ELEC 490 Project Blueprint

AI for Fake News and Deepfake Detection

Submitted by Group 16:

Ivan Samardzic - 21is8, 20296563

Erhowvosere Otubu – 21eo4, 20293052

Mihran Asadullah – 20ma108, 20285090

GitHub Repository: [Group 16 Fake News Detection Capstone Project]

Faculty Supervisor: Ali Etemad

November 3rd, 2025

Executive Summary

With the rise of misinformation in digital media, the spread of inaccurate health information has become a critical concern, capable of negatively influencing public healthcare decisions and undermining trust in professional guidance. This project aims to build an Artificial Intelligence (AI) based web system used for automatic detection of medical-related misinformation. The primary objective is to deliver a minimum viable product (MVP) that provides classification of text (True, False, or Uncertain), alongside respective accuracy metrics to justify validity and model conclusions. Beyond these core requirements, aspirational goals include the development of a browser extension and the addition of multimodal detection for image and video deepfakes.

The most critical product specifications revolve around a classification F1-score accuracy of 80%, and end-to-end latency of 3.0 seconds, reflecting the need for both model correctness and real-time usability. Additionally, the inclusion of the explainability layer serves as the ethical safeguard between the model and end user, providing full prediction transparency. The team has adopted an Agile methodology of shorter sprints, allowing the team to work concurrently and preform testing on the fly, all while keeping the general division of labour consistent between the AI Pipeline (Mihran), Backend API (Application Programming Interface) (Sere), and Frontend interface (Ivan). The systematic testing approach allows for rigorous component validation, followed by a user test that simulates 50 concurrent users to ensure the system's 1% error rate is not exceeded.

The cost of building this project consists of GPU access from Google Colab, estimated at $15.81 CAD/month. The project emphasizes efficiency while relying on open-source tools to keep costs low. This document provides an overview of the full-stack architecture, AI design, testing strategy, and resource allocation of the product. By combining Natural Language Processing (NLP) with an accessible user interface and thoughtful ethical considerations, this project aims to provide an efficient tool to navigate through healthcare information with confidence.

Table of Contents

List of Tables

List of Figures

1. Introduction

The rise of digital media has transformed how people access their news and information, but has also amplified the spread of deception in news, especially in sectors where accuracy is vital. In healthcare specifically, falsified articles can lead to several harmful consequences, such as risky health choices, unsafe self-treatment, and a skewed perception of healthcare in the eyes of the public. During global healthcare crises such as the COVID-19 pandemic, where procedures and guidelines are far understudied, misinformation has the ability to propagate through digital media at rates exceeding the ability for manual validation. A Canadian study estimated that, had misinformation on social media not been present, 198,000 COVID-19 cases and 2,800 fewer deaths would have occurred, as citizens would've accepted the pandemic's severity [1].

While existing approaches for validating misinformation do exist, they are largely limited in scope and require manual reviewers or general-purpose text classifiers, which may not be optimized for medical content specifically. Often times, manual reviewers even contribute to the further spread of misinformation themselves [2]. These existing validation methods generally are not able to process at a scale required to limit the spread of inaccurate claims. This directly highlights the need for an automated method of evaluating news credibility with real-time feedback, specific to a domain such as healthcare and human wellness.

1.1 Design Problem

The central design problem of this project revolves around creating a reliable, low-latency, and ethically responsible artificial intelligence (AI) model for classification of medical misinformation. This project aims to develop a website, where users are able to upload articles or paste text as input, the system then outputs classification results of “True”, “False”, or “Uncertain”. These classification labels will be accompanied by their respective confidence scores, as well as a contextual description to help users visualize the rationale behind the model's predictions.

Within the project design, three primary technical constraints are focused on. Primarily, the core task consists of achieving high accuracy in classifications. This involves selecting and fine-tuning the large transformer models (e.g. BERT/RoBERTa) on specialized medical datasets in order to accurately interpret domain-specific language. Success is classified as achieving high model accuracy on unseen data. Secondly, the system must adhere to performance specifications regarding response time. The final architecture must efficiently integrate the computationally intensive AI pipeline with a scalable backend (Node.js/FastAPI) to deliver near real-time predictions. Finally, a crucial ethical constraint is the involvement of user transparency. The solution must provide the end user with a confidence score and highlighted keywords to actively mitigate the risk of automation bias, which is further explained in 5.1 Health and Safety Risks.

There have been no modifications to the design problem or scope as defined in the original proposal. The focus remains explicitly on the text-based healthcare domain, with image/video detection and browser extensions reserved as optional future goals. These constraints have been discussed and approved by the faculty supervisor.

1.2 System Specification

The project is comprised of several specifications designed to assist with model development, align adequate testing, and deliver a highly practical final product. These specifications are divided into several key requirements for the minimum viable product (MVP), as well as aspirational goals for enhanced functionality and product improvements. All performance metrics closely follow industry benchmarks specific to the high-stakes nature of healthcare and human wellness.

The following set of specifications in Table 1 outline the minimum functional, interface, and performance requirements necessary for successful project completion.

Table 1: Description of required system specifications for MVP.

Functional requirements	**Target value**	**Tolerance**
Classification Output	3-way label: True, False, Uncertain	N/A
Input Acceptance	Raw Text (up to 5,000 words), or article link	N/A
Explainability Layer	Output includes confidence score and > 2 highlighted keywords/phrases	N/A
API Integration	Successful communication between Frontend, Backend, and AI Pipeline	N/A
Interface requirements	**Target value**	**Tolerance**
Cross-Browser Compatibility	Chrome, Firefox, Edge, Safari (latest two major versions)	N/A
Responsive Design	Full functionality and legibility on screen widths > 360px	N/A
Ethical Disclosure	Mandatory, persistent disclaimer visible on the primary interface	N/A
Performance requirements	**Target value**	**Tolerance**
Classification Accuracy (F1-Score)	> 80%	5%
End-to-End Latency	< 3.0 seconds (Time from user click to result display)	1.0 seconds
Reliability	< 1% server error rate during continuous operation (1 hour)	N/A

The targets listed above are justified by comparing general Natural Language Processing (NLP) and transformer performance metrics with real-world expectations, as well as ethical compliance.

When referring to classification accuracy, the F1-score is chosen as the most appropriate performance metric for classification correctness. The F1-score is a machine learning evaluation metric that combines the precision (how many of its positive predictions were correct) and recall (how many of the actual positives cases it found) scores of a model to measure it's accuracy [3]. Due to class imbalance and the existence of false positives/negatives, the F1-score handles error by penalizing both results equally, guaranteeing a balanced and trustworthy classifier [4]. While transformer models generally have no issues achieving high classification scores on general datasets, the complexity of medical terminology as well as difficulty in obtaining medical domain specific data justify the proposed target of 80%. In fact, an F1 score of 80% and greater is considered to be a good score with these factors taken into account [4].

End-to-end latency is a measure of delay from the moment a user prompts the website to analyze text, to the moment the resulting outputs are displayed. This is a key metric in determining whether the system is usable and practical. Web performance research often indicates that a website's bounce rate often increases if the overall system latency exceeds 5 seconds [5]. As a result, achieving a latency goal within 3.0 seconds is a high indicator of webpage usability success.

The last measure of system performance is reliability. This is the fundamental gauge of the system stability and is dictated by the server error rate. For a tool intended to be used in critical domains such as healthcare, an error rate under 1% ensures that the application is continuously available without crashing or returning server errors [6]. A higher reliability score indicates that the tool is dependable and builds user confidence.

