MALLA REDDY COLLEGE OF
ENGINEERING AND TECHNOLOGY

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

for

Adblocker Using Random Forest Algorithm

Version 1.0

Prepared by

S. No	Roll Number	Student Name
1.	22N31A66D9	P. Hussain Setty
2.	22N31A66H6	T. Vijeshwar Reddy
3.	22N31A66J6	Y.Raja Sekhar Reddy

Supervisor:	Mrs. B.Jyothi
Designation:	Associate Professor
Department:	Computational Intelligence
Batch ID:	
Date:	
Supervisor Sign. & Date	

Department of Computational Intelligence

Title of the Project: Adblocker Using Random Forest Algorithm

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	George Orwell	Primary Revision giving an overall view of the project and document.	24/02/25

1 Introduction

1.1 Document Purpose

Online ads are everywhere, often appearing on sites like Facebook and Google, and sometimes matching your recent searches. This happens due to web tracking, which tracks your activity online. While it was created to improve browsing, it is now often misused to show intrusive ads and collect personal data.

Traditional ad blockers help reduce ads, but they rely on outdated methods and can't always detect new ad patterns. This leaves users exposed to unwanted content. To address this, we created Ad Remover, a smarter ad blocker powered by machine learning. It uses a Random Forest model to analyze URLs and accurately block over 98% of ads. Unlike older tools, Ad Remover adapts to new ad types, ensuring a smoother and safer browsing experience.

1.2 Project/Product Scope

The **Ad Blocker using Random Forest Algorithm** is a system designed to detect and block online advertisements, improving the user's browsing experience by removing intrusive ads. It works by analyzing webpage elements such as HTML structure, CSS properties, and URLs to classify content as ads or non-ads. Using the Random Forest algorithm, the system is trained on labeled data to accurately identify and filter out advertisements in real time. The ad blocker will function as a browser extension or proxy-based tool, compatible with major browsers like Chrome and Firefox. Users will have options to customize their ad-blocking preferences, including whitelisting websites and blocking specific ad types like pop-ups and banners. While the system aims to efficiently reduce ads without affecting browsing speed, it may not work on offline applications or counter advanced anti-adblock mechanisms used by some websites. Overall, this project provides a smart and user-friendly solution for blocking unwanted ads and enhancing privacy while browsing the web.

The core objectives of the system include:

- **Ad Detection and Classification:** Use the **Random Forest algorithm** to accurately identify and classify ads based on webpage elements like HTML tags, CSS properties, and URLs.
- **Real-Time Ad Blocking:** Automatically block pop-ups, banners, and video ads while ensuring smooth browsing without delays.
- **Customizable User Control:** Allow users to **whitelist or blacklist websites**, choose which types of ads to block, and adjust settings based on preferences.
- **Integration with Ad-Blocking Lists:** Support **Easy List and other ad filter lists** to improve accuracy and effectiveness.
- **Improved Browsing Experience:** Reduce page load times by blocking unnecessary scripts and media, leading to a **faster and smoother** browsing experience.
- **Enhanced Privacy and Security:** Prevent tracking ads from collecting user data, ensuring better **online privacy and security**.
- **Minimal Resource Usage:** Optimize the system to run **efficiently without slowing down the browser or consuming excessive memory**.

1.3 Existing System

Currently, several ad-blocking solutions are available, primarily in the form of **browser extensions, built-in browser features, and network-level blockers**. The primary methods used today include:

- Filter-Based Ad Blockers: Most existing ad blockers, like AdBlock, AdBlock Plus, and uBlock Origin, use predefined filter lists (e.g., EasyList) to detect and block ads.
- Built-in Browser Ad Blockers: Some browsers, such as Brave and Opera, come with integrated ad-blocking features, but they still rely on static rules rather than intelligent detection.
- Network-Level Ad Blockers: Tools like Pi-hole block ads at the DNS level, preventing ad servers from loading, but they cannot selectively remove in-page ads.
- Limited Intelligence: Traditional ad blockers do not use machine learning, making them less adaptable to new ad formats and unable to detect evolving ad techniques.
- Challenges with Anti-Adblock Scripts: Many websites use anti-adblock mechanisms to detect and block users with ad blockers, reducing their effectiveness.

