Sri Lanka Institute of Information Technology

Software Stack Specifications

Information Security Project 2025 Y3.S2.WE.CS

Group ID: ISP-33

Topic: Browser Extension for Preventing Sensitive Data Leaks

Submitted by:

Student	Student 1	Student 2
Name	Dilshara N.P.G.R	Dulanjana E.J.A.T
IT Number	IT22064790	IT22186874

Table of Contents

1. Executive Summery.

This project proposes the creation of a browser extension designed to avert sensitive data leaks, and to address the increasing risk of sharing information that we did not mean to share when filling in web forms and using interfaces that are driven by AI. With new online services and Large Language Models (LLMs) emerging, users often type or paste confidential information into browsers, including login credentials, financial information, or corporate identifiers. These actions bypass traditional security protections like firewalls or endpoint protection**.**

The solution is designed as a lightweight, privacy-preserving browser extension built with JavaScript, HTML, and CSS, operating entirely on the client side. At its core, the software stack integrates:

- Regex and keyword-based detection for structured and unstructured sensitive data.
- Future-ready machine learning modules for context-aware analysis of inputs.
- A user-friendly alert system that provides real-time pop-ups, allowing users to cancel, edit, or proceed with submissions.
- Privacy by design architecture, ensuring no user data leaves the local device, maintaining compliance with GDPR and HIPAA.

Utilizing an **Agile** development approach, the project embraces an iterative sprint approach to enhance detection accuracy, reduce false positive raids, and optimize performance. The extension will initially be released on the Chrome Web Store but will be scalable in the future through ongoing user feedback and new releases.

This initiative can offer considerable security and commercial value; it helps users avoid accidental disclosures, can assist organizations with compliance, and bolsters confidence in AI-enabled services. Based on monitoring the user's behavior and server-side defenses, the proposed software stack promotes secure interactions online and towards sustainable cybersecurity.

2. Introduction.

2.1 Purpose of Implementing This Project?

There are more individuals than ever relying on internet services and AI applications. In the process, many tend to paste or type personal data, bank information or company data into chat windows and forms without regard for the risk. Modern security software including endpoint security and firewalls only come into play after data is carried from the browser. The most common place where leakage happens is the browser itself is left exposed. It is accomplished to fill that gap by giving users an easy medium which protects their sensitive data before releasing it out.

2.2 Intended Audience.

End users / Employees

- Benefit: Fewer accidental leaks (passwords, bank numbers, API keys).
- Why it matters: Reduces embarrassment, prevents personal liability, and preserves employee productivity.

Security Operations / SOC / Incident Response

- Benefit: Fewer preventable incidents to chase; lower noise from avoidable alerts.
- Why it matters: Frees SOC to focus on real threats and reduces mean time to respond to true incidents.

IT Governance / Risk Management

- Benefit: Proactive control at the point of entry into systems that reduces residual risk.
- Why it matters: Strengthens control of the environment and makes risk registers more manageable.

Compliance / Legal / Data Protection Officer (DPO)

- Benefit: Helps demonstrate “reasonable” technical measures to protect PII (evidence for audits / regulators).
- Why it matters: Supports GDPR/HIPAA/PCI obligations and reduces regulatory / litigation exposure.

CISO / Security Leadership & Board

- Benefit: Measurable reduction in accidental data-exfiltration risk and stronger governance metrics to present to the board.
- Why it matters: Improves security posture and supports strategic risk reporting.

Customer Support / Sales

- Benefit: Prevents accidental disclosure of customer data into external chatbots or web tools.
- Why it matters: Maintains customer trust and prevents reputational loss.

Developers & Product Teams

- Benefit: Protects secrets accidentally pasted into chat windows, logs, or issue trackers.
- Why it matters: Prevents leakage of API keys/credentials that can lead to production incidents or supply-chain problems.

2.3 Reading Suggestions.

To receive a succinct summary, you could first read EXEC_SUMMARY for a succinct overview and business value. Technical reviewers and implementers should read TECH_DOC_ExtensionArchitecture.md for details about the architecture for the system, detection rules and criteria, runtime detail, and deployment procedures. If you are a researcher or evaluator, and you'd like more details about model architecture, the datasets, experiments, and evaluation metrics, etc, then you should read SOFTWARE_STACK pdf.

3. Overall Description.

3.1 Product Perspective.

This browser extension is a lightweight, privacy-first data loss prevention (DLP) product that sits at the user's entry point—complementing existing endpoint and network controls, not replacing them—by integrating lightweight, rule-driven filters with an in-browser ML model to help prevent accidental or adversarial disclosures of PII (personally identifiable information), credentials, and corporate secrets. Designed for Chromium-based browsers, the app is built for client-side inference and nudges users immediately with context-aware warnings about any potential privacy risks while being GDPR/HIPAA compliant (no raw inputs leave the device.

The extension can scale from single-user installs to enterprise organization-wide deployments, complete with optional admin console to manage rollout of corporate policy, telemetry, and SIEM/GRC integration. From a market and governance perspective, the tool targets regulated industries (finance, healthcare, legal), software services with a lot of developers, and managed service providers (MSP) looking for low-friction controls that will reduce SOC noise without introducing too much compliance risk and provide an auditable safeguard. This extension has the potential to be both a practical security control and a strong commercial product.

3.2 Product Features.

The extension combines fast, rules-based detection (regex + keyword heuristics) with a compact in-browser ML model for context-aware PII and credential detection, where all inference happens on the client-side to preserve privacy; it includes real-time input monitoring, unobtrusive popup actions (Cancel / Edit / Proceed), configurable allow-lists and policy profiles, encrypted local settings, and opt-in anonymized telemetry to help tune.

The Enterprise capabilities include centralized policy rollout (via admin console / GPO / MDM), SIEM/GRC integration hooks, signed updates for model and extension artifacts, and telemetry-driven reporting (adoption, flagged submissions, false-positive rates). The product is designed for low latency and cross-Chromium compatibility (Manifest V3 / CSP) and supports model updates (quantized / optimized artifacts) and has testing / fallbacks for guaranteed reliability and minimal disruption to users.

3.3 Operating Environment (Hardware Platform).

Development

Purpose: write extension code, run local builds, debug, and do small model experiments.

- **Minimum (dev laptop):** 4 cores (Intel i5 / Ryzen 5), 16 GB RAM, 512 GB NVMe SSD, modern browser (Chrome/Chromium).
- **Recommended:** 6–8 cores, 32 GB RAM, 1 TB NVMe, dual-monitor setup.
- Notes: macOS, Windows, or Linux are fine — keep one machine per OS if you need cross-platform testing.

Training / Model development hardware

Purpose: train/fine-tune ML models, run experiments, hyperparameter tuning.

- **Small experiments / lightweight models:** single consumer GPU (e.g., NVIDIA RTX 3060 / 3070 / 4060) with 8–12 GB VRAM, 32–64 GB system RAM, NVMe SSD (1 TB+).
- **Medium/serious training (recommended):** NVIDIA RTX 3080/4080 or A10/A30 class with 16–24 GB VRAM, 64 GB+ RAM, 2 TB NVMe.
- **Large models / production training:** data-center GPUs (NVIDIA A100 / V100 / H100) or multi-GPU nodes — usually via cloud (AWS G4/G5/P3/P4, GCP A2, Azure NC/P series).
- **Alternative / low-cost:** Google Colab Pro / Kaggle / Hugging Face / rented cloud GPU instances for bursts.