With the justification for specification targets in mind, the following aspirational goals listed in

Table 2 represent features and performance optimizations that will be pursued following the successful validation of the MVP.

Table 2: Description of optional (enhanced) system specifications.

Functional requirements	**Target value**	**Tolerance**
Multimodal Detection	Basic image classification for deepfakes/manipulation (prototype)	N/A
Browser Extension	Functional content extraction and analysis on common article pages	N/A
Interface requirements	**Target value**	**Tolerance**
Design System Adoption	Adherence to a standard component library (e.g., Tailwind UI)	N/A
Performance requirements	**Target value**	**Tolerance**
Classification Accuracy (F1-Score)	> 85%	5%
End-to-End Latency	< 2.0 seconds	0.5 seconds

2. Methodology/Work Breakdown Structure

2.1 Approach

The team's overall strategy is to follow a logical, milestone-driven approach with established software engineering principles. This approach is a deliberate application of formal methodologies to reduce possible risk associated with the project by ensuring that the three core components (AI Pipeline, Backend, Frontend) can be developed and tested independently before final integration.

2.1.1 Logical Sequence of Steps

The development sequence is structured into four primary phases, which are aligned with the eleven major milestones outlined in more detail in Section 3. Milestone/Division of Labour. The initial phase is to validate the project's core hypothesis. The team (primarily Mihran) will focus on the AI Pipeline. This process involves collecting labeled healthcare misinformation datasets and fine-tuning the baseline BERT/RoBERTa models. The goal is to have a model that achieves the baseline success criterion (> 80% accuracy) on a validation dataset.

Once the team has confirmed the core logic is working as intended, the team will move onto Frontend and Backend development. These modules are developed independently, based on the APIs (Application Programming Interface) that dictates how the two modules will communicate. For the backend development, a secure Node.js and Python-interfacing RESTful and Fast APIs will be built to serve model predictions. This module is developed and unit-tested in isolation, initially using mock data from the AI pipeline.

Next will be modular and integration testing. Each of the three models are first tested independently to verify its own functionality. End-to-end user testing will be conducted to assess the full application’s performance, usability, and reliability. A request is initiated from the Frontend, travels to the Backend API, is processed by the AI Pipeline, and then the result is returned and displayed to the user, following the block diagram (Figure 1) from the Proposal report [7]. This phase is dedicated to identifying and fixing bugs.

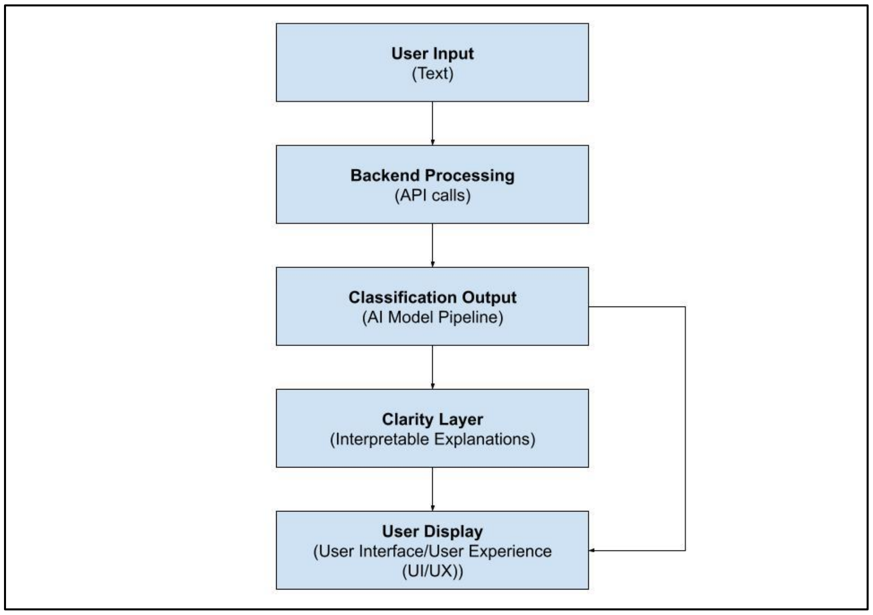


Figure 1: Block diagram from Proposal Report outlining the system’s workflow [7]

Lastly, in the final phase, all tested and stable modules are merged into the final, cohesive web application.

To illustrate how a user interacts with this web application, Figure 2 below provides a visual guide to the User Interface (UI) components and maps them to the system's logical workflow from Figure 1.

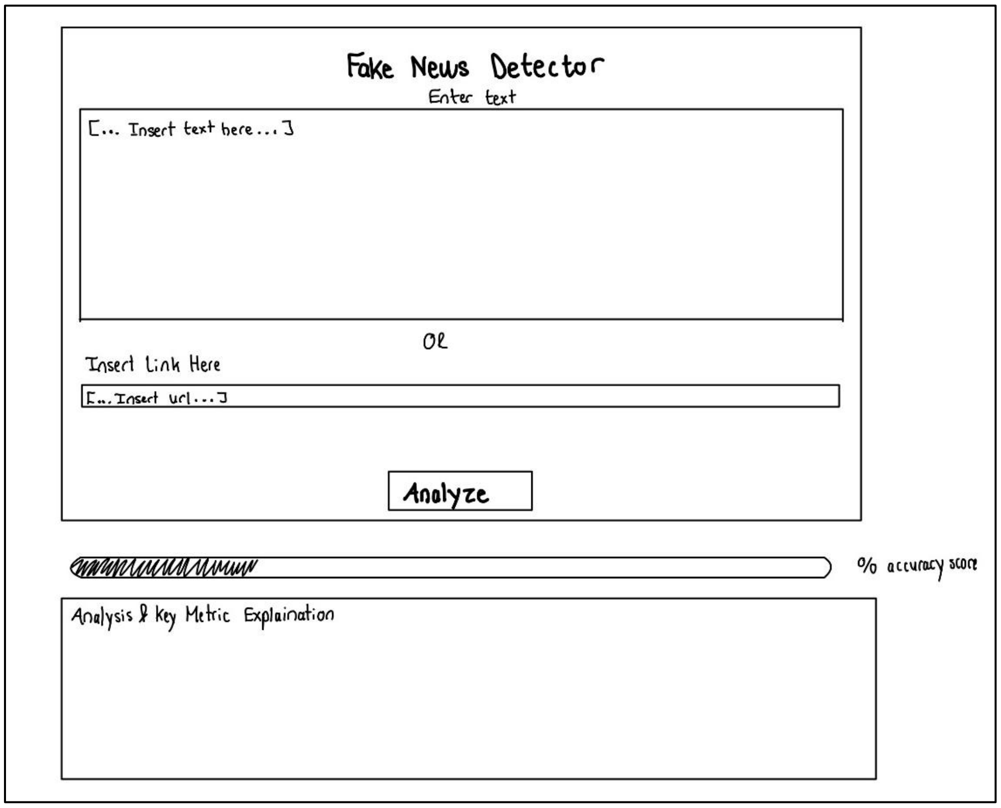


Figure 2: Visual guide of the UI components from the system's workflow

This UI sketch demonstrates the practical application of the system's workflow, connecting the user's actions to the backend and pipeline processes. First, the user initiates the process by providing input data (i.e., a sentence or an article website link) which maps directly to the "User Input (Text)" workflow block. When the user clicks the "Analyze" button, they trigger the "Backend Processing (API calls)" block. If a URL was provided, this block's first action is to make an API call to a scraping service to collect the article's text. Once the text is gathered, a second API call sends this data to the model. On the frontend side, clicking the "Analyze" button returns two distinct forms of data as visualized in the workflow diagram. With regards to the

“Classification Output (AI Model Pipeline),” the results from the model are rendered as a percentage score. This represents the model’s confidence in its classification and is the raw prediction. For the “Clarity Layer”, the model’s output is also fed into this layer which generates a qualitative, readable summary for the user. This is displayed in the “Analysis & Key Metric Explanation” box, which might include details like keywords or sentences that most strongly influenced the model’s final decision.

