Chrome Extension for Monitoring Sensitive Information Exposure in WhatsApp Web

---

Part 1: Specification

Objective:

The primary goal of this Chrome extension is to enhance user privacy and security on WhatsApp Web by detecting and alerting users to potential exposure of sensitive or private information. The extension focuses on identifying various types of personal data within messages.
Addresses: Home or work addresses that could be shared unintentionally in conversations.
Locations: GPS coordinates, locations, or references to specific places that may reveal the user's whereabouts.
Financial Details: Information about bank accounts, credit card numbers, or other financial data.
Personal Routines: Details about users' daily schedules, routines, or habits that could be used maliciously.
By identifying these types of sensitive information, the extension helps protect users from unintended privacy breaches, identity theft, or other forms of exploitation.
The extension focuses on providing users with actionable options when sensitive information is detected in their messages. Upon detecting potential privacy concerns, the extension alerts the user and offers a clear, concise description of the issue, such as the presence of personal or sensitive information. The user will be presented with options to either delete or edit the message, helping them take immediate action to protect their privacy. The interface is designed to be intuitive, allowing users to quickly understand what privacy issue was detected and make decisions on how to address it. This ensures a seamless, user-friendly experience while maintaining privacy and security.

Target Audience:

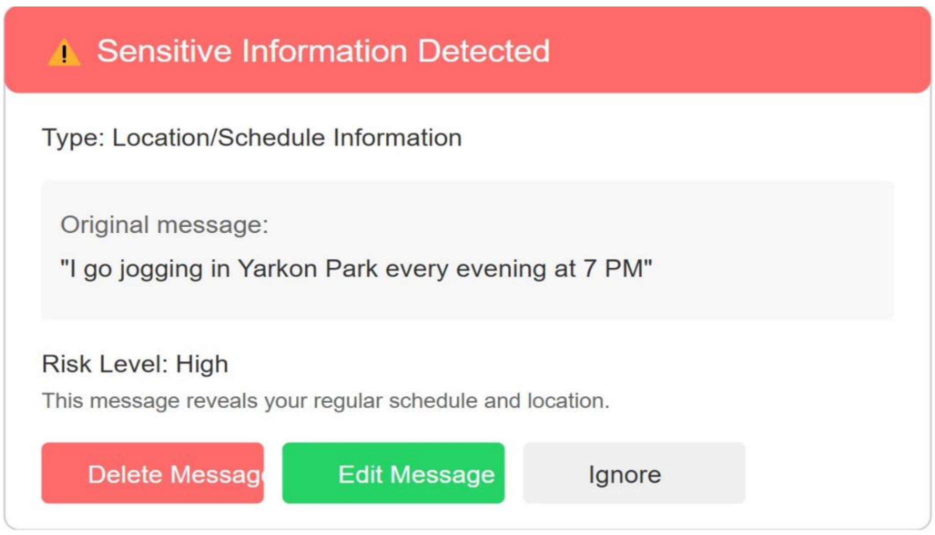
- **Parents**: Monitoring their children's chats to ensure their privacy and safety.
- **Adults**: Helping individuals avoid unintentionally sharing private information.
- **Women and Girls**: Protecting those at higher risk of online exposure.

Key Features:

1. **Real-Time Message Analysis**:
    - Analyzes text in outgoing messages on WhatsApp Web.
    - Uses NLP algorithms to identify sensitive or risky content.
2. **Immediate Alerts**:
    - Notifies the user if a message contains potentially sensitive information.
    - Provides suggestions for alternative actions, such as rephrasing or deleting the message.
3. **Centralized Alert Dashboard**:
    - Allows users to view all alerts generated by their account.
    - Supports viewing alerts for other linked accounts (e.g., children's accounts) with appropriate permissions.
4. **Customizable Sensitivity Settings**:
    - Allows users to adjust the sensitivity of the detection to match their specific needs.
5. **Future Feature - Voice Message Detection:**
    - In addition to analyzing text messages, this feature will enable the detection of sensitive content in voice messages.
    - By converting voice messages to text using speech-to-text technology, the extension will analyze the transcribed text for private or sensitive information.
    - If any risky content is detected, users will receive alerts similar to those for text messages, allowing them to address privacy concerns effectively.