3.4 Operating System and Versions.

End-user (inference / extension runtime) supported client OS

- **Windows 10 / Windows 11 (64-bit)** mainstream corporate desktops.
- **macOS 11 Big Sur or later** (prefer 12+).
- **Ubuntu 20.04+ or other modern desktop Linux (x86_64)** for developer/technical users.
- **Chrome OS (Stable channel, recent versions)** Chromium-based Chromebooks that support extensions.

Browsers (client compatibility note)

- **Chromium-based browsers (Chrome, Edge, Brave, Opera)** use recent stable releases; target Manifest V3 compatibility supported by recent Chromium builds.
- **Safari / Firefox:** Firefox supports extensions, but runtime differences exist (Web Extension API); Safari support for Chrome-style extensions is limited — treat as secondary.
  Important primary target is Chromium-family (Chrome Web Store).

3.5 Software Components and Applications.

Browser Extension (client-side bundle)

- **Purpose:** Real-time input monitoring, detection orchestration, UI popups, local model inference.

- **Subcomponents:**
    - **Content scripts** — inject into pages, capture form/chat input events.
    - **Service worker / background script** — central decision logic, policy checks, update handling (Manifest V3).
    - **Popup UI / options page** — show alerts, user choices (Cancel/Edit/Proceed), preferences.
    - **Local storage module** — encrypted storage of user prefs / allowlists.
- **Suggested stack:** JavaScript (ES6+), Manifest V3, HTML/CSS, Web Extension APIs, secure storage (chrome. Storage with local encryption lib).

Detection Engine (client-side)

- **Purpose:** Combine deterministic rules and ML inference into a single decision.
- **Subcomponents:**
    - **Regex & rules module** — high-confidence structured patterns (credit cards, emails, API keys).
    - **Keyword / heuristic module** — lexicon and context heuristics.
    - **ML inference wrapper** — load & run quantized model in browser (TF.js / ONNX Runtime Web / WASM).
    - **Decision fusion & confidence module** — thresholding, risk scoring, and fallback logic.
- **Suggested stack:** TF.js or ONNX Runtime Web for in-browser inference; plain JS modules for regex/heuristics.

ML Model Training Pipeline (offline)

- **Purpose:** Build/train/validate the context-aware model used by the extension.
- **Components & tools:**
    - **Data preparation scripts** — Python (pandas), labeling helpers, augmentation.
    - **Model code & training scripts** — PyTorch or TensorFlow (training/validation), tokenizers.
    - **Evaluation scripts** — produce precision/recall/F1, confusion matrices.
    - **Export & optimization** — quantization, pruning, convert to TF.js / ONNX format.

- **Suggested stack:** Python 3.10+, PyTorch or TensorFlow, Hugging Face Transformers (if using small transformers), ONNX / TensorFlow Converter, quantization tools.

3.6 Interoperability Considerations

- **Cross-Browser Compatibility:** The extension has been built for Chromium-based browsers, such as Chrome, Edge, Brave, Opera, but should be run through testing for Firefox and Safari for potential increased exposure in organizations with mixed browser environments.
- **Operating System Support**: Since it runs within the browser, the extension will work across Windows, macOS, and Linux desktops. Mobile browser interoperability can be considered in future versions.
- **Data privacy regulations**: created on the basis of "privacy by design," such that no user data is shared outside of the browser, and are aligned with the GDPR, HIPAA and organizational policies while also avoiding interoperability compliance with regulatory restrictions.
- **Updating and maintenance:** automatic browser store updates should support an enterprise patch cycle without disruption to existing security monitoring tools.
- **Machine learning models:** optimized Machine Learning models need to work in a lightweight in-browser inference framework (e.g. TensorFlow.js or ONNX.js) that can infer in any browser without requiring external compute.

3.7 Design and Implementation Constraints.

3.7.1 Hardware Limitations.

- **Processing Power:** Given that the extension runs on end-user machines, it needs to run efficiently using the limited CPU resources on standard desktops and laptops so there is no obvious performance lag in the browser.
- **Memory Usage:** The computational load of the in-browser ML model needs to be small and plug in (quantization/pruning) so that it doesn't exceed even typical RAM availability or adversely affect other browser tabs.
- **Device Variability:** Users may be using a wide variety of hardware (from low-end corporate laptops to full-fledged development workstations), therefore, the solution needs to be able to scale appropriately without reducing the user experience.

3.7.2 Interfaces or Other Applications.

- **Compatibility with Existing Security Infrastructure**: The extension should easily blend into existing organizations' IT security ecosystems, such as SIEM, DLP, and IT governance dashboards. Log export (optional) and APIs are acceptable options for enabling this integration.
- **Cross-Browser Interfaces**: The primary target is for the extension to be used in Chromium-based browsers, however, future consideration for interoperability with Firefox and Safari should be considered for consistent user experience.
- **Enterprise Deployment Tools**: The extension should be easily deployed using enterprise distribution systems, such as Group Policy Objects, Microsoft Intune, or Chrome Enterprise, for organizations looking to roll it out to teams working in their environments.

3.7.3 Specific Technologies, Tools, and Databases.

- **ML Framework Compatibility:** The chosen ML framework (TensorFlow.js or ONNX Runtime Web) must support in-browser inference without requiring external servers.
- **Development Tools:** Node.js, JavaScript/ES6+, HTML, and CSS must be compatible with the Chrome Extensions Manifest V3 standard.

3.7.4 Language Requirements.

- **Programming Language Support**: This covers the necessary JavaScript (for extension logic and UI) HTML/CSS (for interface design) and Python (for model training/export). It may also be useful to be compatible with WebAssembly runtimes as an optimization for ML inference.
- **Development Tools Compatibility**: The build process must be compatible with current development stacks (Node.js LTS, npm, ESLint, Webpack/Vite, etc.) and produce stable browser packages.

3.7.5 Protocols.

- **Browser Security Policies**: Must comply with Chrome's Content Security Policy (CSP) and permissions model to prevent injection risks.
- **Update Mechanism**: Extension and ML model updates should be digitally signed and delivered securely through browser marketplaces (e.g., Chrome Web Store).

3.7.8 Design Standards or Programming Guidelines.

- **Code Maintainability**: Code standards (e.g. module design patterns, naming conventions, ESLint rules) will foster maintainable code and simplify future developer onboarding.
- **Privacy by Design**: Any analysis of sensitive input occurs on-device; no raw PI goes off-device. These are core principles of GDPR and HIPAA.
- **User Experience Standards**: Alerts will be unobtrusive and intuitive to reduce alert fatigue while conveying meaningfully conveying risk.

