**Reflective Report – TruthLens Project**

TruthLens – An AI-Powered Misinformation Detection Tool

Students:  
Aya Higaze – 211910872.   
Khadijah Abed el Ghani – 212060073.

**1. Introduction**

In today's digital era, information spreads at lightning speed. Social media platforms like Twitter, Facebook, and TikTok are full of short statements and viral headlines. While some of these are factual, many are either misleading, taken out of context, or completely false. Unfortunately, not everyone has the tools or time to verify what they read. This leads to the rapid spread of misinformation and biased narratives.

Our project, TruthLens, aims to help solve this problem. It’s a simple but smart tool that allows users to input any short-form content – such as a tweet or a headline – and receive an AI-powered analysis. The tool detects possible logical fallacies, cognitive biases, or signs of misinformation in the input. It also explains why the input may be problematic and suggests sources where the user can read more or verify the claim.

We wanted to build something practical, easy to use, and educational – a tool that not only warns users but also helps them think critically about the information they consume. This aligns with the goals of the course, which emphasizes applying theory to real-world challenges.

**2. Architecture & Design Decisions**

We decided early on to keep the architecture clean and easy to understand. The project follows a client-server model. This separation helped us work independently – Aya focused on the backend, and Khadijah handled the frontend.

Backend (Server-Side)

The backend is built using Spring Boot, a Java framework known for simplicity and power. We created a REST API that accepts a piece of text, sends it to the Google Gemini AI for analysis, and returns a well-structured response to the frontend.

Aya used Jackson for parsing the JSON responses and Lombok to simplify Java model creation. The backend also includes error handling to deal with failed API calls or bad responses.

Frontend (Client-Side)

The frontend is written in HTML, CSS, and vanilla JavaScript. We chose not to use a complex framework like React or Angular to keep the project simple and accessible. The interface is straightforward: it includes a textarea where users can paste content, buttons to analyze or clear, and a results section that updates dynamically.

One key UI feature is the ability to toggle between a clean explanation and the raw JSON response from the AI. This adds transparency and encourages users to understand how AI responses are structured.

**3. Learning Journey – What We Gained**

Khadijah (Frontend & Documentation)

Before this project, I had some experience with HTML and CSS, but I wasn’t confident in JavaScript. I had never worked with asynchronous APIs or dynamic UI updates before. Through this project, I learned how to:

Use fetch() to send requests and handle asynchronous responses.

Parse and display JSON data in the browser.

Handle UI states like loading, success, and error messages.

Design a user interface that is accessible, clear, and responsive.

I also gained experience writing technical documentation. I helped prepare this reflective report, wrote comments in the code, and ensured the README was clear for future users.

Resources I used:

MDN Web Docs (JavaScript basics, fetch API, DOM manipulation)

YouTube tutorials for async JS and form validation

ChatGPT to troubleshoot small bugs or formatting questions

Aya (Backend & AI Integration)

I already had solid experience in Java and Spring Boot, but this project gave me new challenges: integrating AI, handling unpredictable outputs, and crafting prompts. I learned how to:

Send HTTP POST requests to Gemini’s API with the right headers and body.

Format prompts to get structured JSON replies from the model.

Clean up and extract only the needed parts from a complex AI response.

Handle exceptions gracefully when the model responds incorrectly.

Resources I used:

Google Gemini API documentation

Postman (for testing REST endpoints)

Spring Boot tutorials for beginners and advanced users

Blog articles on prompt engineering

Together, we helped each other learn new skills and reviewed each other’s work often.

**4. AI Prompts – How We Talked to Gemini**

The AI doesn’t just magically know what we want. It needs a clear prompt, which is the instruction we send it. We crafted a prompt that asked Gemini to analyze a claim and return the results in a strict JSON format.

Here’s the actual prompt:

String prompt = "Claim: \"" + inputText + "\"\n\n" +

"Analyze the claim for:\n" +

"1. Logical fallacies (if any)\n" +

"2. Cognitive biases (if any)\n" +

"3. A scientific explanation to address or correct the claim, if needed.\n\n" +

"If the claim is purely subjective (such as an opinion or personal statement), and does not contain misinformation, logical fallacies, or cognitive biases, respond with an empty flags list and a simple explanation saying it's a personal opinion or neutral statement.\n\n" +

"Return the `sources` field as a list of plain URLs in string format only.\n\n" +

"Respond STRICTLY with a valid JSON object. Do NOT use Markdown formatting (no ```), and do NOT include any explanation or surrounding text. JSON only.\n\n" +

"Required format:\n" +

"{\n" +

" \"flags\": [\"...\"],\n" +

" \"explanation\": \"...\",\n" +

" \"sources\": [\"...\"]\n" +

"}";

This prompt helped reduce hallucinations, markdown formatting, and irrelevant text. It still wasn’t perfect – sometimes the AI sent markdown (```json), or included non-JSON content. To fix this, Aya wrote a method to extract only the valid JSON section of the reply.

**5. AI Strengths & Weaknesses**

Gemini AI did a great job identifying issues like:

Appeals to emotion

Confirmation bias

Slippery slope arguments

Over-generalizations

But it also had limits:

Sometimes Gemini responded with vague sources like “Wikipedia” instead of giving actual links.