1.4 Problems with Existing System

Main Problems in the Existing System

- **Rule-Based Limitations:** Traditional ad blockers rely on **predefined filter lists**, which need frequent updates and fail to detect **new ad formats**.
- **Ineffective Against Anti-Adblock Techniques:** Many websites use **scripts to detect and bypass ad blockers**, forcing users to disable them to access content.
- **High False Positives and Negatives:** Some ad blockers **mistakenly block useful content** or fail to detect certain ads, leading to an incomplete browsing experience.
- **Performance Issues:** Some ad blockers **consume too much system memory and CPU power**, slowing down web browsing, especially on ad-heavy websites.
- **Limited Adaptability:** Traditional ad blockers **cannot learn from new ads** and require manual updates, making them **less effective over time**.

1.5 Proposed System

The proposed system uses the **Random Forest algorithm** for accurate ad classification, ensuring that advertisements are correctly identified and blocked. Unlike traditional ad blockers, it **dynamically adapts to new ad patterns**, allowing it to remain effective even as advertisers change their techniques. The system also provides **customizable ad-blocking preferences**, enabling users to whitelist or blacklist specific websites and control which types of ads they want to block. By reducing **false positives**, it enhances the overall **user experience**, ensuring that useful content is not mistakenly blocked. Additionally, the system is optimized for **efficiency**, offering **faster ad detection** while minimizing its impact on browser performance.

1. Document Selection

- **Ad and non-ad datasets** from sources like **IAB dataset**, web scraping results, or publicly available ad databases.
- **HTML, CSS, and JavaScript elements** that contain ad-related features, such as <iframe>, <script>, or specific class names like ad-banner.
- **URL patterns and network request logs** that help in identifying ads based on domains commonly used for advertising.
- **Existing filter lists** like **EasyList** to compare with the machine learning-based detection approach.

2. Text Summarization

For the **Ad Blocker using Random Forest**, summarization can be useful in processing **ad-related data**, filtering unnecessary content, and improving model training efficiency by focusing on **relevant ad features**. It helps in extracting meaningful insights from **large datasets, user feedback, or ad descriptions** for better classification and ad-blocking decisions.

3. Script Generation

In the context of the **Ad Blocker using Random Forest**, script generation can be used for:

- Web Scraping Scripts: Automating the collection of ad-related data from various websites for training the model.
- Feature Extraction Scripts: Generating scripts that analyze webpage elements (HTML, CSS, JavaScript) to extract ad-related features like URL patterns, <iframe>, and tracking scripts.
- Model Training and Testing Scripts: Writing Python scripts to train the Random Forest model using labeled datasets of ads and non-ads.

1.6Advantages of Proposed Systems

- **High Accuracy:** Uses the **Random Forest algorithm** for precise ad classification and blocking.
- **Adapts to New Ads:** Dynamically learns from **new ad patterns**, making it more effective over time.
- **Customizable Controls:** Allows users to **whitelist or blacklist websites** and choose which ads to block.
- **Improved Browsing Speed:** Blocks ads efficiently, leading to **faster webpage loading** and better performance.
- **Enhanced Privacy:** Prevents tracking ads from collecting **user data**, ensuring better online security.
- **Multi-Browser Compatibility:** Can be used as a **browser extension** for **Chrome, Firefox, and other popular browsers**.

2 Overall Description

2.1 Feasibility Study

A **feasibility study** evaluates whether the proposed **Ad Blocker using Random Forest** is practical and achievable. It analyzes different aspects to ensure successful implementation.