3.7.9 Architecture of Extensions.

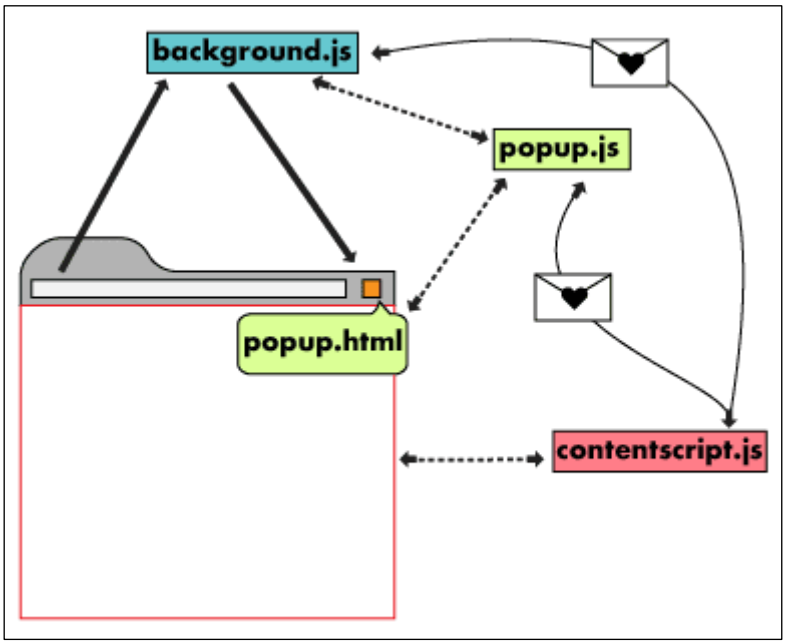


Figure 1 - Extensions Architecture

3.7.10 Browser extension privacy detection framework.

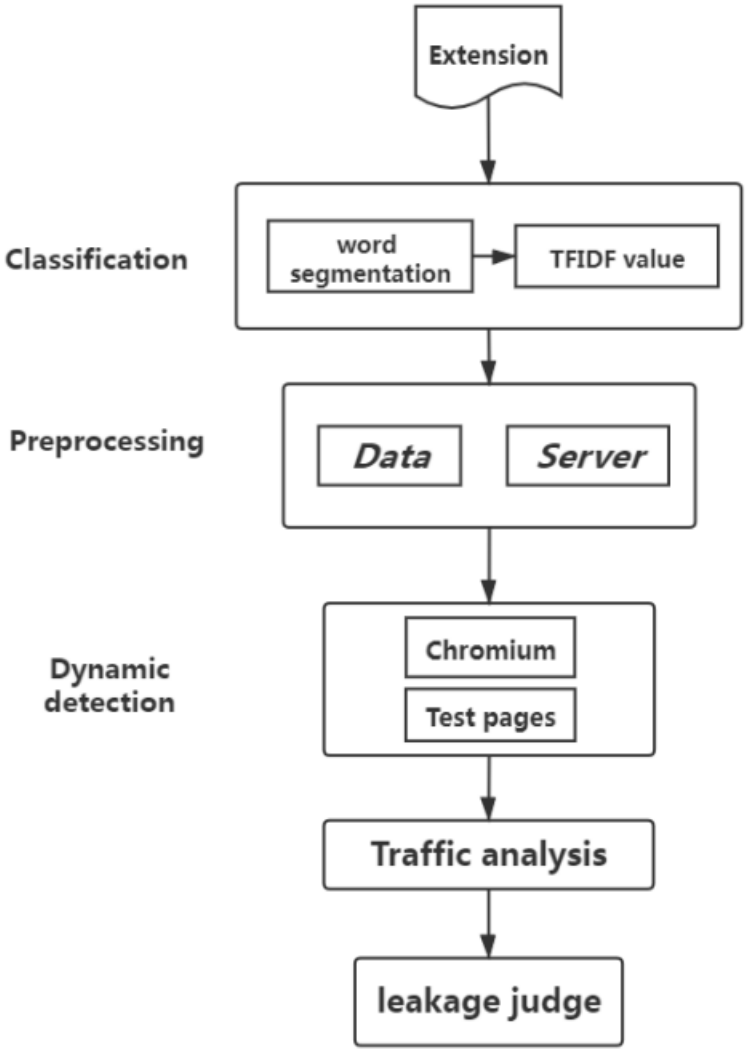


Figure 2 - Procedures of detection method

3.7.11 Model Training and Testing using Flow Diagram.

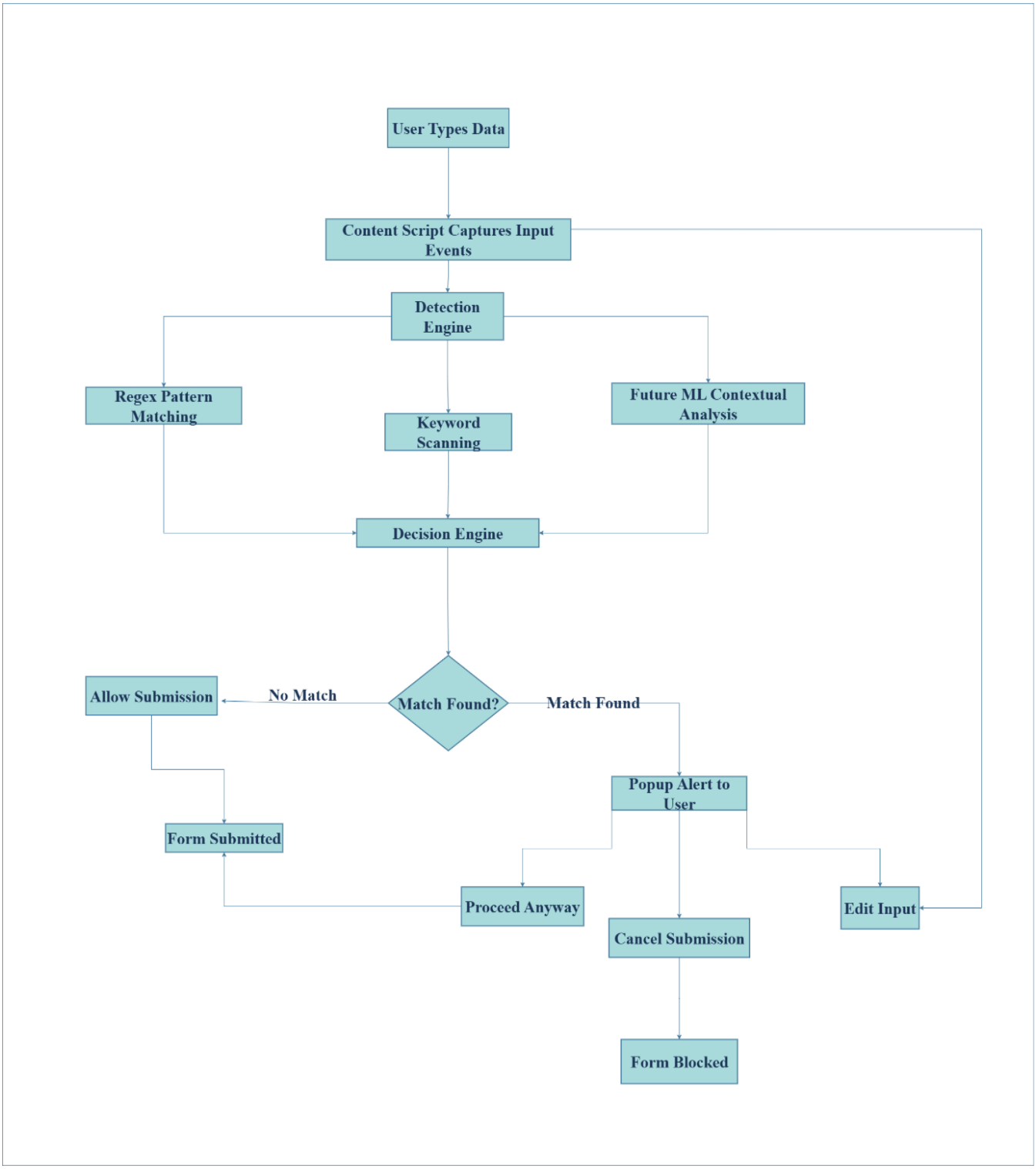


Figure 3 - Model Training and Testing Diagram

3.7.12 Assumptions and Dependencies.

Assumptions

- **User Environment**: Users will use modern Chromium-based browsers (e.g., Chrome, Edge, Brave, or Opera), which have implemented Manifest V3 extension support.

