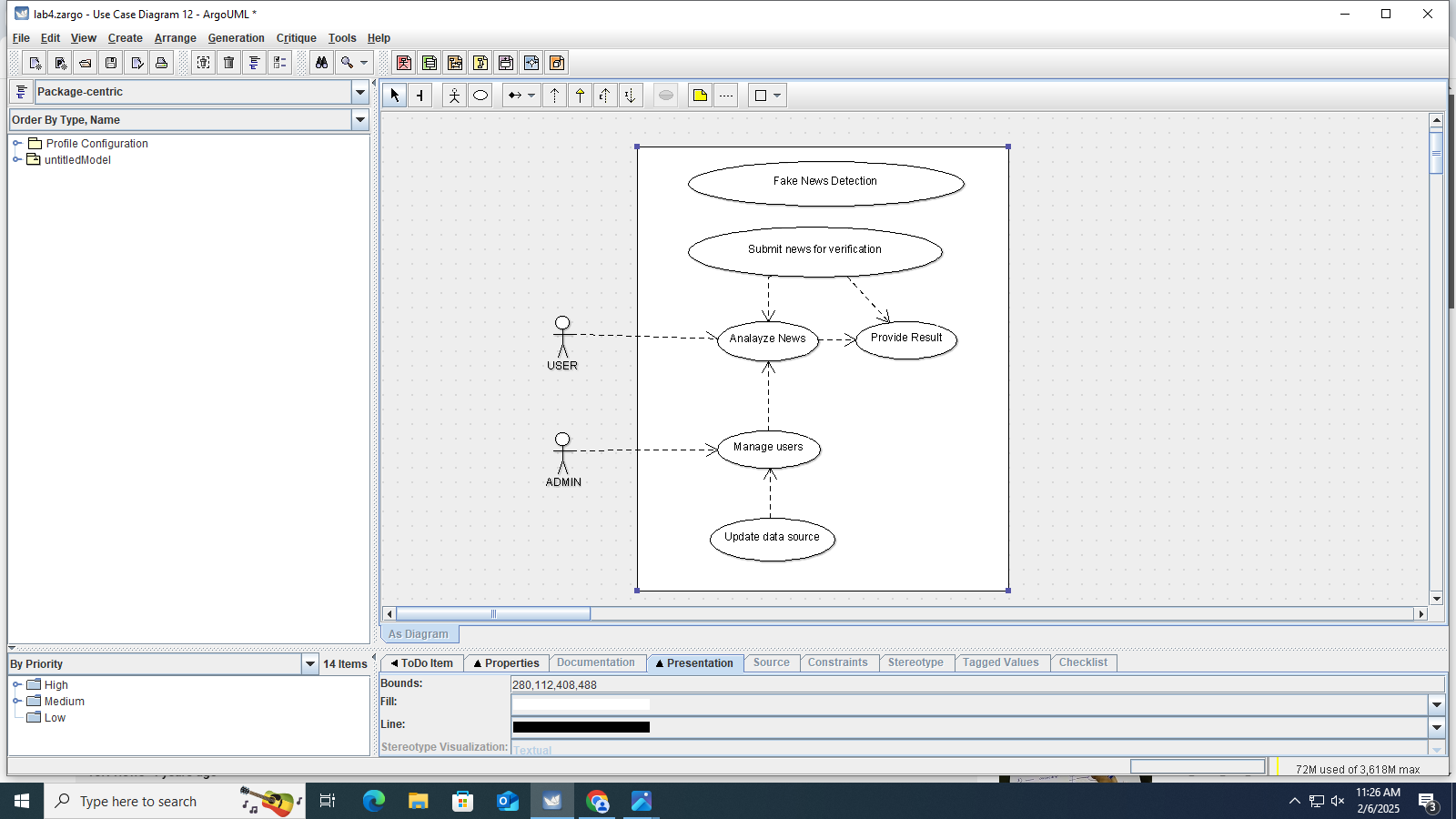


Figure no. 11: use case diagram

**PRACTICAL – 5** Fake News Detection Recommendation’s Release Map and User Stories

**Release Map**

A Release Map is a timeline-based structure that outlines the major development stages, deliverables, and goals over the course of the project. It helps the development team prioritize tasks, allocate resources effectively, and ensure that every release brings measurable progress toward the final system. In this project, the release map is divided into five key phases carried out over six months, each focusing on essential components of the system.