- **Technical Feasibility**

1. Uses Random Forest, a well-established machine learning algorithm, for accurate ad detection.
2. Can be implemented as a browser extension (Chrome, Firefox) or a proxy-based filtering tool.
3. Requires a labeled dataset of ad and non-ad elements for training and testing.
4. Compatible with existing ad-filtering lists (e.g., EasyList) for improved performance.

- **Economic Feasibility**

1. Low development cost, as it relies on open-source tools like Python, Scikit-Learn, and Selenium.
2. No need for expensive hardware; can run on a standard computer or cloud-based system.
3. Can be monetized through premium features (e.g., advanced filtering, AI-powered customization).

- **Operational Feasibility**

1. Enhances user experience by removing intrusive ads and improving page load speed.
2. Provides customizable ad-blocking options, allowing users to whitelist or blacklist websites.
3. Works in real-time with minimal impact on system performance.

- **Legal and Ethical Feasibility**

1. Complies with **privacy regulations** by blocking tracking ads without collecting user data.
2. Must adhere to **browser extension policies** (Google Chrome, Mozilla Firefox).
3. Ensures ethical use by allowing **users to customize their experience** without completely disabling web monetization.

2.2 Product Functionality

The **Ad Blocker using Random Forest** is designed to intelligently detect and block online advertisements, enhancing the user's browsing experience. The key functionalities of the product include:

1. **Ad Detection and Classification** – Uses the **Random Forest algorithm** to identify and classify ads based on webpage elements like HTML tags, CSS properties, and URLs.
2. **Real-Time Ad Blocking** – Automatically removes **banner ads, pop-ups, and video ads** while ensuring smooth website functionality.
3. **Customizable Ad Filtering** – Allows users to **whitelist or blacklist websites**, giving control over which ads to block or allow.
4. **Privacy Protection** – Blocks **tracking ads** to prevent advertisers from collecting user data, ensuring better online security.
5. **Multi-Browser Compatibility** – Can be implemented as a **browser extension** for **Chrome, Firefox, Edge, and other popular browsers**.

2.3 Design and Implementation Constraints

While developing the **Ad Blocker using Random Forest**, certain constraints must be considered to ensure smooth design and implementation.

1. Performance Constraints

- The system must be **lightweight** to avoid slowing down webpage loading times.
- Ad detection should be **real-time** with minimal CPU and memory usage.

2. Data Constraints

- Requires a **large dataset** of labeled ads and non-ads for training the Random Forest model.
- Webpage structures vary, making **consistent feature extraction** challenging.

3. Browser and Platform Constraints

- The ad blocker must be compatible with **multiple browsers** (Chrome, Firefox, Edge).
- Browser **security policies and extension APIs** may restrict certain functionalities.

4. Accuracy and Adaptability Constraints

- The model must be **regularly updated** to adapt to **new ad formats**.
- Needs to minimize **false positives (blocking useful content)** and **false negatives (missing ads)**.

5. Legal and Ethical Constraints

- Must comply with **browser extension policies** and **privacy laws** (e.g., GDPR).
- Should allow users **customization options** rather than enforcing complete ad blocking.

2.4 Assumptions and Dependencies

Assumptions

- Websites Use Common Ad Formats – Ads are placed using standard HTML elements like <iframe> or <script>, making them easier to detect.
- Training Data is Available – There are enough labeled datasets of ads and non-ads for training the Random Forest model.
- Users Want to Block Ads – People using this tool want an ad-free experience and will adjust settings as needed.
- Browsers Allow Ad Blocking – Modern browsers support extensions that help detect and block ads.
- Internet Connection is Needed – The tool may need updates and access to ad-blocking lists online.

Dependencies

- **Machine Learning Tools** – Uses **Scikit-Learn, Pandas, and NumPy** to train the Random Forest model.
- **Browser APIs** – Relies on **Chrome and Firefox extension APIs** to block ads.
- **Ad Filtering Lists** – Uses lists like **EasyList** to improve accuracy.
- **Regular Updates** – The system needs **ongoing training** to detect new ads.
- **User Settings** – Works based on **user choices**, such as whitelisting certain websites.