- **OS Compatibility**: The extension will function on recent versions of desktops running Windows, macOS, or Linux and will be equipped with appropriate CPU and RAM to conduct lightweight in-browser ML inference.

- **Network Availability**: The assumption is that users will have network connectivity to install or update the extension as well as to acquire signed updates and extensions from the Chrome Web Store.

- **User Understanding**: The assumption is that users will have a basic understanding of security prompts and know how to act in good faith when faced with the Cancel / Edit / Proceed popup.

- **Data Privacy**: Users and their organizations will adopt the solution under the impression that no raw sensitive data is sent off the local machine (this is privacy by design assumption).

- **Enterprise Adoption**: Organizations adopting this at scale will have the ability to distribute the extension in a centralized fashion (GPO/Intune/MDM).

Dependencies

- **Browser Vendor Policies**: The extension functionality is dependent on support for Manifest V3 APIs and WebAssembly/WebGPU features in Chromium-based browsers.

- **ML Frameworks**: Client-side inference is dependent on TensorFlow.js or ONNX Runtime Web being compatible with the browser runtime and the JavaScript engines available.

- **Extension Stores**: Distribution and automatic updates create a dependency on the practices of extension distribution mechanisms like the Chrome Web Store.

- **Development Toolchain**: The build and package rely on serving and packaging with Node.js LTS, npm, Webpack/Vite, ESLint, and other libraries.

- **Cloud/Offline Training Infrastructure**: Model training and optimization rely on training with Python-based ML frameworks (TensorFlow or Pytorch), GPU availability, and tools used for preparing the training/test dataset.

- **Security Compliance**: The project has a dependency on compliance with security frameworks (GDPR, HIPAA, PCI DSS) and enterprise IT governance scopes.

- **Third-Party Integrations**: Optional integrations for SIEM/GRC reporting depend on the organization's security infrastructure (Splunk, ELK, etc.).

4. System Features.

4.1 Real-time Input Monitoring.

The extension is always monitoring text that is typed or pasted into online forms, login fields, and AI chat portals. This will flag all sensitive information (like passwords and financial account numbers) that is about to leave the browser.

4.2 Multi-layered Detection.

It combines multiple techniques for stronger accuracy:

- *Regular Expressions* to catch structured data like credit card numbers or emails.
- *Keyword/heuristics* to detect terms such as “password” or “SSN.”
- *Machine Learning* for context-aware analysis (e.g., spotting hidden secrets in long sentences).

4.3 User-Friendly Interface.

- The extension is simple to install from the Chrome Web Store, takes up little memory and automatically loads in the background without causing a slow down in browsing. Alerts and popups are clear and easy to read.

4.4 Policy Customization.

- Organizations can tailor the detection engine by adding custom regex patterns, keywords, or allow-lists (for safe domains). This ensures the tool fits business-specific needs (e.g., detecting internal project codes).

4.5 Cross-Browser Support.

- Initially built for Chromium-based browsers (Chrome, Edge, Brave), with plans to adapt to Firefox in the future. This makes it usable across a wide variety of enterprise setups.

5. User Interface.

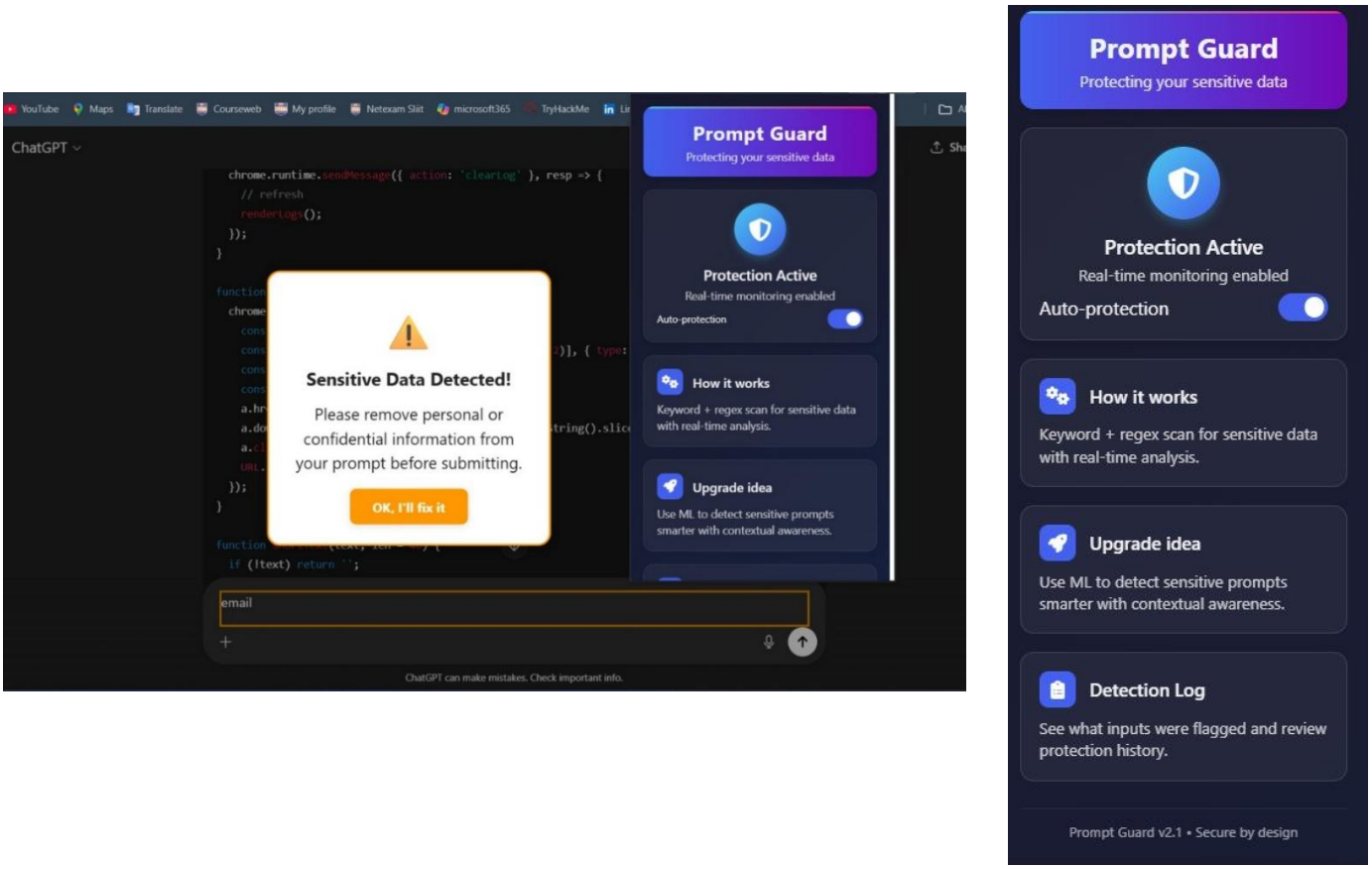


Figure 4 - Main UI

6. Other Nonfunctional Requirements.

6.1 Performance Requirements.

The extension needs to provide real-time detection and response while the user is typing or submitting a form – that is, it must not interrupt (or noticeably delay) the user experience. It should be lightweight and efficient to avoid impact to the browser's performance.