The first month is dedicated to research and planning, where the scope is defined, relevant datasets are collected, and the tools and technologies required for development are selected. The next two months focus on model development, where text data is preprocessed, machine learning algorithms are trained and tested, and the most accurate model is selected for deployment.

In the fourth month, the team works on building the MVP (Minimum Viable Product), which includes developing a user-friendly interface and integrating the trained model into a working web application. This version of the system allows users to input news content and receive predictions about its authenticity.

The fifth month is reserved for beta testing, which includes functionality testing, bug fixes, and gathering user feedback to improve the system. Finally, in the sixth month, the fully tested and refined system is deployed to a live environment, where it is monitored and maintained. This phase also includes plans for future iterations and upgrades based on real-world usage.

A screenshot of a computer

AI-generated content may be incorrect.

Figure no. 12: Release map

**User Stories**

1️. User Story: News Content Submission- *As a general user,I want to submit a news article or URL into the system,So that I can verify its authenticity before trusting or sharing it.*

2️. User Story: Instant Fake/Real Prediction- *As a reader,I want the system to instantly analyze the news I provide,So that I can get immediate feedback on whether it is fake or genuine.*

3️. User Story: Prediction Explanation - *As a curious user,I want to see an explanation of why the article is classified as real or fake,So that I can understand how the system made its decision.*

4️.  User Story: Feedback Reporting - *As a responsible user,I want to report incorrect results back to the system, So that the detection model can be improved over time.*