Usage Flow:

1. **Message Sending**:
   - The user types and sends a message in WhatsApp Web.
   - The extension detects when the message is sent and captures its content.
2. **Detection and Alert**:
   - The extension analyzes the message content.
   - If sensitive information is detected, an alert is triggered:
     - A pop-up notification appears in the browser.
     - The alert is also stored in a daily report, which can be accessed later.



⚠ Sensitive Information Detected

Type: Location/Schedule Information

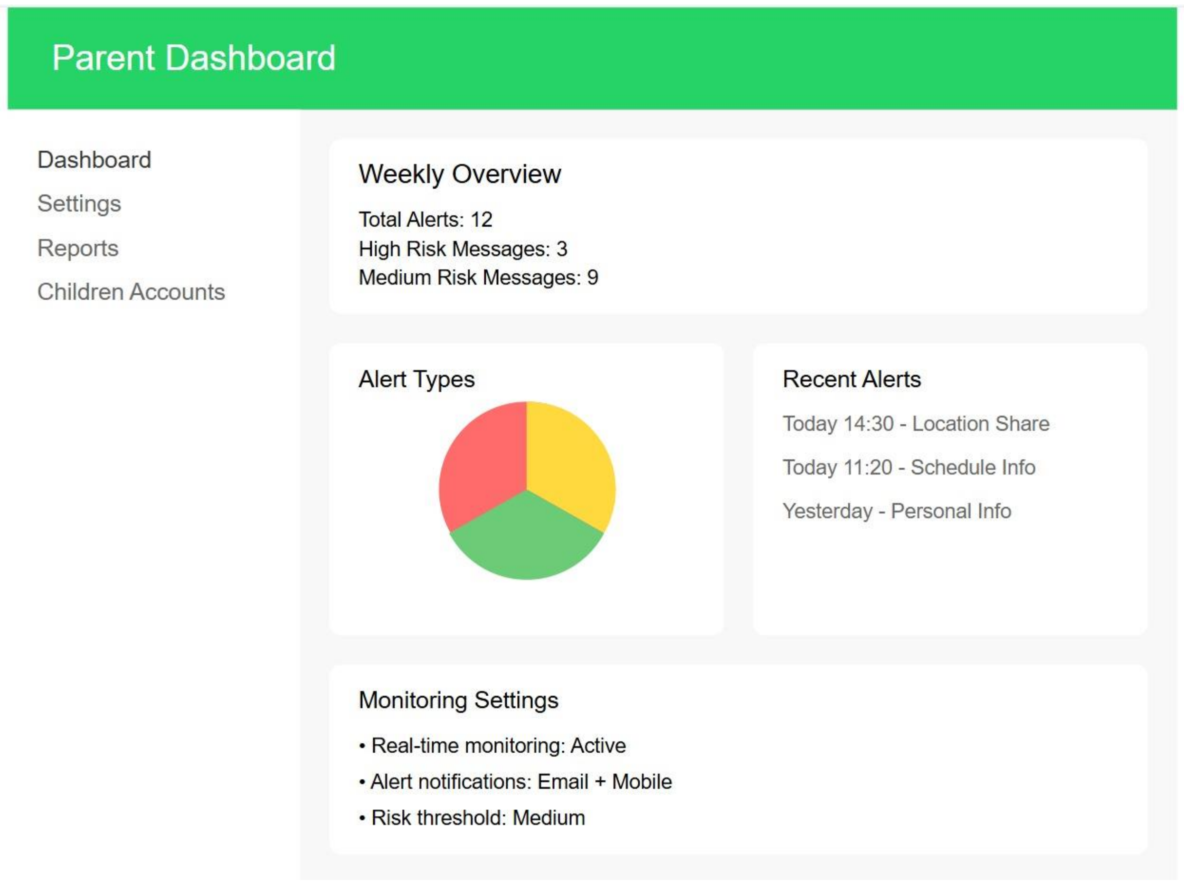
Original message:
"I go jogging in Yarkon Park every evening at 7 PM"

Risk Level: High
This message reveals your regular schedule and location.