2.1.2 Guiding Principles and Methods

The team's workflow will be guided by a combination of three industry-standard software engineering principles. These principles work together to provide a framework for managing the project. A detailed modular design will allow the team to break the system down into subsystems so that each can be developed and tested independently before integration. Following an Agile methodology will allow the team to tackle sprints with regular check-ins, so that project can adapt as new challenges or feedback arise. Lastly, iterative prototyping will allow the team to break the project into successive versions to gradually refine performance.

2.1.2.1 Modular Design

The team’s system is broken down into three distinct subsystems which form a 3-tier architecture. The multi-tier architecture, also known as the n-tier architecture, is a design pattern in software engineering where an application is divided into multiple logical layers each responsible for specific aspects of the application’s functionality [8]. The purpose of this architecture is to organize code into more manageable sections, improve scalability, and separate concerns, making the application easier to develop, maintain, and scale [8]. This design pattern separates the system into a Presentation Tier (Frontend), an Application Tier (Backend API), and Data/Logic Tier (AI Pipeline) [9]. The main benefit of this model design is that the three tiers are developed and maintained independently, allowing any of the three tiers to be upgraded or replaced independently in response to changes in requirements [9]. This architecture is the foundation of the project workflow and directly aligns with the division of labour detailed in this report, where each team member has clear ownership. This separation of concerns helps to prevent one part of development from holding up another part.

2.1.2.2 Iterative Prototyping

The team will employ an iterative prototyping model to help guide the development of the system. Unlike traditional means of development, where testing is deferred to the final stages, the strategy follows a process that ensures a functional and testable product at every major prototype. This will help enable early validation, continuous feedback, and agile adaptation to challenges.

Each version of the process represents a meaningful step towards the final product with increasing functionality and integration. The first prototype version consists of an AI pipeline and backend infrastructure with the fine-tuned BERT/RoBERTa model. It is implemented as a Python script that takes in input from the command line and classifies healthcare related text into three categories, "True", "False", or "Uncertain", and then outputs a percentage confidence score ranging from 0 to 100%. In prototype 2, the model is wrapped in a FastAPI interface and exposed via a Node.js backend. This version allows for internal backend routing, API testing and latency measuring. It marks the first step to modular integration. By prototype 3, the Backend API is connected to a React-based frontend. This version allows for users to input and submit text or URLs (Uniform Resource Locator) through a web interface and receive classification results with a percentage confidence score. This will serve as a foundation for useability testing and UI feedback. The final version will be able to correctly classify text and highlight key phrases with percentage confidence scores. It will also include all performance optimizations with the user feedback changes implemented. Additionally, this version will implement a browser extension that will allow users to use the model across various tabs.

2.1.2.3 Agile Development

The team will adopt an Agile development methodology to manage the team and individual's progress effectively and respond to new challenges with ease. The eleven project milestones serve as the team's development sprints. Each sprint is approximately 3-6 weeks with a clear set of deliverables, applying the agile principle of delivering working software frequently [10]. To ensure synchronization between team member's independent work, the team will hold weekly meetings. These meetings are used to review progress, identify, and resolve obstacles, and adapt the plan. This upholds the agile value of “individuals and interactions over processes and tools” [10]. This agile structure allows the team to adapt to new challenges, as identified in the team’s Contingency Plan from the Proposal report [7]. This embodies the principle of “responding to change over following a plan” [10]. For example, if the team’s weekly check-ins during Milestone 1 reveals that RoBERTa is underperforming, the team can immediately pivot to evaluate BERT or BioBERT in the next cycle without hindering the Frontend or Backend sprints.

2.2 Design Tools, Hardware, Instrumentation

To successfully implement the AI-powered fake news detection system, the team has chosen a strong set of tools and platforms to support the development, testing, deployment, and collaboration needed for a scalable web application. These resources include software frameworks, hardware platforms, development environments, and data sources. Each resource was selected to fit the project's technical goals, budget limits, and timeline.

2.2.1 Software Tools

The software stack in Table 3 used for this project is categorized into several key areas:

Table 3: An outline of the software stack used for the project

Software	**Purpose**
Operating Systems: Windows/MacOS	Team members are using a mix of operating systems based on personal preference and hardware availability. All tools are cross compatible to ensure no issues arise during development
Programming Language: Python [11]	Python is the primary language for developing the AI pipeline. Its extensive library of machine learning packages, including PyTorch and Hugging Face Transformers, makes it more than ideal for tasks.

Programming Language: JavaScript / TypeScript	These frameworks are used for frontend development and browser extension logic.
IDE: Visual Studio Code (VS Code) [12]	VS Code is the primary integrated development environment (IDE) used by all team members. It supports extensions for Python, JavaScript, TypeScript, Git integration, and debugging tools, making it a versatile environment for project development.
IDE: Google Colab [13]	Google Colab provides GPU-powered notebooks for training and testing machine learning models. It is particularly useful for training complex models with large datasets
CI/CD Tool: GitHub	GitHub is used for version control, issue tracking, and collaborative development. It enables the team to manage code changes, conduct peer reviews, and maintain a centralized repository.

Table 4: Overview of core frameworks and libraries used in the project

Framework / Library	**Description**
PyTorch [14]	A widely adopted deep learning framework that supports dynamic computation graphs and GPU acceleration. It is used to fine-tune transformer models such as BERT and RoBERTa on domain-specific healthcare misinformation datasets.
Hugging Face Transformers [15]	Provides access to state-of-the-art pre-trained NLP models and tokenizers. Simplifies the integration of transformer architectures and supports fine-tuning with minimal overhead.

FastAPI [16]	A modern, high-performance web framework for building APIs with Python. It serves the AI model and handles classification requests from the frontend. Its asynchronous capabilities and automatic documentation generation make it ideal for scalable backend services.
Node.js [17]	Used to build the backend infrastructure that interfaces with FastAPI and manages routing, security, and request handling. Its event-driven architecture supports high-performance web applications.
React [18] / Next.js [19]	React is used for building the user interface, while Next.js provides server-side rendering and routing capabilities. This combination ensures a responsive and performant frontend experience across devices and browsers.
Chrome / Firefox Extension APIs	These APIs will be used to develop a browser extension that allows users to analyze content directly from web pages. This feature enhances accessibility and usability by enabling real-time misinformation detection across tabs.

2.2.2 Hardware and Instrumentation

Although the project is primarily software-based, certain hardware platforms are essential for development and testing. Each team member will be using a personal laptop or desktop that either meets or exceeds the minimum specifications required for this project. These machines are sufficient for local development, frontend web testing, and lightweight model training/testing.

For more intensive tasks such as model fine-tuning, cloud-based resources will be used. This includes Google Colab for training and fine-tuning transformer models. The free tier provides access to GPUs suitable for small-scale experimentation, while the Pro tier offers faster and more reliable access for time-sensitive tasks. Additionally, the team has access to Queen’s University’s academic high-performance computing resources, which serve as a backup for large-scale model training. These clusters offer enhanced computational power and storage, ensuring scalability and reliability.