- **Response time** - Alerts must be triggered within milliseconds of detecting risky text.
- **Data processing efficiency** - Regex, keyword checks, and ML inference must run locally, and utilize minimal CPU and memory resources.
- **Scalability** - Must be capable of deployment on hundreds or thousands of enterprise endpoints, with no degradation in performance.
- **Flexibility** - Must allow organizations to customize detection rules, allow-lists, and sensitivity thresholds by organization.
- **Accuracy** - Must have high detection capability with low false positives and false negatives to preserve user trust.

6.2 Security Requirements.

The system must ensure **data protection, privacy, and resistance to unauthorized access**, complying with relevant security standards (e.g., GDPR, HIPAA).

- **CIA Requirements:**

- **Confidentiality** – All sensitive data is scanned locally; no raw data leaves the user's device.
- **Integrity** – Detection results and stored settings must not be altered maliciously or unintentionally.
- **Availability** – The extension interface (popups, alerts, settings page) must always remain responsive.

- **AAA Requirements:**
- **Authentication** – Enterprise deployments must verify users or devices before applying policies.
- **Authorization** – Only authorized admins can configure rules or export logs.
- **Accountability** – Optional anonymized logs ensure traceability without exposing PII.

6.3 Safety Requirements.

The system must be used in a way that promotes safe use and minimizes potential misuse, manipulation by the user or some other individual or group, or negative or detrimental impact on the user browsing experience.

- **Backup and Restoration** - Configurations and allow lists must be saved and able to be restored according to usage patterns, including in the case of corruption or when re-installation is required.
- **Error Handling** - ML inference and regex modules must demonstrate a graceful failure mode (e.g., defaulting to rules-only detection), when those fail without error and/or other catastrophic failures.
- **Data Integrity** - Alerts and logs must accurately report detected events regarding undistinguished event loss or corrupted event.
- **User Safety** - Alerts and pop ups must not interfere with important, required user actions (such as online banking transactions, etc.).
- **Best Practice and Industry Compliance** - Design must adhere to privacy-by-design and the industry's best practice for cybersecurity.

7. Overall Software Stack Summery.

In the project "Browser Extension for Preventing Sensitive Data Leaks Using Machine Learning," the Software Stack specification contains several interrelated components that work together to enable effective detection and prevention of sensitive information being disclosed by users while they use web applications. The specification specifies the appropriate software, frameworks, libraries, and tools necessary to build, deploy, and manage the browser extension, as well as its detection engine based on machine learning. Below is the Software Stack specification in regard to the project,

7.1 Operating Systems and Platforms.

- **Client Platforms:** Windows, macOS, Linux, and ChromeOS, supporting Chromium-based browsers (Google Chrome, Microsoft Edge, Brave).
- **Development & Testing Platforms:** VMware Workstation and Oracle VirtualBox environments to validate cross-OS/browser compatibility.
- **Optional Enterprise Deployment Platforms:** Windows Server or Linux servers for centralized policy rollout, log aggregation, and SIEM integration.

7.2 Development Languages and Frameworks.

- **JavaScript (ES6+), HTML, CSS:** Core technologies for building the extension (content scripts, background scripts, popup UI).
- **Chrome Extension APIs (Manifest V3):** For secure integration with browser runtime, storage, and scripting.
- **Node.js with Webpack/Vite:** For bundling, packaging, and building the extension.
- **Python (3.10+):** For model training, preprocessing, and dataset handling.

7.3 Machine Learning and Data Analysis.

- **TensorFlow / PyTorch:** For training and validating the ML model.
- **TensorFlow.js / ONNX Runtime Web:** For running the trained model in-browser during real-time detection.
- **Pandas / NumPy:** For preprocessing, cleaning, and preparing textual datasets.
- **Scikit-learn:** For evaluation, feature extraction, and baselines.

7.4 Network Analysis and Management Tools (Dataset Collection).

- Wireshark / Burp Suite (optional): For gathering sample traffic in order to create actualistic datasets for training (without disclosing any true sensitive data).
- Publicly Available Datasets: Use of anonymized PII and sensitive data corpora for model training.
- Custom Data Simulation Tools: Scripts to simulate/create synthetic datasets (emails, passwords, financial information) for supervised training and testing.

7.5 Database and Storage Solutions.

- **Local Encrypted Storage** (Browser Storage API): For storing user settings, allow-lists, and detection policies.
- **Object Storage (S3/MinIO):** For storing trained ML models and versioned artifacts.

7.6 Security and Encryption Tools.

- **TLS/HTTPS**: For secure transmission when connecting to external services (e.g., admin console, SIEM).
- **Code Signing / Extension Signing**: To ensure the integrity of distributed browser extension packages.
- **7.7 Integration and Deployment Tools**
- **Git and GitHub/GitLab** - Used for version control and collaborative team development.
- **CI/CD Tools** (such as GitHub Actions or GitLab CI) - Used for automated testing of the package and publishing of extension builds.
- **Chrome Web Store Developer Dashboard** - Used for deployment and distribution of updates to end users.

8. References.

[1] *Mitigating information leakage in large language models: Evaluating the impact of code obfuscation on vulnerability detection*. (2025, June 30). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/11129599

[2] *Fine-tuned Large Language Models (LLMS): Improved prompt injection attacks detection*. (2025, July 8). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/11126597

[3] *On Protecting the data privacy of Large Language Models (LLMs): a survey*. (2024b, June 20). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/11062758

[4] *Privacy and security challenges in large language models*. (2025, January 6). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10903912

[5] *LC-SID: Developing a local LLM-based chain-of-thought framework for enhanced sensitive information detection*. (2024, December 2). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10925018

[6] *Detection of inconsistencies in privacy practices of browser extensions*. (2023, May 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10179338

[7] *ExtensionGuard: Towards runtime browser extension information leakage detection*. (2016, October 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/7860481

[8] *Privacy model: Detect privacy leakage for Chinese browser extensions*. (2021). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9369313

[9] *Privacy-Preserving detection of sensitive data exposure*. (2015, May 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/7038200

[10] *Privacy model: Detect privacy leakage for Chinese browser extensions*. (2021b). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9369313

9. Progress.

The Machine Learning (ML) and Artificial Intelligence (AI) components of the detection engine for the Browser DLP Extension are crucial for accurately detecting sensitive information (such as PII, credentials, and financial data) in real-time. Below is a description of the systems, development frameworks, and methodologies required,

9.1 Data Requirements.

- Labeled collections of sensitive text and non-sensitive text, for example: e-mails, passwords, credit card numbers, internal codes.
- Synthetic data generated for specific testing of leak scenarios.
- Privacy data made public which is anonymized PII and credentials.

9.2 Data Pre-processed.

- Tokenization, text normalization, and noise removal. Positive/negative samples balanced to avoid model bias.

9.3 Specifications for the Model.

Model Type:

- Lightweight NLP-based classification model (e.g., Transformer-based or LSTM-based) aimed at real-time inference.

9.4 Performance Specifications.

- The model must produce low-latency inference (<100 ms each text).
- The model must have very high accuracy with precision and recall exceeding 90% in controlled test conditions.