3. **Post-Action Options**:
   - User can:
     - Acknowledge and dismiss the alert.
     - Modify the alert settings for future messages.

4. **Parental Monitoring (Optional):**
   - Parents can receive daily reports summarizing the alerts generated by their children's accounts.

- These reports are provided at the end of the day, allowing parents to monitor their child's messages without receiving constant pop-up alerts. This approach offers a less intrusive way to keep track of potential privacy issues while still providing a comprehensive overview of the day's activity.
- The report includes high-level summaries of the detected sensitive content, but without revealing the full message content, ensuring the child's privacy is maintained.



Weekly Overview

Total Alerts: 12
High Risk Messages: 3
Medium Risk Messages: 9

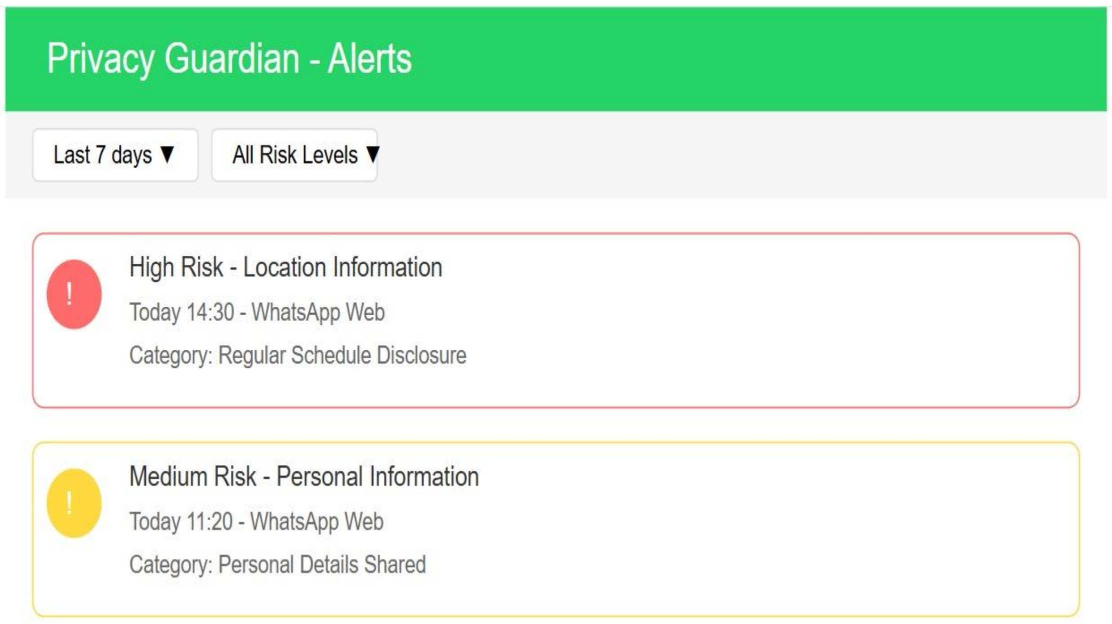
Alert Types

Recent Alerts

Today 14:30 - Location Share
Today 11:20 - Schedule Info
Yesterday - Personal Info

Monitoring Settings

- Real-time monitoring: Active
- Alert notifications: Email + Mobile
- Risk threshold: Medium