No specialized instrumentation is required due to the nature of the project.

2.2.3 Development and Collaboration Tools

Effective collaboration and project management are critical to the success of a multi-member software project. Tools such as GitHub are used to help with communication, documentation, and task tracking. This offers a centralized platform for coding, tracking, and pull request management. It ensures that all team members can contribute to the project while maintaining code integrity and accountability.

Additionally, messaging platforms such as Discord, iMessage, and Outlook are used for real-time communication, including daily updates, troubleshooting, and coordination. Channels are organized by subsystems (AI, Backend, Frontend) to streamline discussions.

2.2.4 Data and Knowledge Resources

The success of the AI model used is heavily dependent on the scale accuracy of the datasets used to train it. Therefore, there are many datasets that will be evaluated that are outlined in further detail in Table 5.

Table 5: Summary of datasets for health misinformation detection

Dataset Name	Description
CoAID [20]	Contains labeled articles and social media posts related to COVID-19, categorized as true or false.
FakeHealth [21]	Offers a broader range of health-related articles with varying credibility. Supports the development of a robust classifier capable of generalizing across medical domains.
Kaggle Datasets [22]	Additional sources for both text and image-based misinformation, including datasets for fake news detection.

2.3 Validation and Testing

Validation and testing are essential for the project's success. They ensure that the system is accurate, stable, usable, and ethically compliant. The team will take a systematic and modular approach to confirm that all results are credible and measurable. GitHub Actions will be the primary tool used to automate critical testing pipelines and enforce immediate feedback on code integrity. The testing methods listed below connect directly to the technical specifications in Section 1.2 System Specification providing the final, measurable evidence of success. Table 6 below links the main requirements to the specific test methods, metrics, target values, and team members responsible.

Table 6: Test accountability matrix

Requirement / Spec	**Test Method**	**Measurement / Metric**	**Target Value**	**Responsible Member(s)**
Classification Accuracy (> 80% F1)	Evaluation on reserved 10% test split	F1-Score (Macro-Averaged)	≥ 80%	Mihran
End-to-End Latency	Automated script run 100x with standardized input	Time-to-Render (TTR) in milliseconds	≤ 3000 ms	Erhowvosere & Ivan
Reliability	Simulating 50 concurrent users over 1 hour	Hypertext Transfer Protocol (HTTP) Status Code 500/400 failure rate	≤ 1%	Erhowvosere
Explainability Layer	Testing pre-flagged misinformation samples	Binary Success (Keywords present and accurately associated with prediction)	100 % Functional Success	Ivan & Mihran

Ethical Disclosure	Testing across target browser/mobile viewports	Binary Success (Disclaimer is persistent and non-dismissible)	100% Functional Success	Ivan

2.3.1 Component-Level Unit Testing

Unit testing validates the isolated functionality of each of the three architectural tiers before integration. Testing begins immediately after the fine-tuning process. The final model is evaluated exclusively on a frozen, reserved test set (10% of the final dataset) to prevent data leakage and ensure its generalization ability. Performance is measured using the F1-Score, which is essential for ensuring a balanced penalty for both false positives and false negatives, a requirement for the high-stakes healthcare domain. All preprocessing functions (e.g., tokenization, cleaning) are unit-tested to ensure data consistency prior to model input.

For Backend API testing, the team uses a specified framework to evaluate the FastAPI and Node.js components. This approach checks that the JSON request and response formats work correctly. It also ensures that all endpoints handle requests and return the right HTTP status codes (200 for success and 400 for bad input).

For Frontend testing, the team evaluates the main user components one by one using a framework like React Testing Library. This checks that the input forms, state management, and visual display logic, such as showing colors for "False" versus "True" results, work properly across different browsers and mobile views, regardless of the live API connection.

Unit testing validates the isolated functionality of each of the three architectural tiers before integration. GitHub Actions workflows will be configured to automatically trigger component-level unit tests upon every code push and pull request to ensure continuous code integrity for integration and deployment.

The model is evaluated on a frozen, reserved test set (e.g., 10% of the final dataset) to prevent data leakage and ensure the model’s generalization ability. Performance is measured using the F1-score. This metric is calculated as a Macro-Averaged F1-Score using Python’s Scikit-learn or

PyTorch's internal metrics to ensure a balanced metric across all three classes ("True," False," "Uncertain"). All preprocessing functions (e.g., tokenization, cleaning) are unit-tested to ensure data consistency prior to model input.

For backend API testing, the team will utilize Pytest and API tools like Postman to evaluate the FastAPI and Node.js components. This approach checks that the JSON request and response formats work correctly. It also verifies the routing logic to ensure all endpoints return the correct HTTP status codes (e.g., 200 for success, 400 for bad input). The GitHub Action pipeline will delay deployment if any unit test fails.

For frontend UI testing, the team evaluates the main user components using a framework like React Testing Library. This checks input validation (e.g., the 5,000-word limit) and verifies visual display logic across target browser viewports, operating independent of the live API connection. The GitHub Action will execute tests to ensure the visual display logic like color-coding for "False" versus "True" results has not unintentionally changed, guaranteeing the stability of the interface across commits.

2.3.2 System-Level Testing

The End-to-End Latency will be measured using a standardized, automated test script executed 100 times. The script will use a consistent, average-length text input and will measure the client-side Time-to-Render (TTR), capturing the entire cycle (e.g., request, model inference, response, UI rendering, etc.). The final metric will be reported using the 95th percentile of the runs to ensure stable, real-world performance is captured under typical peak load, minimizing the influence of network noise outliers. This validates the effectiveness of model optimization techniques like quantization in meeting the 3.0 second target.

Testing for the Clarity Layer validates the ethical safeguard of the system. A small, curated set of input texts is used where the project team has manually pre-validated the specific target keyword (e.g., the harmful phrase or dubious claim). The test ensures that the feature importance logic not only classifies the text correctly but also correctly highlights the known target keyword in the final displayed result. This proves the system is providing useful, traceable evidence to the user, meeting the 100% functional success criterion.

The system's reliability is validated by simulating a realistic load on the production environment. A tool such as Locust or JMeter will be used to simulate 50 concurrent users submitting analysis requests over a continuous period of at least one hour. The test specifically tracks the rate of catastrophic failures (HTTP 500 or 400 error codes). To meet the stringent specification, this failure rate must not exceed 1%, validating the backend's capacity for resource management and its stability under peak load.

2.4 Potential Problems and Mitigation Strategies

Risk management is a key aspect of the team’s project plan and ensures the project execution remains uninterrupted by ay unforeseen technical or logistical roadblocks. The following identifies one currently recognized issue within the project plan, and three potential future risks that may impact the project schedule.

2.4.1 Dataset Quality

Unlike general news, medical data often requires careful validation. The sourcing of high quality and labeled medical datasets is difficult to come across due to existing biases or outdated claims. Choosing a data source with insufficient quality may cause difficulty in generalizing data, as well as achieving F1-score accuracy of 80%. A risk as such could pose a 2–3-week delay in project plans and the general timeline.

To proactively mitigate this risk, the team has reserved time and effort into researching higher quality academic or commercial datasets, should the public ones prove to be inadequate. While the data pipeline is designed to include stages of normalization and data augmentation prior to training, a reserve strategy has been developed in case issues occur with the existing data. If the data produces insufficient F1 scores during Milestone 2, Baseline Model (RoBERTa Fine-Tuning & Validation), the team will shift to a semi-supervised approach with fewer labeled but higher quality datasets. Pairing this up alongside a large volume of unlabeled medical data will help improve the model’s understanding of domain-specific terminology. This should help counter the need for large volume of perfectly labelled data, allowing the team to process with Milestone 3, Backend API (FastAPI Model Wrapper & Latency Test), according to schedule.