Model Delivery:

- Models must be converted to TensorFlow.js or ONNX Runtime Web, and executed in the browser.
- The model is required to support quantization and/or pruning to facilitate a model size of less than 10 MB for fast loading.

9.5 Algorithm and Framework Requirements.

- Training Frameworks: Python (3.10+) with TensorFlow or PyTorch.
- Libraries for Preprocessing & Evaluation: Pandas, NumPy, Scikit-learn, Hugging Face Transformers (if transformer-based models are used).
- Browser Inference Frameworks: TensorFlow.js or ONNX Runtime Web to ensure client-side execution.

9.6 Hardware requirements to Develop a Model.

Technical Requirements:

- GPU-enabled environment (either local, such as NVIDIA RTX 3060 or better, or cloud GPU, e.g., AWS EC2 G4/G5, Google Colab Pro, or Azure ML).
- Minimum 16 GB RAM for local training / GPU use.

Inference Hardware:

- Run on a standard end-user device (e.g., laptop, desktop, Chromebooks). No dedicated GPU required.

Sample data sets collected

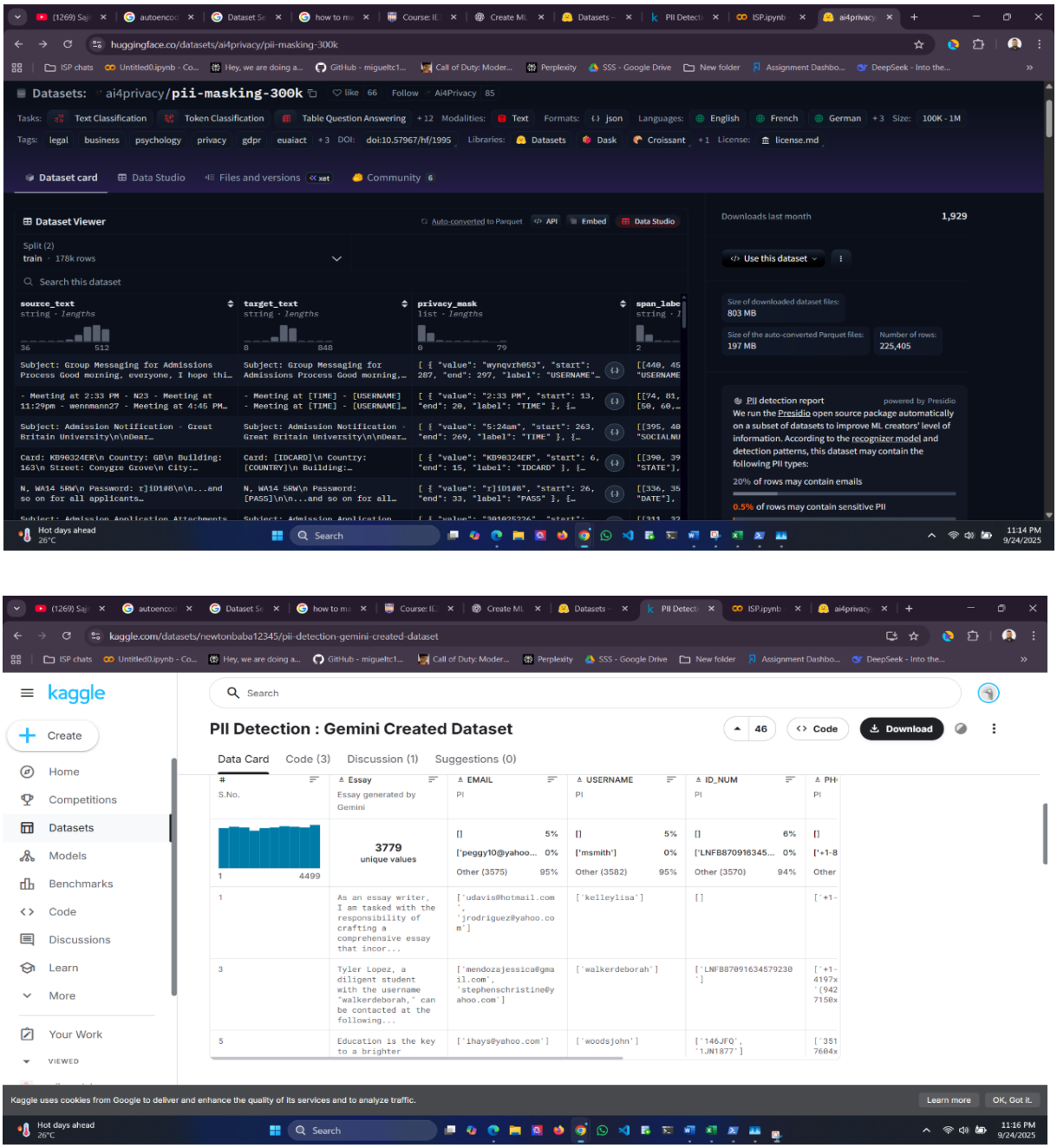
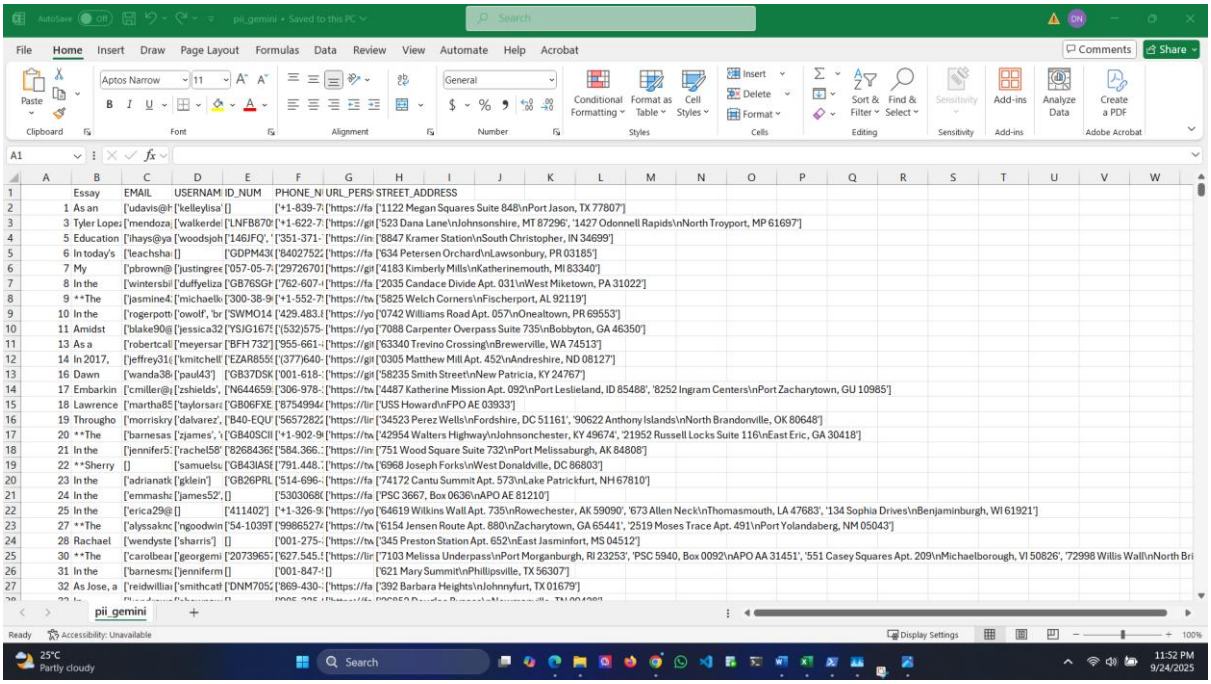