It failed to understand humor or sarcasm – for example, “Aliens landed in Canada” was treated seriously.

It occasionally flagged neutral or personal opinions as problematic.

Markdown formatting broke our JSON parsing.

We handled these by improving prompts, adding fallback responses, and offering raw data viewing for transparency.

**6. Challenges and Solutions**

|  |  |
| --- | --- |
| Challenge | What We Did |
| AI returned markdown or invalid JSON | Wrote custom extraction and cleaning logic |
| Frontend errors or UI freezing | Added error states, loading indicators, and clear resets |
| Claim phrasing changed results | Tested multiple input phrasings and adjusted prompt |
| Gemini’s vague sources | Added extra search links via SimpleFactCheckService |
| Confusion about sarcastic posts | Added notes to users about AI limitations |

One interesting discovery was how the same sentence with a small change could lead to a totally different analysis. For example, “Coffee cures cancer” vs. “Some believe coffee helps with cancer” triggered completely different outputs. This showed us how sensitive models are to wording.

**7. Biases and Fallacies We Noticed**

While developing and testing TruthLens, we didn’t just detect biases in online content — we also noticed how our own thinking was affected by various cognitive biases. This realization was an unexpected but valuable learning experience. Here are the main biases we encountered during our work, with explanations and examples:

🔸 Confirmation Bias

This bias happens when we search for or interpret information in a way that confirms our pre-existing beliefs, while ignoring or downplaying opposing views. In our case, we often selected example tweets or headlines that we personally agreed with or believed to be false, expecting the AI to confirm our assumptions. For example, when testing a claim like "Vaccines are dangerous," we already assumed it's misinformation and were satisfied when the AI flagged it. But we didn’t always test claims we disagreed with or were unsure about.

This behavior limited our tool’s testing diversity. We learned that to evaluate the AI fairly, we needed to input a wider variety of claims – some that challenge our beliefs – to see how consistent and balanced the model really is.

🔸 Automation Bias

Automation bias occurs when people over-rely on automated systems, assuming they are always correct. At first, we trusted Gemini’s answers without much skepticism. If the model said a statement had no fallacies, we accepted that, even if we felt something was missing. Or if it flagged a neutral opinion, we assumed it had to be right.

We realized that AI, while impressive, is not perfect. It can misunderstand context, tone, or cultural nuance. This taught us to treat AI as a helpful assistant, not a final authority. We began double-checking responses and questioning unexpected results.

🔸 Framing Effect

The framing effect is when how something is worded changes how people (or AI) interpret it. We observed this directly with Gemini. For example, changing "Coffee cures cancer" to "Some people believe coffee may help prevent cancer" led to completely different results — one was flagged as misinformation, the other as neutral.

This showed us that word choice and phrasing matter a lot in how the model analyzes content. It also reminded us how easily headlines can influence readers simply by framing information in a more extreme or emotional way.

**8. Ethics & Responsibility**

Labeling content as “misinformation” is serious. It can impact reputations, opinions, and even political decisions. So we were very careful:

We used the word “flags” rather than “true/false”

We always provided source links

We allowed users to see the raw JSON to decide for themselves

We clarified that our tool is a guide, not a judge

Ethical tech use is critical, especially in AI. We wanted to build something that promotes thinking, not controlling.

**9. Future Improvements**

Here are features we would love to add:

Multilingual support – Arabic, Hebrew, and more

Support for images and videos – e.g., analyze memes or news clips

Advanced models – like GPT-4 or Claude with improved factual grounding

User voting system – to rate the AI’s accuracy

Data export or history – for journalists or researchers

Dark mode and mobile responsiveness

We believe TruthLens has great potential for classrooms, media literacy programs, and even browser extensions.

**10. What Each of Us Did**

|  |  |  |
| --- | --- | --- |
| Task | Aya | Khadijah |
| Backend setup | ✅ | ❌ |
| AI integration & prompt crafting | ✅ | ✅ |
| JSON parsing and validation | ✅ | ❌ |
| Frontend development | ❌ | ✅ |
| Error handling (UI/UX) | ❌ | ✅ |
| Writing the reflective report | ✅ | ✅ |
| Ethics and bias discussion | ✅ | ✅ |
| Testing with real-world claims | ✅ | ✅ |

We worked side-by-side online, shared code on GitHub, discussed bugs over WhatsApp, and regularly met to review progress. Our collaboration was balanced and respectful, and we learned from each other.

**11. Conclusion**

This project was more than just a programming task. It was a chance to take the concepts of bias, truth, and digital responsibility and build something real from them. We both grew as developers and thinkers. We faced technical issues, ethical questions, and communication challenges — and we handled them all.

TruthLens is a simple app, but behind it is a big message: don’t trust everything you read online, and don’t trust AI blindly either.

We hope our tool helps others learn, reflect, and make smarter choices in the digital world. And we’re proud of what we’ve created together.

**Attachments**

GitHub Repo: [https://github.com/AyaRaedHigaze/misinfo-detector]

YouTube Demo: [https://youtu.be/lb9cKWG6OyY]