Privacy Guardian - Alerts

Last 7 days ▼ All Risk Levels ▼

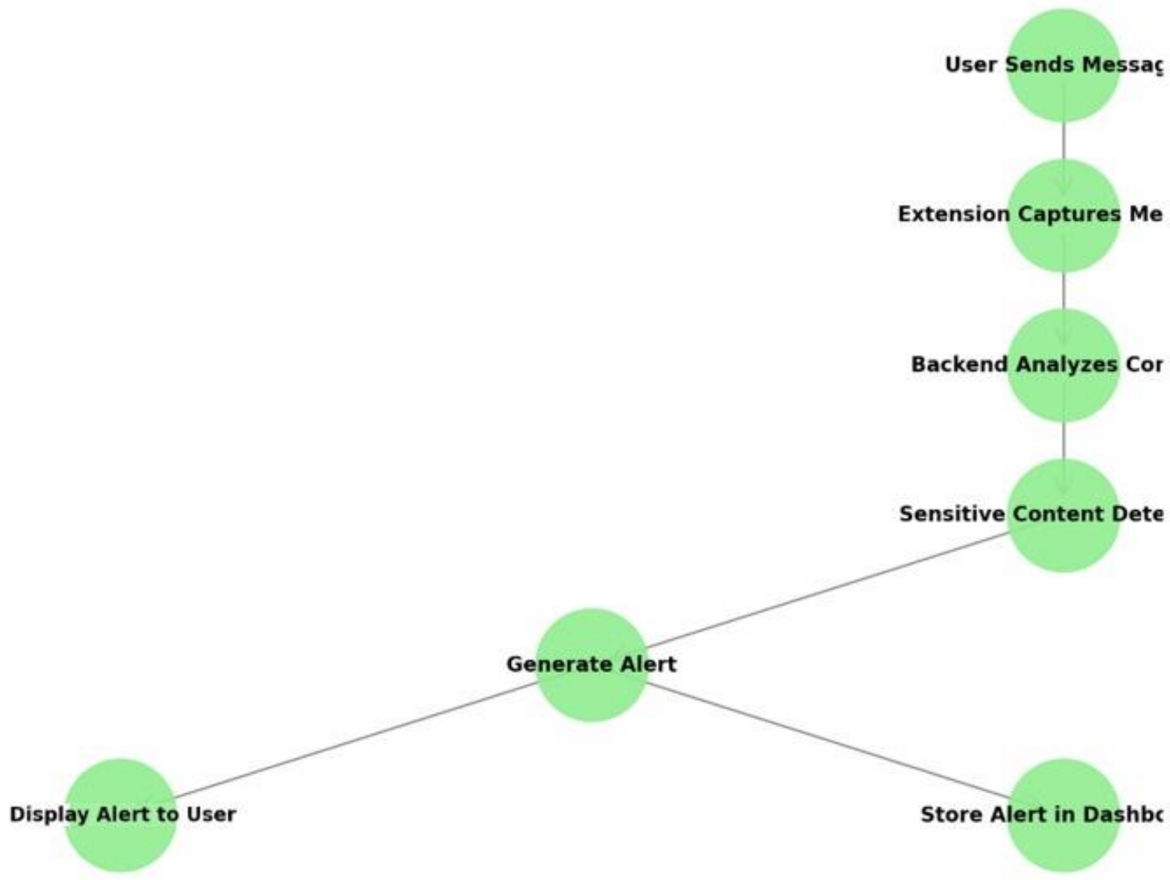
High Risk - Location Information
Today 14:30 - WhatsApp Web
Category: Regular Schedule Disclosure

Medium Risk - Personal Information
Today 11:20 - WhatsApp Web
Category: Personal Details Shared

Example Scenarios:

1. **Sensitive Content Alert**:
   - User types: "I'll be home alone after 4 PM because my parents work late."
   - Alert: "This message contains personal schedule details. Consider removing this information to protect your privacy."
2. **Privacy Education**:
   - User repeatedly shares location information.
   - Extension provides: "Sharing your location can pose a security risk. Avoid providing precise locations in casual conversations."
3. **Parental Monitoring**:
   - Child types: "I'm on vacation in Eilat until the weekend."
   - Child types: "Come play at my house; I live close to school.