Figure 5 - Dataset Exploration and PII Detection Summary



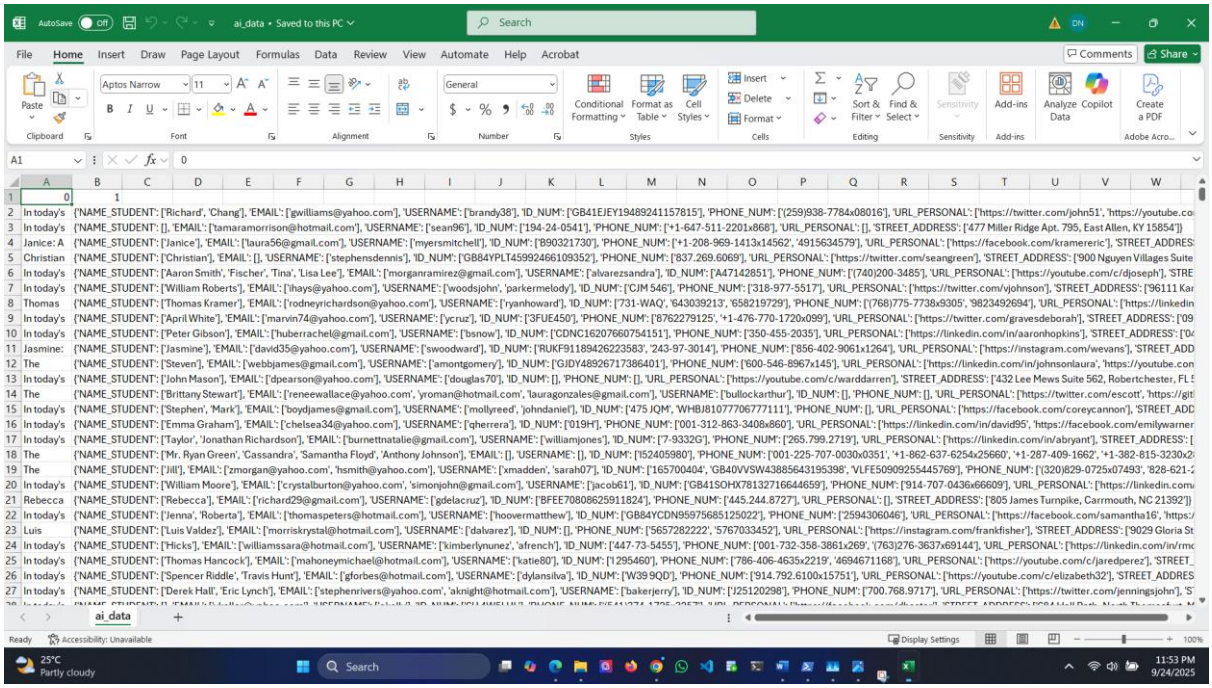


Figure 6 - Raw and Processed PII Data Examples in Spreadsheet Format

Project Begin proof – Model Creation.