2.4.2 Software Bugs/Integration Failure

Integrating three independent codebases creates a high potential for software bugs such as resource usage errors, mismatched API endpoints, and general coding logic errors. These issues are likely to arise during Milestone 8, Integration & System Testing (End-to-End MVP Delivery), and may potentially cause a 1-2 week delay to the final MVP, depending on how difficult the integration or software error is to diagnose.

The mitigation strategy for software bugs and integration issues involves enforcing a strict coding standard at the API endpoint between the Frontend and Backend. Defining exactly which data fields, formats, and types the Frontend/Backend expect to receive from each other ensures that Ivan and Erhowvosere can write their code independently, while aligning with the agreed upon specifications. Additionally, using a TypeScript/JSON Schema validation allows for the verification of data before the code is pushed into the live environment. Rather, these tools flag for errors before major changes are made. A key additional mitigation strategy is staggered integration, which is the process of introducing and testing components gradually rather than all at once. This allows for early detection of issues, smoother integration and reduced overall project risk. Ivan will integrate with a mock API, in Milestone 5, before connecting to the live Python/FastAPI service, in Milestone 6. If major integration issues are identified during Milestone 8, Integration & System Testing (End-to-End MVP Delivery), the team will shift the Frontend/Backend development to a joint pair programming in order to debug and ensure resource contention is resolved.

2.4.2 Team Member Absence

As the current project ownership is divided into three categories (Ivan: Frontend, Erhowvosere: Backend, Mihran: AI), the prolonged absence of a team member can create significant potential issues with meeting project milestones and final showcase deadlines. Specifically, if a team member lead is unavailable during a crucial integration phase, such as API integration with the frontend in Milestone 6, Frontend (API Integration & Dynamic Data Display), the parallel work and Agile sprint structure takes a large setback. This may cause a 2-6-week delay in overall progress.

To help mitigate the loss of a team member for a prolonged period, the team has set up a GitHub repository where all code changes are updated immediately after new additions are added. Alongside daily repository updates, all code changes are supported with detailed documentation, and weekly code walkthroughs are conducted to ensure adequate knowledge transfer between teammates. If team member absence occurs, the recovery strategy consists of immediately redistributing tasks between the remaining two members, and the scope will also be limited to MVP only. This means completely omitting Milestone 9, Optional Extension: Browser Extension Development, and Milestone 10, Optional Extension: Image/Video Detection Prototype.

2.4.3 Loss of GPU Access

The majority of the project relies on the use of high-performance computing services offered by Queen's University (Queen's High-Performance Computing (HPC) Centre and Bain Lab) for any GPU intensive tasks ad fine-tuning of transformers. Should lab closures or network downtime occur, this may lead to potential 1-2-week setback with the team's training timeline, and a direct impact to Milestone 2, Baseline Model (RoBERTa Fine-Tuning & Validation).

To help combat any GPU and model training setbacks, the team has allocated a contingency budget of around $20/month to ensure resources such as Colab Pro are able to be purchased, and GPU intensive processes are able to immediately resume. Should existing lab or network closures on Queen's campus occur, the team will immediately shift and continue to the cloud-based GPU services for the remainder of the project, and notify the faculty regarding the change in budget utilization. This ensures that Milestone 2, Baseline Model (RoBERTa Fine-Tuning & Validation), resumes with no more than a couple-day delay.

2.4.4 Risk Mitigation Summary

Table 7 outlines all risk mitigation strategies as described above.

Table 7: Summary of risk mitigation strategies.

Problem	**Potential Resource Loss**	**Mitigation Strategy**	**Recovery Strategy**
Dataset Quality and Bias	2–3-week schedule delay	Look into semi-supervised learning; Investigate commercial/academic datasets	Switch to BERT model; Reduce dependency on labeled data
Software & Integration Bugs	1–2-week schedule delay	API endpoint validation; Staggered integration (mock API first)	Switch to joint collaborative programming between Frontend/Backend leads (Sere and Ivan)
Team Member Absence	2–6-week schedule delay	Enforce GitHub documentation; Weekly code base walkthroughs	Reduce scope to MVP only; Re-assign tasks to remaining members
Loss of GPU Access	1-2-week schedule delay	Use Google Colab Pro as a pre-paid, immediate hardware contingency; Schedule training off-peak	Colab Pro subscription; Shift training resources to the cloud

3. Milestones/Division of Labor

The project responsibilities are distributed across three functional sections as follows:

- Ivan: Frontend and UI development, user interface design, and API integration.
- Erhowvosere: Documentation, backend inference infrastructure, and system design architecture.
- Mihran: AI pipeline, model training, optimization, and testing.

As indicated in the project schedule below in Table 8, work is broken down into eleven milestones. This structure facilitates the Agile development method by providing short, accountable sprints and enabling parallel work on independent components. The due dates are precisely aligned with the broader date ranges established in the Project Proposal [7].

Table 8: Summary of major milestones, core objectives, and planned timeline.

Milestone	**Due**	**Responsible member(s)**
1. Data Pipeline (Acquisition & Preprocessing)	Oct 26th	Mihran Asadullah
2. Baseline Model (RoBERTa Fine-Tuning & Validation)	Oct 26th	Mihran Asadullah
3. Backend API (FastAPI Model Wrapper & Latency Test)	Nov 16th	Erhowvosere Otubu
4. Backend API (Node.js/Next.js Request Routing & Security)	Nov 23rd	Erhowvosere Otubu
5. Frontend (Core UI Design & Basic Component Build)	Dec 14th	Ivan Samardzic
6. Frontend (API Integration & Dynamic Data Display)	Dec 21st	Ivan Samardzic
7. Testing: Explainability & Clarity Layer Integration	Jan 25th	Mihran Asadullah, Ivan Samardzic
8. Integration & System Testing (End-to-End MVP Delivery)	Jan 25th	Erhowvosere Otubu, Ivan Samardzic
9. Optional Extension: Browser Extension Development	Feb 8th	Ivan Samardzic
10. Optional Extension: Image/Video Detection Prototype	Feb 28th	Mihran Asadullah
11. Final Report Submission & Project Showcase Prep	Feb 28th	All Group Members

4. Budget

As our project is a purely software-based web application, the budget does not include any physical hardware components. The costs are exclusively related to software licensing and the cloud infrastructure required for AI model training and web hosting.

4.1 Software Bill of Materials (BOM)

Our project leverages a modern, open-source software stack. This is a core part of our strategy to keep development costs low. Table 9 outlines all tools required for development, which are available under permissive licenses.

Table 9: Summary of software BOM.

Software	Purpose	License	Cost
Python	Core programming language for AI	Open-source (PSF License) [23]	$0
PyTorch	ML framework for model training	Open-source (Modified BSD) [24]	$0
Hugging Face	Library for pre-trained models (BERT/RoBERTa)	Open-source (Apache 2.0) [25]	$0
Node.js	Server-side runtime	Open-source (MIT License) [26]	$0
React/Next.js	UI framework for web application	Open-source (MIT License) [27]	$0
JavaScript/TypeScript	Core programming language for frontend	Open-source (ECMA) [28]	$0
Chrome/Firefox APIs	SDKs for browser extension development	Open-source	$0
Visual Studio Code	Code editor	Open-source (MIT License) [29]	$0
GitHub	Version control & collaboration	Free (for public/academic repos) [30]	$0
FastAPI	Python framework for serving the AI model as an API	Open-Source (MIT License) [16]	$0
Total			$0

4.2 Infrastructure Budget

While software licensing is free, the main and only costs associated with this project are for cloud computing and hosting. The team have identified free/low-cost options for all project infrastructure needs.