Improved Flow Diagram: Sensitive Content Detection Process



Part 2: High-Level Design

Architecture Overview:

The extension is divided into three primary components:

1. **Frontend (Browser Extension):**
    - **Function:** This component monitors WhatsApp Web DOM for outgoing messages. It injects a content script into the web page and listens to new massages sent from the user.
    - **Responsibilities:**
        - Captures the text of outgoing messages as they are sent.
        - Sends message content to the backend for analysis.
        - Displays alerts if sensitive information is detected in the message.
        - Provides users with actionable options (delete, edit) upon receiving an alert.
        - Shows a user-friendly interface for interacting with the extension.
2. **Backend (Spring Boot NLP Engine and Dashboard Server):**
    - **Function:** This backend handles the processing of messages for sensitive content. It provides the necessary NLP (Natural Language Processing) algorithms to detect personal data and other sensitive information.
    - **Responsibilities:**
        - **Spring Boot:** Serves as the application framework to handle HTTP requests and manage message processing logic.
        - **NLP Engine:** Analyzes the messages for sensitive information like addresses, locations, financial details, etc., using libraries like **Spacy**, **Apache OpenNLP**, or **NLTK**.
        - **WebSocket:** Facilitates real-time communication between the extension and the backend for sending alerts back to the user's browser immediately after sensitive data is detected.

        - **Reports & Alerts:** The backend stores and manages the alerts. It generates daily reports that are accessible to the user (or parents for monitoring).

3. **Centralized Alert Dashboard:**
    - **Function:** This is a separate web-based dashboard where users can view aggregated alerts from their WhatsApp Web usage. This dashboard is especially useful for parental monitoring.
    - **Responsibilities:**
        - Provides a secure authentication system to link multiple accounts (e.g., children's accounts).
        - Displays a list of alerts with actionable information.
        - Allows users to customize their notification settings and sensitivity thresholds.

Component Breakdown:

---

1. Frontend (Chrome Extension)

- **Language & Frameworks:**
    - **JavaScript**: For the logic of detecting and sending messages to the backend.
    - **HTML/CSS**: To build the user interface (UI) for alerts and configuration settings.
- **Key Functions:**
    - **Monitor WhatsApp Web DOM:** Interacts with the WhatsApp Web page to detect when a message is sent.
    - **Capture Message Content:** Extracts the message content from the DOM and sends it to the backend for processing.
    - **Display Real-Time Alerts:** Using browser pop-ups and in-page notifications to inform the user when sensitive information is detected.

    - **Allow User Action:** Displays options to either delete or modify the message, and also adjust settings like sensitivity or notification preferences.
- **Communication with Backend:**
    - Sends HTTP requests via **AJAX** or **Fetch API** to the Spring Boot backend for message processing.
    - Receives alerts and updates via **WebSocket** for real-time interaction.

---

2. Backend (Spring Boot NLP Engine)

- **Language & Framework:**
    - **Spring Boot**: For building the RESTful API to handle requests from the extension, managing user settings, and serving the dashboard.
    - **NLP Libraries**: **Spacy**, **Apache OpenNLP**, or **NLTK** for processing messages and identifying sensitive content like addresses, locations, and financial details.
- **Key Functions:**
    - **Message Analysis**: The backend receives messages and analyzes them using NLP models and regex patterns. It assigns a risk score based on the identified sensitive data.
    - **Alert Generation**: When sensitive content is detected, the backend generates an alert that is sent back to the extension in real-time via **WebSocket** or HTTP.
    - **Alert Management**: The backend keeps track of all alerts, storing them in the database for later access (e.g., daily reports, parental monitoring).
    - **Reports**: The backend compiles alerts into daily or periodic reports, especially for parental monitoring.
    - **User Management**: Handles user preferences and accounts, including sensitivity settings and notification preferences.