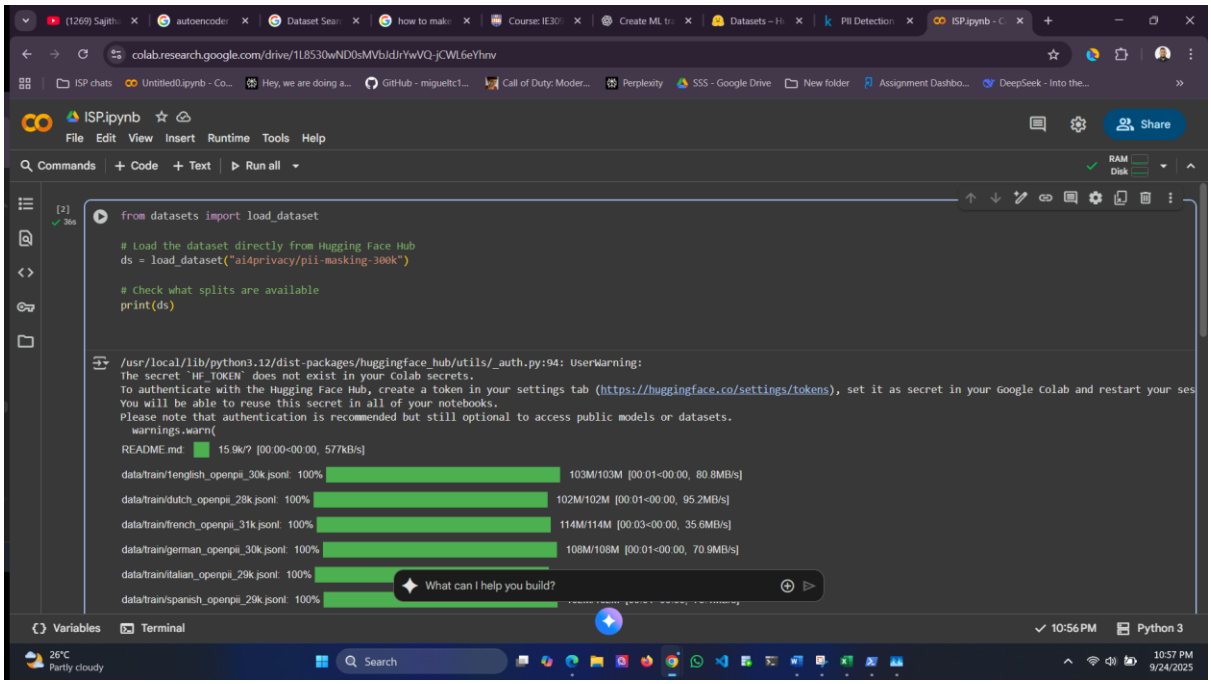


Figure 7 - Data Loading and Preparation Progress

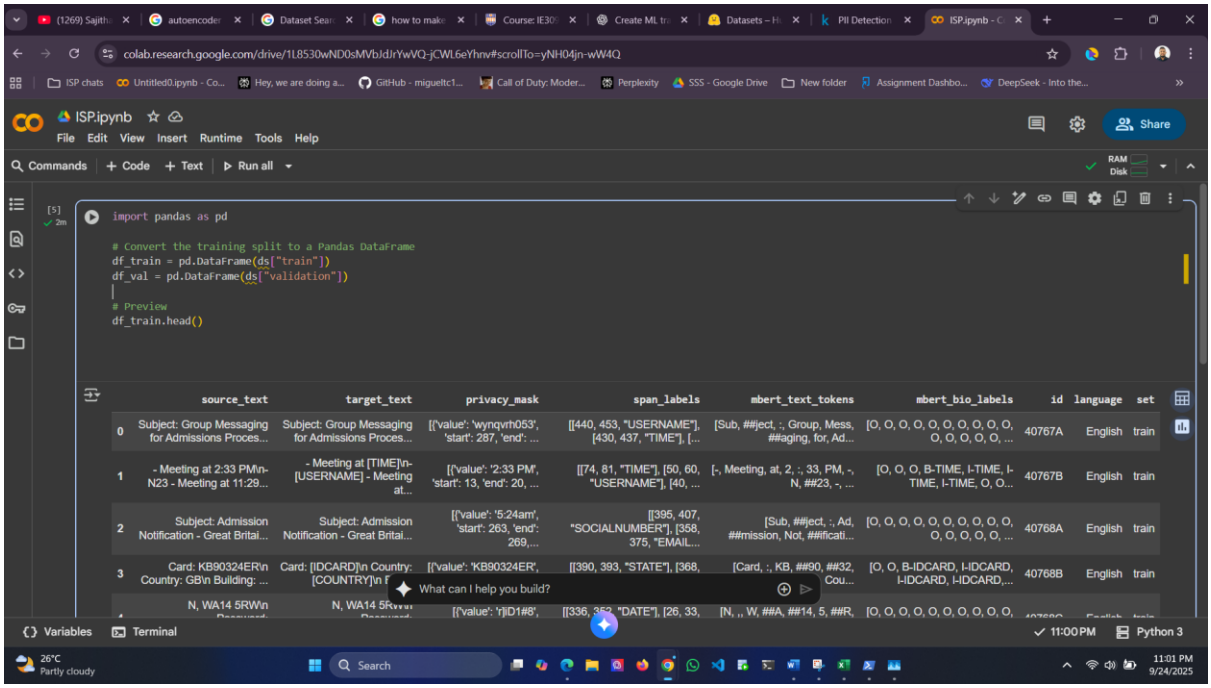


Figure 8 - Data Transformation and Feature Engineering Overview

10 Project Stack – Model Development.

The machine learning (ML) model is an essential part of the Browser DLP Extension that detects sensitive information in browser activity in near real time. The model development

stack encompasses tools and frameworks for preparing a dataset, serving as the first hack in the development lifecycle, all the way through model training, modern versions of optimization, and onward deployment in the browser.

10.1 Programming Language.

- Python 3.10+ - the main programming language used for data preprocessing, training, and evaluating the model.
- JavaScript (ES6+) - this will be used to integrate the trained model into the browser extension via inference frameworks.

10.2 ML Frameworks.

- TensorFlow / PyTorch - the machine learning framework used to train (and fine-tune for) text classification or token-level detection models.
- Hugging Face Transformers - for pre-trained LLMs (eg, DistilBERT, BERT-mini) that have been adapted to detect sensitive data.
- Scikit-learn - for baseline classifiers, feature extraction, and evaluation metrics.

10.3 Data Processing & Analysis.

- Pandas / NumPy - for cleaning, preprocessing, and transforming text data.
- NLTK / spaCy - for tokenization, stopword removal, and linguistic preprocessing.
- Custom scripts - for generating samples of synthetic sensitive data (eg, dummy credit cards, passwords).

10.4 Optimization and Conversion of the Model.

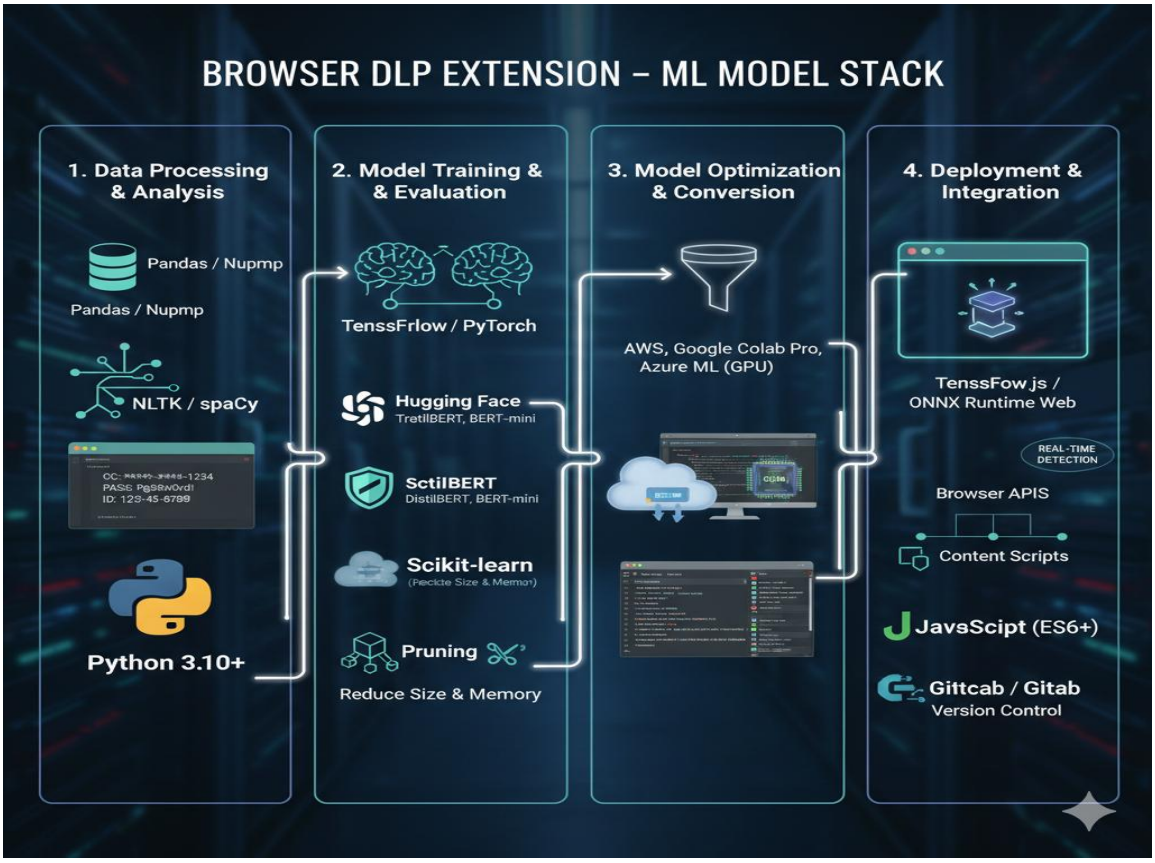
- ONNX/TensorFlow.js Converter: convert trained models to lighter-weight browser-compatible formats.
- Quantization and Pruning Tools to decrease the size of the model and memory for quicker execution within the browser.

10.5 Training and Evaluation Environment.

- Local GPU Machine or Cloud Services (AWS, Google Colab Pro, Azure ML) equipped with a GPU supporting CUDA.
- Jupyter Notebook / Visual Studio Code - to iterate on the development and test the model.
- Evaluation Metrics - Precision, Recall, F1-score, and confusion matrices.

10.6 Deployment and Integration.

- TensorFlow.js/ONNX Runtime Web to run the optimized model in a browser extension.
- Browser APIs to enable integration of the ML inference engine with the content scripts observing the user's inputs.
- Version Control (Github/GitLab) to manage the model code, datasets, and artifacts.


BROWSER DLP EXTENSION – ML MODEL STACK
1. Data Processing & Analysis
Pandas / Nupmp
Pandas / Nupmp
NLTK / spaCy
Python 3.10+
2. Model Training & & Evaluation
TenssFrlow / PyTorch
Hugging Face
TretIlBERT, BERT-mini
SctilBERT
DistilBERT, BERT-mini
Scikit-learn
Pruning
Reduce Size & Memory
3. Model Optimization & Conversion
AWS, Google Colab Pro, Azure ML (GPU)
4. Deployment & Integration
TenssFow.js / ONNX Runtime Web
REAL-TIME DETECTION
Browser APIS
Content Scripts
JavsScipt (ES6+)
Gittcab / Gitab
Version Control

Thank You!