4.2.1 Computing/Cloud Resources

The main computational cost of the project comes from using the BERT and RoBERTa transformer models. The team is not training these models from the ground up as that process is expensive and takes a lot of time. Instead, the team has chosen to fine-tune the models, which just adjusts the final layers of a pre-trained model.

As outlined in Table 10, the main resource for training and testing will be on Google Colab's Pro GPU-enable notebooks, which are ideal for our iterative development process. We have also identified Queen's University's academic clusters as another option for the beginning stages of development.

Table 10: Resources required for the GPU-intensive task of fine-tuning transformer models.

Resource	Provider	Purpose	Estimated Cost
Bain Lab / Queen's HPC	Queen's University	Primary option for model fine-tuning	$0
Google Colab (Free Tier)	Google	Secondary option for model fine-tuning	$0
Google Colab Pro	Google	Contingency for faster GPU access	$15.81 / month

4.2.2 Networking/Hosting Infrastructure

As this project is a software-based web application, networking and hosting requirements are minimal and can be efficiently managed using free or low tier cloud services. The team's infrastructure strategy prioritizes free tier deployment during the development and prototype phases, while maintaining flexibility to scale up to paid plans if necessary.

The hosting architecture consists of two main components, the AI-model backend infrastructure and the frontend web architecture, both of which are hosted on cloud services that support both secure and reliable communication.

4.2.2.1 Networking and Hosting Resources

Table 11 below outlines key infrastructure components for this project, detailing their chosen platforms, their purposes and the associated plans or estimated costs.

Table 11: Summary of hosting resources used

Infrastructure Component	**Provider / Platform**	**Purpose**	**Plan / Estimated Cost**
Model Hosting & API Deployment	Hugging Face Spaces	Host and serve the fine-tuned BERT/RoBERTa model through an interactive UI and API endpoint.	Free Tier, upgradeable to Pro ($9 USD/month) if required for increased uptime or compute.
Backend Service (FastAPI)	Render	Deploy FastAPI backend to handle inference and communication with the frontend.	Free Tier, upgradeable to Starter Plan ($7 USD/month) if higher request throughput is needed.
Frontend Hosting (React / Next.js)	Vercel / GitHub Pages	Host the user interface for interaction with the AI model, with continuous integration and CDN caching.	Free Tier, upgradeable to Pro ($20 USD/month) for enhanced performance and analytics.
Alternative Hosting (Contingency)	Shehala IT Limited	Provides backup hosting for both the web and model services, ensuring availability if public clouds are limited.	Free (via partnership)

4.2.2.2 Networking and Security Considerations

All selected platforms include secure HTTPS protocols and data encryption for both API and web endpoints. The project will use token-based access for API requests, ensuring that model inference endpoints are only accessible to authorized clients. Version control, and deployment automation will be managed through GitHub, which integrates directly with Vercel and Render for seamless updates.

4.2.2.3 Budget Summary

At the current stage of development, the project requires no additional budget allocation for networking or hosting infrastructure. All platforms selected offer free tier plans sufficient for prototyping and early user testing. If scaling becomes necessary (e.g., during demonstration or public release), the total cloud expenditure is expected to remain below $50 CAD per month, covering upgrades to Hugging Face Pro, Render Starter, and Vercel Pro.

5. Risk Mitigation, Environmental Impact, and Legal Considerations

This section outlines the management strategies and safety risks, environmental impact, and legal considerations associated with this capstone project.

5.1 Health and Safety Risks

As the project is a software-based web application, it poses no direct physical health or safety risk to any user. However, there are several ethical and technical risks that must be considered to ensure a reliable and responsible use of the application.

The primary risk associated with this project is indirect and pertains to the interpretation and application of the classification results from the project's AI model. A user might make a critical health decision based on the model's output. The model incorrectly classifying a harmful piece of medical misinformation as "True" could potentially contribute more to the spread of misinformation or encourage a user to unknowingly follow harmful advice. This is the most dangerous risk, as it provides a false sense of security. The real-world consequences of believing such misinformation are severe. For example, during the COVID-19 pandemic, misinformation about the cure led to "800 people dying, 5,876 being hospitalized, and 60 becoming completely blind from drinking methanol" [31]. The model failing to flag a false statement like that would make it complicit in causing harm. On the other hand, the model incorrectly classifying a valid piece of medical advice as "False" could potentially causing a user to mistrust the application.

Additionally, there is the risk of overreliance on the model's output. This is where the user treats the model's classification as an absolute truth rather than a probabilistic assessment, neglecting to do their due diligence and confirming the results on their own. This is a well-documented cognitive phenomenon known as "automation bias" which refers to "our [humans] tendency to favor suggestions from automated decision-making systems and to ignore contradictory information made without automation, even if it is correct" [32]. Marcus Schabacker, MD, president, and CEO of the Emergency Care Research Institute, a leading patient safety organization, warns that tools "claimed to be decision-supporting, very quickly become decision-making. If doctors are pressed for time, you are going to rely on anything that seems reasonable" [33]. With the demand for health care going up as Canada's population has gotten older and sicker, and the average hours doctors work dropping by 20% from 1987 to 2020, a user might not have a chance to speak with a doctor as soon as they would like and take heed to the model's results [34].

Our mitigation strategies are designed to directly counter these risks by upholding the principle of respect for autonomy [35]. Similar to other GenAI chatbots like ChatGPT and Gemini, the web interface of our application will feature a clear and persistent disclaimer stating that the tool is not a substitute for professional medical advice. This disclaimer directly combats automation bias by reminding the user to treat the tool as a support, not a definitive authority. The team will also develop a confidence scoring system with its corresponding classification output. Moreover, we plan to also use the "Clarity Layer" as our primary mitigation strategy against automation bias. By highlighting *why* a text was flagged, we force a shift from passive consumption to active, critical evaluation. The user is given the evidence to verify the results, allowing them to make a truly informed decision.

5.2 Environmental Impact

The environmental impact of this project is minimal but comes from the energy consumption of AI model training and server hosting. The most significant environmental cost in AI is often training models from scratch. A 2019 study estimated that training a single large NLP model from scratch can emit over 626,000 pounds of CO2 equivalent, roughly equal to the lifetime emissions of five cars [36]. Our strategy avoids this because as previously mentioned, we are fine tuning pre-trained transformer models. This fine-tuning process is far more efficient since fine-tuning is much faster, less computationally intensive, and performs quite well with small amounts of data, typically requiring only hundreds or a few thousand labeled samples [37]. This choice will help to minimize our project's carbon footprint.