3 Functional Requirements

3.1 Software Requirement Specifications

The Adblocker Using Random Forest requires the following software components:

1.IDE: Visual Studio

2.Python: 3.8 or later

3.Framework: Flask

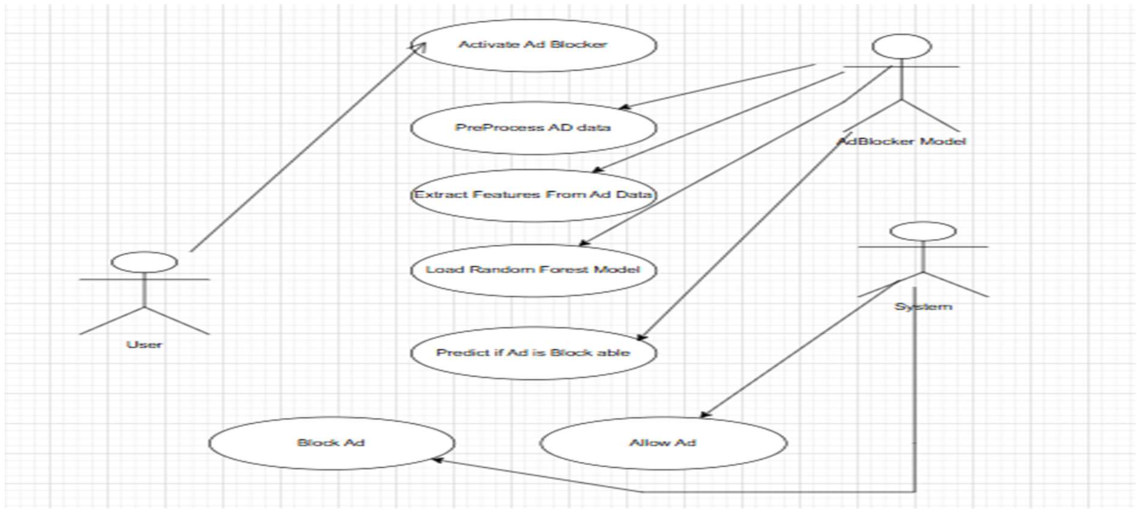
4.Web Server: XAMPP

3.2 Hardware Requirements Specifications

The Adblocker Using Random Forest Algorithm processes require:

1. RAM: 8GB and above
2. Hard disk:256GB SSD and above
3. Processor: intel i5 or i7

3.3 Use Case Model



3.3.1 Use Case

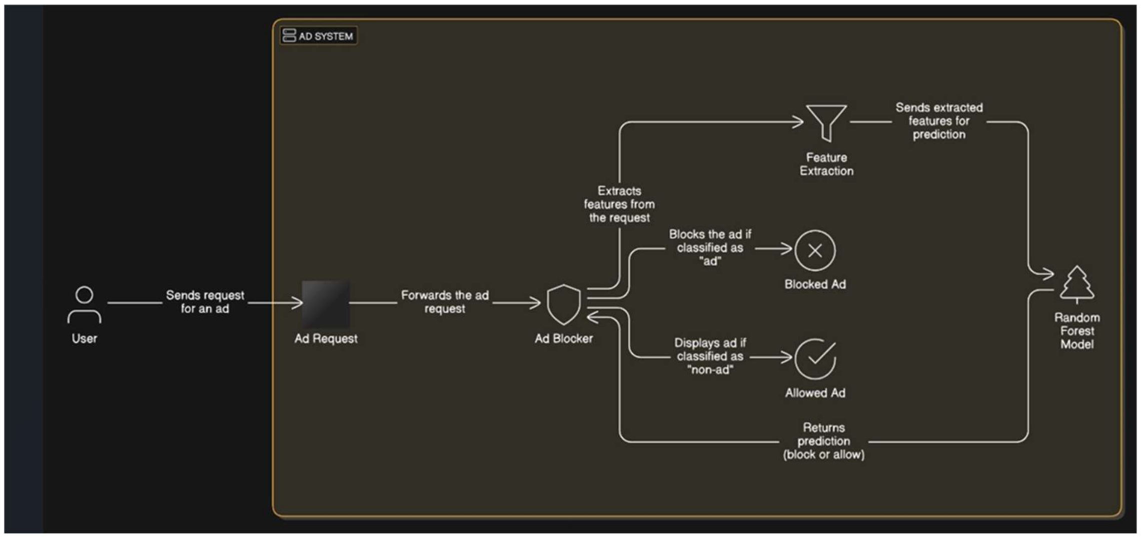
Actors:

1. **User** – The person activating the ad blocker.
2. **Ad Blocker Model** – The system that processes ad data and determines if an ad should be blocked.
3. **System** – Decides the final action: block or allow the ad.

Use Cases (Processes in the System):

1. **Activate Ad Blocker** – The user enables the ad blocker.
2. **Preprocess Ad Data** – The system processes ad-related information.
3. **Extract Features from Ad Data** – Identifies characteristics of ads for classification.
4. **Load Random Forest Model** – Loads the trained machine learning model.
5. **Predict if Ad is Blockable** – Uses the model to decide if the ad should be blocked.
6. **Block Ad / Allow Ad** – Based on the prediction, the system either **blocks** or **allows** the ad.

3.3.2 Data Flow Diagram


AD SYSTEM
User
Sends request for an ad
Ad Request
Forwards the ad request
Ad Blocker
Extracts features from the request
Feature Extraction
Sends extracted features for prediction
Random Forest Model
Blocks the ad if classified as "ad"
Blocked Ad
Displays ad if classified as "non-ad"
Allowed Ad
Returns prediction (block or allow)

4 Other Non-functional Requirements

4.1 Performance Requirements

The performance requirements ensure that the **Ad Blocker system** operates efficiently and effectively while minimizing any impact on the user experience.

1. **Fast Ad Detection & Blocking**

   The system should **detect and block ads within 100ms** to ensure a seamless browsing experience.

2. **High Accuracy of Ad Classification**

   The **Random Forest model** should achieve an **accuracy of at least 95%** in distinguishing ads from non-ad content.

3. **Low False Positive Rate**

   The system should **minimize false positives to less than 5%**, ensuring that non-ad content is not mistakenly blocked.

4. **Efficient Resource Utilization**

   The ad blocker should **consume less than 10% CPU and memory usage** to avoid slowing down the browser or system performance.

5. **Scalability & Adaptability**

   The system should be able to **handle a large number of webpages and ad formats** dynamically by updating its model regularly.

6. **Real-time Model Processing**

   - The Random Forest algorithm should process and classify ads **in real time without noticeable lag** for the user.

4.2 Safety and Security Requirements

The **Ad Blocker system** must ensure a **safe and secure** browsing experience by protecting user data, preventing malicious ads, and maintaining system integrity.

Key Safety and Security Requirements:

1. **User Data Protection**

   The system **must not store or share** any personal user data, ensuring privacy and compliance with data protection laws (e.g., GDPR, CCPA).

2. **Secure Ad Classification**

   The **Random Forest model** should accurately detect and block **malicious ads** (phishing, malware, trackers) to prevent security threats.

3. **No Unauthorized Access**

The ad blocker should have **secure access controls** to prevent third-party interference or modifications.

4. **Prevent Bypass by Advertisers**

   The system should be able to **detect and counteract** new techniques used by advertisers to evade ad blocking.

5. **Minimal Performance Risk**

   The ad blocker should not **crash the browser** or cause system instability due to excessive resource usage.

6. **Secure Software Updates**

   Updates should be securely delivered with **integrity checks** to prevent malicious code injection.

7. **Protection Against Code Injection**

   The system should be resistant to **code injection attacks** that attempt to disable the ad blocker or introduce vulnerabilities.