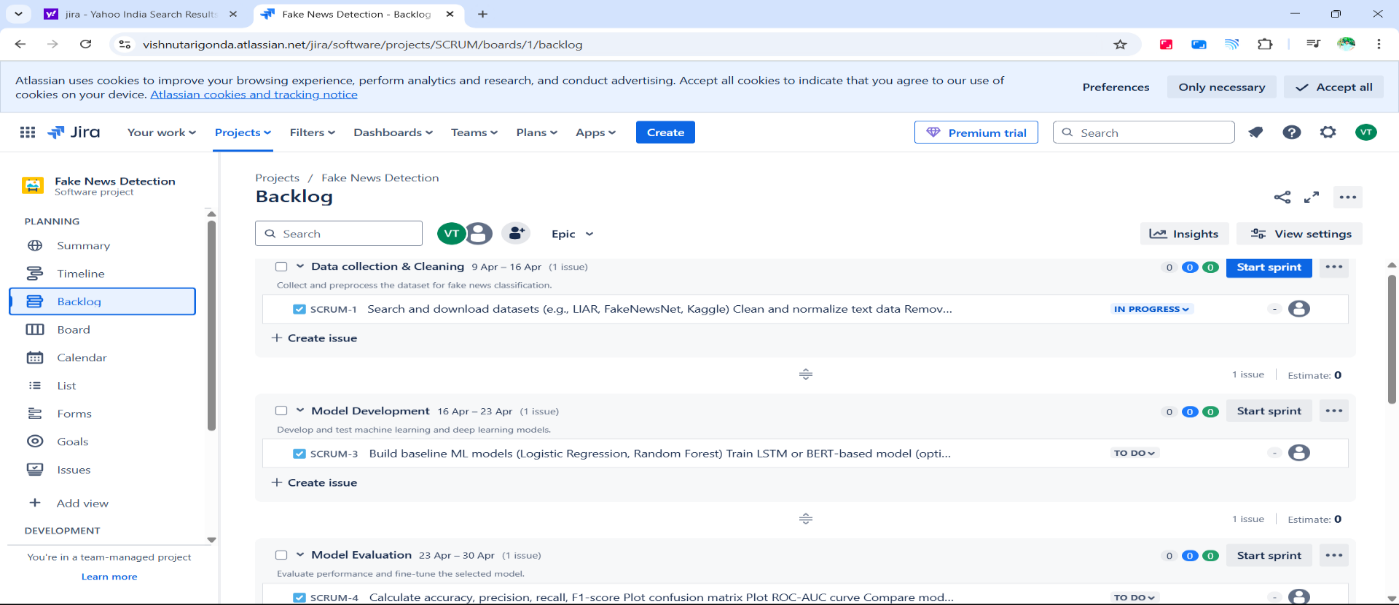


Figure no. 13: User stories

A screenshot of a computer

AI-generated content may be incorrect.

Figure no. 14: User stories

**PRACTICAL – 6**

Identifying Storyboarding Tasks & Release Plan

## **Breaking User Stories into Tasks**

Each user story is broken down into tasks for development, design, testing, and deployment.

### News Content Submission

### **Tasks:** Design input form for text or URL submission Validate input data (text or link) Integrate AI model to analyse skill gaps Develop backend API for submission handling Test UI and backend integration

### Instant Fake/Real Prediction

### **Tasks:** Connect ML model to backend logic Trigger prediction on user submission

Display result as "Fake" or "Real" with confidence score

Test response speed and accuracy

Prediction Explanation

**Tasks:**  
Generate summary of keywords/phrases influencing the prediction  
Add visual indicators for transparency  
Develop explanation display component  
Test with multiple news inputs for clarity

Feedback Reporting

**Tasks:**

Create feedback/reporting form  
Link reports to user accounts (if logged in)  
Store reported results for model improvement

User Account Management

**Tasks:**  
Implement user login/registration system  
 Store and retrieve user analysis history  
 Add password encryption and session handling  
 Test secure login/logout flow

## **Sprint Planning & Team Assignment**

Using Agile **Scrum methodology with 2-week sprints:**

**Sprint 1–2: Core System & MVP Setup**

Set up project architecture and development environment  
Build frontend interface for news input (text/URL)  
Integrate dataset and begin model training

**Team**: Backend Devs (2), Frontend Devs (2), ML Engineer (1), UX Designer (1)

**Sprint 3–4: Model Enhancement & Explanation Module**

Finalize and deploy best-performing ML modelBuild explanation engine to justify predictions

**Team**: Backend Devs (1), ML Engineer (2), Data Analyst (1),Front End Developer(1)

**Sprint 5–6: User Management & Feedback System**

Implement user login, registration, and session handling

Create feedback/report module for misclassified results

Enable history tracking for logged-in users

**Team:** Backend Developer(1),Frontend Developer(1),Database Engineer(1)

Sprint 7–8: Final Testing & Deployment

Perform end-to-end testing of all modules  
Deploy application on a live server  
Prepare system for final demonstration

**Team**: QA Testers (2), Full Team for final fixes

A screenshot of a calendar

AI-generated content may be incorrect.

Figure no. 15: Sprints

**PRACTICAL – 7**

Design Product Roadmap

A Product Roadmap is a strategic, high-level visual summary that outlines the vision, direction, and progress of a product over time. In Agile development, it helps guide the team in delivering incremental value, aligning project goals with stakeholder expectations. For the Fake News Detection System, the roadmap defines clear development phases over a span of 12 months, ensuring steady progress from core functionality to deployment and future enhancements.

The roadmap is structured into four key phases:

In Phase 1 (Months 1–3), the focus is on laying the foundation by identifying system requirements, collecting datasets, and building the minimum viable product (MVP). During this phase, the basic news input interface and the initial machine learning model are developed and integrated.