5.3 Legal Considerations

The team will work hard to address potential legal ramifications by focusing on user liability, data privacy, and intellectual property. The primary legal risk is liability from an incorrect classification (as discussed in 5.1 Health and Safety Risks). For example, if our application classifies an article from a person or organization as "False", they could perceive this as public reputational harm, potentially leading to legal challenge for libel defamation [38]. The way we plan to mitigate this risk from a legal perspective is to require a user to accept a mandatory "Terms of Service" (ToS) agreement before using our application. By accepting the ToS, users formally acknowledge that the tool's classifications are probabilistic, not statements of facts, and are for informational purposes only. This agreement, combined with the confidence score and prominent disclaimers, forms a robust legal defense against claims of libel, as it defines the tool's reputation as an opinion, not an assertion of fact. In terms of protecting the user's input data privacy during a session with the application, the team must comply with Canadian privacy laws, specifically the *Personal Information Protection and Electronic Documents Act* (PIPEDA) [39]. The team will ensure that privacy is protected by not storing or logging any user-submitted text on the project's servers. The user's input will be processed in-memory, used to generate the classification, and then immediately discarded. This ensures that no personal information is ever collected, stored, or at risk of being breached. This policy will also be clearly stated in the ToS. For intellectual property, there is the risk of using software without the correct licenses as it is a violation of copyright law. However, this risk is fully mitigated. As detailed in Section 4.1 Software Bill of Materials (BOM), all software, frameworks, and libraries used in this project are governed by permissive open-source licenses. These licenses explicitly grant the right to use, modify, and deploy software for free [40].

6. Progress to Date

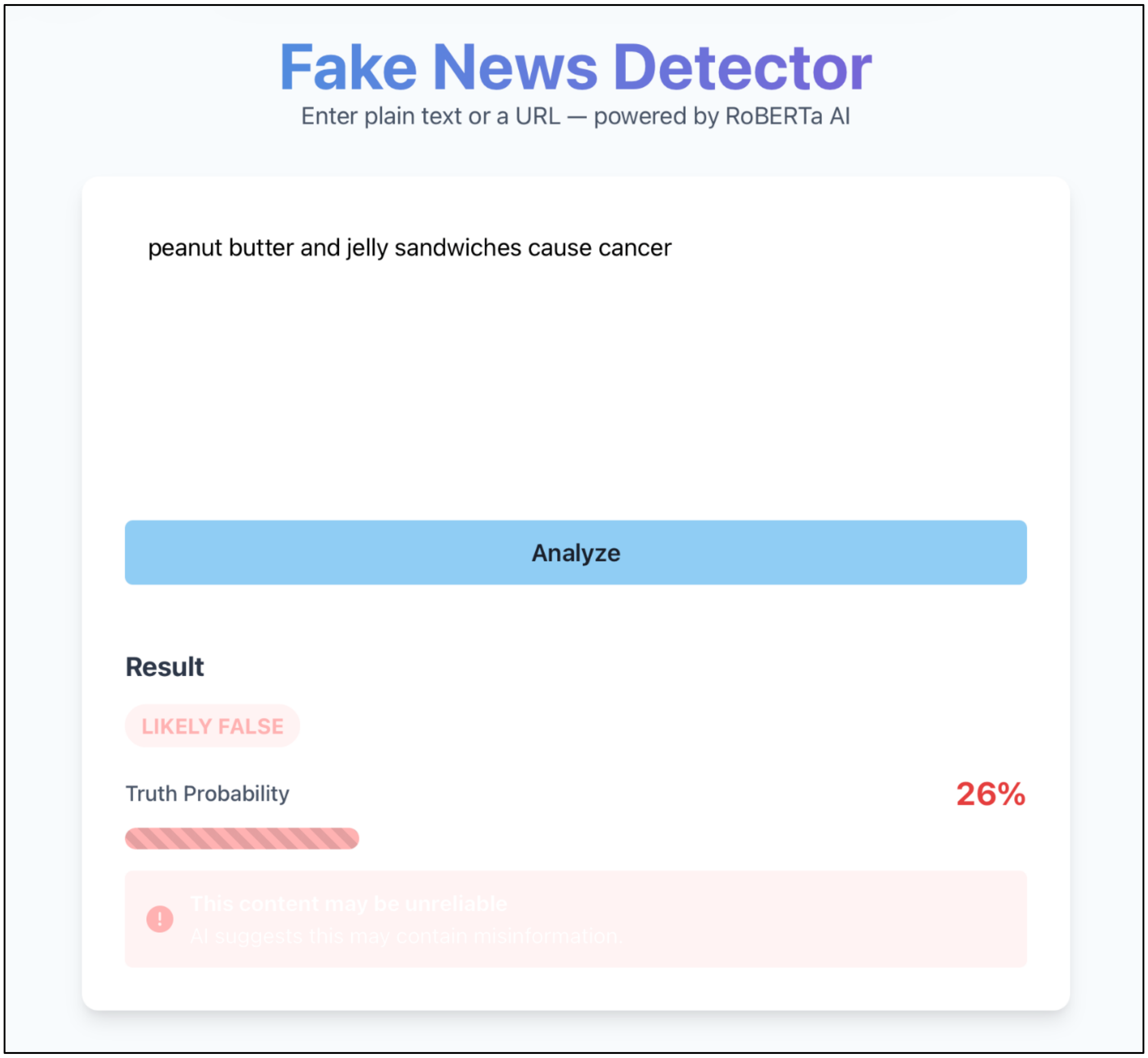
As of October 27th, the team is wrapping up initial development of the AI pipeline and backend setup. The current system features a fine-tuned RoBERTa model that can classify healthcare-related text into “True” or “False” categories. This model is integrated into a Python-based Node.js backend. It takes raw text input through a command-line interface (CLI) and provides classification results with confidence scores.

The team has found medical-care related datasets to do fine-tune training on not just the RoBERTa model, but also the BERT and BioBERT models. After collecting a baseline score for the macro-average F1 score of each model, the team can come to a finalize decision of the model that will be used for the project.

Additionally, the backend infrastructure has been scaffolded using FastAPI, allowing for future expansion into a RESTful API. This sets the foundation for upcoming milestones involving frontend integration and full-stack deployment.

No major delays have been encountered so far. The team remains on schedule, with the next focus being the development of the backend API wrapper and latency testing, followed by frontend component design and integration.

Figure 3 shows a preliminary UI design created by the team using React. This early design illustrates the foundational layout that the team will reference when developing the application. The UI will have a clean and responsive design that allows users to input either raw text or URLs for analysis, as well as a loading bar to show the progress of the model’s classification output percentage, alongside its “True”, “False” or “Uncertain” label.


Fake News Detector

Enter plain text or a URL — powered by RoBERTa AI

peanut butter and jelly sandwiches cause cancer

Analyze

Result

LIKELY FALSE

Truth Probability 26%

This content may be unreliable

AI suggests this may contain misinformation.

Figure 3: Preliminary UI Design using React

7. Conclusion

In summary, the blueprint highlights the target vision of the project thus far and outlines a detailed engineering plan for carrying out the remainder of the project. The team is currently on track with their milestone goals, specifically developing the preliminary prototype for validating the core logic, and training the transformers on existing open-source medical datasets. The frontend/backend development are set to commence within the next few weeks, as dictated in the project milestones, while the MVP and final testing is set to be complete by the end of January 2026. While early research has shown that dataset quality may pose timeline risk, the team has developed mitigation strategies to shift to semi-supervised learning in order to achieve an F1 classification score greater than 80%. In conclusion, the Blueprint provides a clear roadmap for delivering a functional, ethical, and high-performance misinformation detection system.

References

[1] J. Ashby, "The Effects of Medical Misinformation on the American Public," Ballard Brief, March 2024. [Online]. Available: https://ballardbrief.byu.edu/issue-briefs/the-effects-of-medical-misinformation-on-the-american-public. [Accessed 26 October 2025].