8. **Safe Handling of Scripts and Web Requests**

   The ad blocker should **analyze and block harmful scripts** while ensuring legitimate website functions are not disrupted.

4.3 Software Quality Attributes

The **Ad Blocker system** must meet various **software quality attributes** to ensure efficiency, security, and user satisfaction. Below are the key attributes:

---

1. Performance

- The system should **detect and block ads within 100ms** to maintain a seamless browsing experience.
- **Minimal CPU and memory usage** (≤10%) to avoid slowing down the system.

2. Accuracy & Reliability

- The **Random Forest model** must achieve at least **95% accuracy** in identifying ads.
- False positives should be **less than 5%** to prevent blocking useful content.

3. Security

- Prevent **malicious ads, trackers, and phishing attempts** from loading.
- Ensure **secure software updates** to prevent unauthorized modifications.

4. Usability

- The interface should be **easy to use** with simple ad-blocking controls.

- Users should have options to **customize blocking preferences** (e.g., allow specific ads).

5. Scalability

- The system should **adapt to new ad formats and evolving advertising techniques**.
- Capable of handling a **large volume of web traffic** without performance degradation.

6. Maintainability

- The codebase should be **modular and well-documented** for easy updates.
- Future improvements should be **easily integrated** without affecting performance.

7. Compatibility

- The ad blocker should work **across multiple browsers** (Chrome, Firefox, Edge, etc.).
- Compatible with **various website structures** without breaking essential functionality.

8. Robustness & Fault Tolerance

- The system should handle **unexpected failures gracefully** (e.g., network issues, script errors).
- In case of failure, **it should not crash the browser** or disrupt the user experience.

9. Efficiency

- The system should efficiently **process web pages and classify ads in real-time**.
- Should consume minimal **bandwidth and computing resources**.

10. Customizability

- Users should be able to **whitelist or blacklist specific websites**.
- **Adjustable filtering levels** (e.g., strict, moderate, lenient).

- Users should have options to **customize blocking preferences** (e.g., allow specific ads).

5. Scalability

- The system should **adapt to new ad formats and evolving advertising techniques**.
- Capable of handling a **large volume of web traffic** without performance degradation.

6. Maintainability

- The codebase should be **modular and well-documented** for easy updates.
- Future improvements should be **easily integrated** without affecting performance.

7. Compatibility

- The ad blocker should work **across multiple browsers** (Chrome, Firefox, Edge, etc.).
- Compatible with **various website structures** without breaking essential functionality.

8. Robustness & Fault Tolerance

- The system should handle **unexpected failures gracefully** (e.g., network issues, script errors).
- In case of failure, **it should not crash the browser** or disrupt the user experience.

9. Efficiency

- The system should efficiently **process web pages and classify ads in real-time**.
- Should consume minimal **bandwidth and computing resources**.

10. Customizability

- Users should be able to **whitelist or blacklist specific websites**.
- **Adjustable filtering levels** (e.g., strict, moderate, lenient).

6 References

[1]. K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie. P-Scout: Analyzing the Android Permission

Specification. In CCS, (2021).

[2]. Android Permissions.

http://developer.android.com/guide/topics/security/permissions.html.

[3]. M. Ikram, N. V. Rodriguez, S. Seneviratne, D. Kaafar, and V. Paxson. An analysis of the privacy

and security risks of android VPN permission-enabled apps. In ACM IMC, (2022).

[4]. A. Lerner, A. K. Simpson, T. Kohno, and F. Roesner. Internet jones and the raiders

SRS DOCUMENT REVIEW

CERTIFICATION

This Software Requirement Specification (SRS) Document is reviewed and certified to proceed for the project development by the Departmental Review Committee (DRC).

Date of SRS Submitted:	
Date of Review:	
Supervisor Comments:	
Supervisor Sign. & Date.	
Coordinator Sign. & Date	
HOD Sign. & Date	
Dept. Stamp	