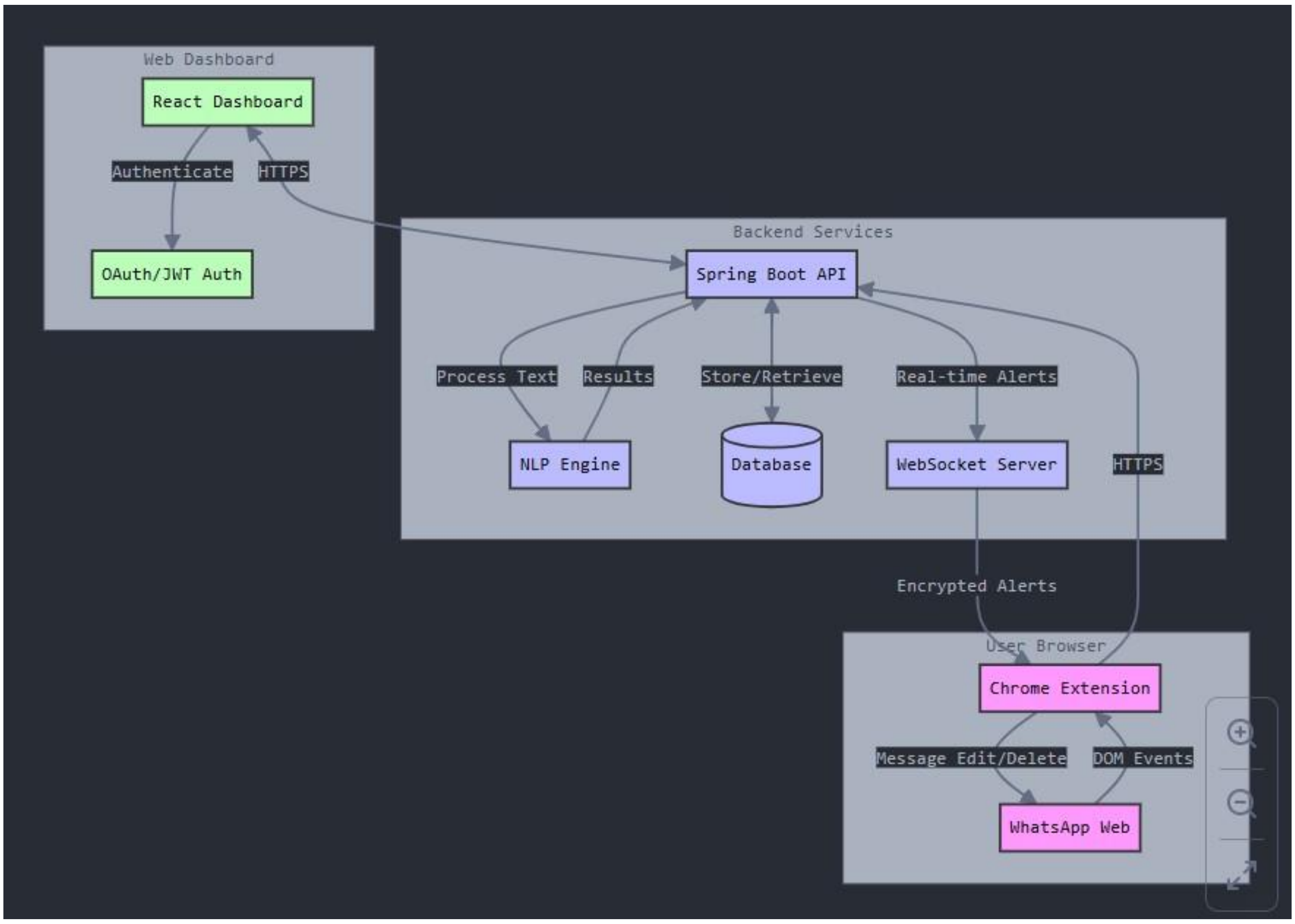
- **Security Considerations:**
  - **Encryption**: All communication between the extension, backend, and the dashboard is encrypted using HTTPS and secure WebSocket connections.
  - **Data Privacy**: Sensitive information like full message content is not stored on the backend; only anonymized data and risk scores are logged for analysis.