Phase 2 (Months 4–6) emphasizes enhancing the model and integrating additional features such as explainable AI, user authentication, and a feedback system. This ensures the system is functional and ready for early testing.

During Phase 3 (Months 7–9), the system is expanded with analytics, reporting tools, and improved user transparency. Feedback from real users is collected and used to refine model performance and usability.

Finally, Phase 4 (Months 10–12) focuses on deployment, stability, and optimization. The application is made publicly accessible, and final testing, security checks, and performance improvements are completed. This phase also includes preparation for long-term updates like support for multiple languages or mobile platforms.

The product roadmap ensures that the Fake News Detection System is developed in a structured, phased manner—delivering working features at every step while allowing room for improvement through continuous user feedback and testing.

A screenshot of a calendar

AI-generated content may be incorrect.

Figure no. 16: Road Map

**PRACTICAL – 8**

Design Story Mapping

Story Mapping is a visual and strategic technique used in Agile development to organize user stories based on the flow of user interactions within a system. It provides a structured overview of the user journey by aligning features and tasks in a sequence that reflects real-world usage. This helps the development team prioritize functionality, understand system behaviour, and focus on delivering the most valuable features first.

For the Fake News Detection System, the story map outlines the full user experience—from initial engagement to the final output. The journey begins with basic access features like signing in, followed by the core process of submitting a news article, running it through the detection model, and receiving an authenticity result. Additional features such as explanation of results, user feedback, and account-based tracking are layered in to enhance functionality and user trust.

Each major function is broken down into smaller, actionable user stories and development tasks. This structure helps the team identify what is essential for the Minimum Viable Product (MVP) and what features can be added in future iterations. It also ensures that development aligns with user needs while maintaining flexibility for feedback-driven updates.

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure no. 17: Story Mapping

**PRACTICAL – 9**

Compiling the Product Backlog

In Agile development, a Product Backlog is a dynamic and prioritized list of all features, enhancements, bug fixes, and technical requirements that are needed for a system. It represents the complete scope of the project from a development perspective and serves as the single source of work items for the team.

The backlog is typically composed of epics, user stories, and tasks. Epics represent broad functionalities, while user stories break those down into smaller, user-focused components. Tasks are further technical actions required to implement the stories.

For the Fake News Detection System, the product backlog is organized around core functionalities such as user input, news classification, prediction explanation, feedback submission, and user account management. These are broken into stages to support the system's incremental development and continuous improvement.

The early sprints focus on building the Minimum Viable Product (MVP)—the essential features required for users to input news and receive detection results. Mid-stage sprints introduce deeper functionality such as explanation logic, login systems, and feedback handling. Final sprints address security, UI improvements, and deployment readiness.

A screenshot of a computer

AI-generated content may be incorrect.

Figure no. 18: Backlog

**PRACTICAL – 10**

Compiling the Sprint Backlog

A Sprint Backlog is a subset of the product backlog that includes the tasks and user stories selected for implementation in a specific sprint. It outlines the immediate work to be done by the development team during a short, time-boxed iteration—typically 1 to 2 weeks in Agile Scrum.

The sprint backlog is created during sprint planning and is driven by the team’s capacity and the priority of features. It provides a clear and focused view of what needs to be built, tested, and delivered within the sprint. Each item in the sprint backlog is broken down into technical tasks, allowing developers to work efficiently and track progress daily.

For the Fake News Detection System, the sprint backlog is structured to progressively implement all major functionalities. Early sprints cover the foundation—input forms, basic ML model integration, and displaying results. Mid-stage sprints handle user management, prediction explanation, and feedback systems. Final sprints focus on optimization, bug fixes, and deployment preparation.

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure no. 19: Sprint Backlog

**PRACTICAL – 11**

Demonstrate Sprint Releases

In Agile development, Sprint Releases refer to the outcome of each sprint, where a potentially shippable product increment is delivered. These releases allow teams to showcase progress, test real functionalities, and gather feedback regularly. Demonstrating sprint releases provides transparency, tracks project velocity, and ensures that each iteration adds value to the overall system.