[2] S. Orbanek, "Study shows verified users are among biggest culprits when it comes to sharing fake news," Temple Now, 9 November 2021. [Online]. Available: https://news.temple.edu/news/2021-11-09/study-shows-verified-users-are-among-biggest-culprits-when-it-comes-sharing-fake. [Accessed 26 October 2025].

[3] R. Kundu, "F1 Score in Machine Learning: Intro & Calculation," V7 Labs, 16 December 2022. [Online]. Available: https://www.v7labs.com/blog/f1-score-guide. [Accessed 26 October 2025].

[4] N. Buhl, "F1 Score in Machine Learning," Encord, 18 July 2023. [Online]. Available: https://encord.com/blog/f1-score-in-machine-learning/. [Accessed 26 October 2025].

[5] L. Nikolov, "What's the difference between API Latency and API Response Time?," Sentry, 11 October 2024. [Online]. Available: https://blog.sentry.io/whats-the-difference-between-api-latency-and-api-response-time/. [Accessed 26 October 2025].

[6] App Signal, "What are good and acceptable error rates?," App Signal, [Online]. Available: https://www.appsignal.com/learning-center/what-are-good-and-acceptable-error-rates. [Accessed 26 October 2025].

[7] I. Samardzic, E. Otubu and M. Asadullah, "ELEC 490 / 498 Proposal - AI for Fake News and Deepfake Detection," Group 16, Kingston, 2025.

[8] V. Buglaev, "Multi Tier Architecture," Medium, 19 September 2024. [Online]. Available: https://medium.com/software-architecture-patterns-and-styles/software-architecture-patterns-and-styles-part-1-multi-tier-architecture-bc5064ea6329. [Accessed 24 October 2025].

[9] Wikipedia, "Multitier architecture," [Online]. Available: https://en.wikipedia.org/wiki/Multitier_architecture. [Accessed 204 October 2025].

[10] "Manifesto for Agile Software Development," Manifesto for Ahile Development, 2001. [Online]. Available: https://agilemanifesto.org/. [Accessed 26 October 2025].

[11] "Python," Python, [Online]. Available: https://www.python.org/. [Accessed 26 October 2025].

[12] "Visual Studio Code," Microsoft, [Online]. Available: https://visualstudio.microsoft.com/. [Accessed 26 October 2025].

[13] "Google Colaboratory," Google Colab, [Online]. Available: https://colab.google/. [Accessed 26 October 2026].

[14] "Get Started," PyTorch, [Online]. Available: https://pytorch.org/. [Accessed 26 October 2025].

[15] "Transformers," Hugging Face, [Online]. Available: https://huggingface.co/docs/transformers/index. [Accessed 26 October 2025].

[16] "FastAPI," FastAPI, [Online]. Available: https://fastapi.tiangolo.com/. [Accessed 26 October 2025].

[17] "Node.js v25.0.0 documentation," Node.js, [Online]. Available: https://nodejs.org/docs/latest/api/. [Accessed 26 October 2025].

[18] "React," React, [Online]. Available: https://react.dev/. [Accessed 26 October 2025].

[19] "Next.js Docs," Next.js, [Online]. Available: https://nextjs.org/docs. [Accessed 26 October 2025].

[20] L. Cui and D. Lee, "CoAID: COVID-19 Healthcare Misinformation Dataset," The Pennsylvania State University, 3 November 2020. [Online]. Available: https://arxiv.org/abs/2006.00885. [Accessed 26 October 2025].

[21] E. Dai, S. Yiwei and S. Wang, "Ginger Cannot Cure Cancer: Battling Fake Health News with a Comprehensive Data Repository," 2020. [Online]. Available: https://github.com/EnyanDai/FakeHealth. [Accessed 26 October 2025].

[22] "Datasets," Kaggle, [Online]. Available: https://www.kaggle.com/datasets. [Accessed 26 October 2025].

[23] Python documentation, "History and License - Python 3.14.0 documentation," [Online]. Available: https://docs.python.org/3/license.html. [Accessed 26 October 2025].

[24] PyTorch, "License - FBGEMM 1.4.0 documentation," [Online]. Available: https://docs.pytorch.org/FBGEMM/general/License.html. [Accessed 26 October 2025].

[25] Hugging Face, "Licenses," [Online]. Available: https://huggingface.co/docs/hub/en/repositories-licenses. [Accessed 26 October 2025].

[26] L. Tal, "Node.js licensing and security considerations," Snyk, [Online]. Available: https://snyk.io/articles/node-js-licensing-and-security-risks/. [Accessed 26 October 2025].

[27] "React License (MIT)," Meta Platforms, Inc., [Online]. Available: https://github.com/facebook/react/blob/main/LICENSE. [Accessed 26 October 2025].

[28] "Ecma International policy on submission, inclusion and licensing of software," ECMA International, [Online]. Available: https://ecma-international.org/policies/by-ipr/ecma-international-policy-on-submission-inclusion-and-licensing-of-software/. [Accessed 26 October 2025].

[29] "Microsoft Software License Terms - Microsoft Visual Studio Code," Visual Studio Code, [Online]. Available: https://code.visualstudio.com/license. [Accessed 26 October 2025].

[30] "Licensing a repository," GitHub Docs, [Online]. Available: https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/licensing-a-repository. [Accessed 26 October 2025].

[31] J. Ashby, "The Effects of Medical Misinformation on the American Public," Ballard Brief, March 2024. [Online]. Available: https://ballardbrief.byu.edu/issue-briefs/the-effects-of-medical-misinformation-on-the-american-public. [Accessed 23 October 2025].

[32] B. Hoffman, "Automation Bias: What It Is And How To Overcome It," Forbes, 10 March 2024. [Online]. Available: https://www.forbes.com/sites/brycehoffman/2024/03/10/automation-bias-what-it-is-and-how-to-overcome-it/. [Accessed 23 October 2025].

[33] R. Southwick, "Why AI remains a top concern for patient safety," Chief Healthcare Executive, 12 March 2025. [Online]. Available: https://www.chiefhealthcareexecutive.com/view/why-ai-remains-a-top-concern-for-patient-safety. [Accessed 23 October 2025].

[34] "How many doctors are there in Canada?," Canadian Medical Association, [Online]. Available: https://www.cma.ca/healthcare-for-real/how-many-doctors-are-there-canada. [Accessed 23 October 2025].

[35] A. A. Abujaber and A. J. Nashwan, "Ethical framework for artificial intelligence in healthcare research: A path to integrity," PubMed Central, 20 September 2024. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11230076/. [Accessed 23 October 2025].

[36] E. Strubell, A. Ganesh and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," 5 June 2019. [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://arxiv.org/pdf/1906.02243. [Accessed 23 October 2025].

[37] CatarinaG, "Fine-tuning vs. Training from Scratch: Deciding the Best Approach for Model Development," Medium, 27 February 2024. [Online]. Available: https://medium.com/@caterine/fine-tuning-vs-training-from-scratch-deciding-the-best-approach-for-model-development-f82c73e168a5. [Accessed 24 October 2025].

[38] Government of Ontario, "Libel and Slander Act, R.S.O. 1990, c. L.12," [Online]. Available: https://www.ontario.ca/laws/statute/90l12. [Accessed 24 October 2025].

[39] Office of the Privacy Commissioner of Canada, "The Personal Information Protection and Electronic Documents Act (PIPEDA)," [Online]. Available: https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/. [Accessed 24 October 2025].

[40] Open Source Initiative, "The Open Source Definition," 16 February 2024. [Online]. Available: https://opensource.org/osd. [Accessed 24 October 2025].