---

3. Centralized Alert Dashboard

- **Frontend:**
  - **React.js**: For building a dynamic and interactive dashboard to display alerts and allow users to manage their settings.
  - **UI Components**: Includes a user login interface, alert history, and sensitive content summary.
- **Backend:**
  - **Spring Boot**: Provides APIs for the dashboard to interact with the backend, fetching user alerts and managing preferences.
  - **User Authentication**: OAuth 2.0 or JWT for secure authentication and linking of multiple accounts (e.g., for parents monitoring their children's messages).
- **Features:**
  - **User Authentication**: Secure login system to link accounts and monitor alerts.
  - **Alert Management**: Allows users to view, delete, or manage alerts, set custom sensitivity levels, and configure parental monitoring settings.
  - **Reports**: Access to daily or periodic reports that summarize detected sensitive information without revealing full message content.

System Design:



Data Flow:

---

1. **Message Capture**:
   - The Chrome extension detects when a user sends a message through WhatsApp Web. The message content is intercepted and sent to the Spring Boot backend for processing.
2. **Content Analysis**:
   - The Spring Boot backend analyzes the content using NLP algorithms and regex to identify any sensitive information (addresses, locations, financial details, etc.).
   - If sensitive content is detected, it assigns a risk score and generates an alert.
3. **Risk Evaluation**:
   - The backend evaluates the message and calculates a risk score based on the identified sensitive elements. The higher the score, the more sensitive the message is deemed.
4. **User Notification**:
   - The backend sends the results back to the extension in real-time using **WebSocket** or **AJAX**. A pop-up alert is triggered in the user's browser to notify them of the detected risk.
5. **Parental Monitoring**:
   - Alerts will save in the parental dashboard, providing high-level information (e.g., type of content detected) without disclosing full message details to maintain privacy.

Database Structure (Example):

Users Table:

Field	**Type**	**Description**
UserPhoneID	String	Unique Phone for each user
UserName	String	User Name
Password	String	User Password
AllowedUsers	List<String>	Connected Children for parental dashboard

Alerts Log Table:

Field	**Type**	**Description**
LogID	String	Unique log entry ID
Alert	String	Alert content
RiskScore	Integer	Calculated risk score
UserPhoneID	String	Unique Phone for each user

Tools and Libraries:

- **Frontend**:
    - **Chrome Extension APIs**: For message detection and interaction with WhatsApp Web.
    - **React.js**: For building the dashboard UI .
- **Backend**:
    - **Spring Boot**: For building the RESTful API and managing the backend logic.
    - **NLP Libraries**: **Spacy**, **Apache OpenNLP**, **NLTK**.

    - **WebSocket**: For real-time communication between the backend and frontend.
    - **Database**: PostgreSQL or MongoDB for storing user preferences and alert data.
- **Testing and Debugging**:
    - **Mocha/Jest**: For frontend and backend unit testing.
    - **Chrome DevTools**: For debugging content scripts.

Additional Considerations:

Security and Privacy:

1. **Data Minimization**: Ensure minimal data is sent to the backend, only sending anonymized or essential information.
2. **Encryption**: Use HTTPS for secure communication and WebSocket encryption to ensure data integrity.
3. **Parental Monitoring Privacy**: Ensure no full message content is sent to parents, protecting child privacy.

Deployment and Maintenance:

1. **Browser Compatibility**: Focus on Chrome for the extension, with potential support for Firefox and Edge.
2. **Extension Updates**: Regular updates to improve NLP models, detection accuracy, and user experience.
3. **Scalability**: The backend is designed to be scalable, allowing for the addition of new features and handling increased user load.

This design outlines the architecture and components required to build the Chrome extension with Spring Boot as the backend. It provides real-time sensitive information detection and alerts while focusing on privacy and user control.