For the Fake News Detection System, sprint releases are planned in 2-week cycles. Each sprint builds upon the previous one, gradually developing the system from a basic prototype to a complete and user-friendly application. The output of every sprint includes implemented features, tested modules, and refined components.

During Sprint 1, the core system is developed, including input submission and basic model integration. By Sprint 2, enhancements such as feedback capture and improved result output are added. Subsequent sprints introduce login features, explanation mechanisms, and performance optimization.

To track sprint progress, tools like Jira boards and Burndown Charts are used. The sprint board organizes tasks into “To Do,” “In Progress,” and “Done” columns, helping the team visualize their workflow. The burndown chart monitors remaining work over time, ensuring the team stays on track with goals.A screenshot of a computer

AI-generated content may be incorrect.

Figure no. 20: Product Summary

**PRACTICAL – 12**

Final Demonstration of the Entire Project

The final demonstration is a key stage in Agile-based software development. It involves presenting a fully functional version of the system that has gone through all development phases, including planning, design, implementation, testing, and deployment. The goal is to showcase the final product in its most complete and stable form, reflecting all major features outlined in the original scope.

For the **Fake News Detection System**, the demonstration includes end-to-end execution of core features:

* A user inputs a news article or URL.
* The system processes the input using NLP techniques.
* A trained machine learning model classifies the news as “Fake” or “Real.”
* The system displays the result along with a brief explanation.
* Logged-in users can view their history and submit feedback on the result.

Throughout the development cycle, the system was refined over multiple sprints using Agile Scrum. Regular reviews, feedback integration, and retrospectives ensured continuous improvement and alignment with user needs. The final version includes secure user authentication, a responsive UI, real-time prediction capabilities, and a feedback loop for future model training.

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure no. 21: Scrum sprint

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure no. 22: Summary

A screenshot of a computer

AI-generated content may be incorrect.

Figure no. 23: Priority Breakdown

Conclusion:

The Fake News Detection System successfully demonstrates how machine learning and natural language processing can be used to combat the spread of misinformation. Developed using Agile methodology, the project delivered a functional, user-friendly platform capable of identifying fake news in real time. The system meets its objectives of enhancing media credibility and promoting responsible information sharing. This project highlights the practical application of AI in addressing real-world challenges in the digital age

REFERENCE

[1] A. H. Alhindi, S. M. Petridis, and H. M. Saba, "Detecting Fake News on Social Media Using Machine Learning Techniques," *Procedia Computer Science*, vol. 179, pp. 1045–1052, 2021. [Online].  
Available: https://doi.org/10.1016/j.procs.2021.01.088

[2] R. Oshikawa, J. Qian, and W. Y. Wang, "A Survey on Natural Language Processing for Fake News Detection," in *Proceedings of the 12th Language Resources and Evaluation Conference*, Marseille, 2020, pp. 6086–6093. [Online].  
Available: https://aclanthology.org/2020.lrec-1.747/

[3] K. Shu, D. Mahudeswaran, and H. Liu, "FakeNewsNet: A Data Repository with News Content, Social Context and Dynamic Information for Studying Fake News on Social Media," *Big Data*, vol. 8, no. 3, pp. 171–188, 2020. [Online].  
Available: https://doi.org/10.1089/big.2020.0062

[4] T. A. Conroy and M. R. Rubin, "Automated deception detection: Methods for finding fake news," in *Proceedings of the 78th ASIS&T Annual Meeting*, St. Louis, MO, USA, 2021. [Online].  
Available: <https://asistdl.onlinelibrary.wiley.com/doi/full/10.1002/pra2.2021.14505801070>

[5] H. Allcott and M. Gentzkow, "Social Media and Fake News in the 2016 Election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–236, 2017. [Online].  
Available: https://doi.org/10.1257/jep.31.2.211