CS Senior Design Specification Report

NOVA Browser

Roshni Varkey, Braden Norum
Faculty Advisor: Aman Luthra

Spring 2025
May 16, 2025

Abstract

The NOVA Browser project addresses escalating concerns over digital privacy and user data security in contemporary internet usage. Existing browsers offer varying degrees of privacy protection but often compromise functionality or fail to integrate advanced technologies effectively. NOVA Browser, built upon Firefox's modular extension architecture, proposes an innovative solution that prioritizes comprehensive privacy without sacrificing user experience. It incorporates advanced privacy features, including a secure incognito mode, integrated ad-blocking, local data processing for AI-driven functionalities, and encrypted data management. Additionally, NOVA offers customizable user profiles, robust plugin management, real-time network monitoring tools, and seamless cross-platform compatibility. The prototype successfully demonstrates enhanced security and usability, leveraging Firefox's robust security infrastructure and Gecko rendering engine. Testing confirms the browser's effectiveness in protecting user privacy, blocking intrusive tracking mechanisms, and providing intuitive and high-performance browsing experiences, effectively setting a new standard in privacy-centric browsing.

Chapter 1: Project Scope and Vision

Section 1.1: Project Scope and Vision

In today’s digital landscape, privacy has become a paramount concern. With most web browsers collecting extensive user data for targeted advertisements and analytics, individuals are increasingly vulnerable to data exploitation and cyber threats [6], [7]. Furthermore, the growing complexities of privacy laws like GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act) place significant pressure on both users and service providers to ensure compliance [6], [7]. Traditional privacy measures, such as incognito modes, often offer a false sense of security, as they primarily prevent local data storage without addressing external tracking mechanisms [11]. This project seeks to redefine the browsing experience by emphasizing true user privacy and security.

Several existing solutions attempt to address these challenges but have notable limitations. Google Chrome, while user-friendly, offers a privacy mode that is insufficient against advanced tracking by advertisers and websites [10]. Firefox and Brave take stronger stances on privacy with features like enhanced tracking protection and ad-blockers [10]. However, these browsers lack robust integration of AI-powered tools, which limits their ability to provide a personalized yet secure experience [8]. DuckDuckGo offers a highly privacy-focused environment but sacrifices some functionality, leaving advanced users wanting more [8]. Each of these solutions addresses specific aspects of the privacy problem but fails to combine robust privacy with cutting-edge features in a unified platform.

Our vision is to create a browser prototype that goes beyond these limitations. This browser will prioritize privacy by default, integrating features like a secure incognito mode, powerful ad-blocking, and encrypted local data processing for AI-driven functionalities [1], [2]. Users will also benefit from customizable profiles, plugin management, and advanced network analysis tools [10], [11]. By seamlessly blending user-centric AI capabilities with top-tier privacy protections, our solution will stand apart from competitors. It will empower users to browse confidently, free from intrusive tracking and security risks, while delivering a fast, intuitive, and innovative browsing experience across platforms.

Our browser addresses increasing societal concerns about digital privacy and surveillance. Nova seeks to empower users with control over their personal information by minimizing data collection, blocking trackers, and offering secure AI powered features locally. There has been an increase in the demand for privacy respecting technology which helps mitigate risks like identity theft and data misuse which are ever growing concerns in our modern society. Our tool also promotes ethical computing by integrating privacy by design.

We adopted a modular Firefox-extension-based design allowing us to leverage Firefox's security infrastructure while being able to customize privacy and performance features on top. This approach was chosen over building a browser from scratch to maximize stability, minimize security risks, and accelerate development.

We successfully delivered a functional prototype that includes a true incognito mode, advanced ad and tracker blocking, customizable plugin support, and a built-in AI assistant while maintaining fast, secure, cross-platform performance. The project met all wanted design goals, and testing confirmed usability, security, and compatibility across platforms.

Chapter 2: Project Goals

Section 2.1: Project Goals

1. **Develop a Privacy-Centric Browser**: Build a web browser that prioritizes user privacy with robust incognito mode functionality, ensuring no data is stored during private sessions.

2. **Implement AI-Powered Features**: Seamlessly integrate AI capabilities, including real-time search suggestions and personalized user experiences, all while processing data locally to maintain privacy.

3. **Incorporate Advanced Plugin Management**: Create a secure plugin management system that allows users to install, enable, and customize plugins, including those for email, themes, and AI-driven enhancements.

4. **Enable Network Analysis Tools**: Integrate diagnostic tools like Wireshark to allow users to monitor and analyze real-time network traffic for advanced security and performance insights.

5. **Enhance Security and Ad-Blocking**: Provide built-in ad-blocking functionality to prevent intrusive ads and trackers, complemented by regular security updates derived from Firefox's patching system.

6. **Support Custom User Profiles**: Enable users to create and manage multiple profiles with personalized settings, themes, and preferences to enhance the browsing experience.

7. **Ensure Multi-Platform Compatibility**: Deliver a consistent and seamless browsing experience across Linux, Windows, and macOS platforms, ensuring accessibility for all

users.

8. **Focus on Performance and Stability**: Release a stable, fully tested browser version with consistent updates to ensure optimal performance and reliability.

9. **Leverage Firefox Foundation**: Utilize Firefox's security updates and underlying Gecko engine for a solid foundation while enhancing features for modern user needs.

Chapter 3: Requirement Specification

Section 3.1 System Perspective

This section provides an overview of the browser's interaction with users and external systems. Key privacy tools include an incognito mode that prevents data storage during private sessions and a built-in ad-blocker that blocks intrusive ads, pop-ups, and tracking scripts. Customization options allow users to tailor their browsing experience through personalized profiles, themes, and settings, as well as custom commands for efficient navigation. AI-powered search suggestions provide enhanced insights, while plugin support extends functionality, enabling users to securely install and manage additional tools, such as email integrations. For technically inclined users, a packet tracing monitor offers real-time network analysis, adding an extra layer of security awareness.

On the technical side, the browser ensures a smooth interaction with external systems by effectively rendering web pages using HTML, CSS, and JavaScript. It incorporates Firefox's regular security updates to strengthen its defense against vulnerabilities. Diagnostic tools, such as network traffic monitoring, give users insights into their online activity, empowering them to detect and mitigate potential risks. Additionally, the browser delivers a consistent experience across Linux, Windows, and macOS platforms, ensuring accessibility for a wide range of users. Combining cutting-edge privacy protections with intuitive design and powerful features, the browser aims to redefine modern web browsing by addressing both everyday needs and advanced use cases.

Section 3.2. User Stories

This section details the interactions between users and the browser through user stories, which represent specific scenarios to demonstrate the browser's functionality. Each user story is structured to include a rationale, priority, and cross-references to relevant user interfaces (UIs) and functional requirements (FRs). The following table outlines the user stories for this project, describing key actions like enabling incognito mode, blocking ads, setting custom commands, and managing plugins.

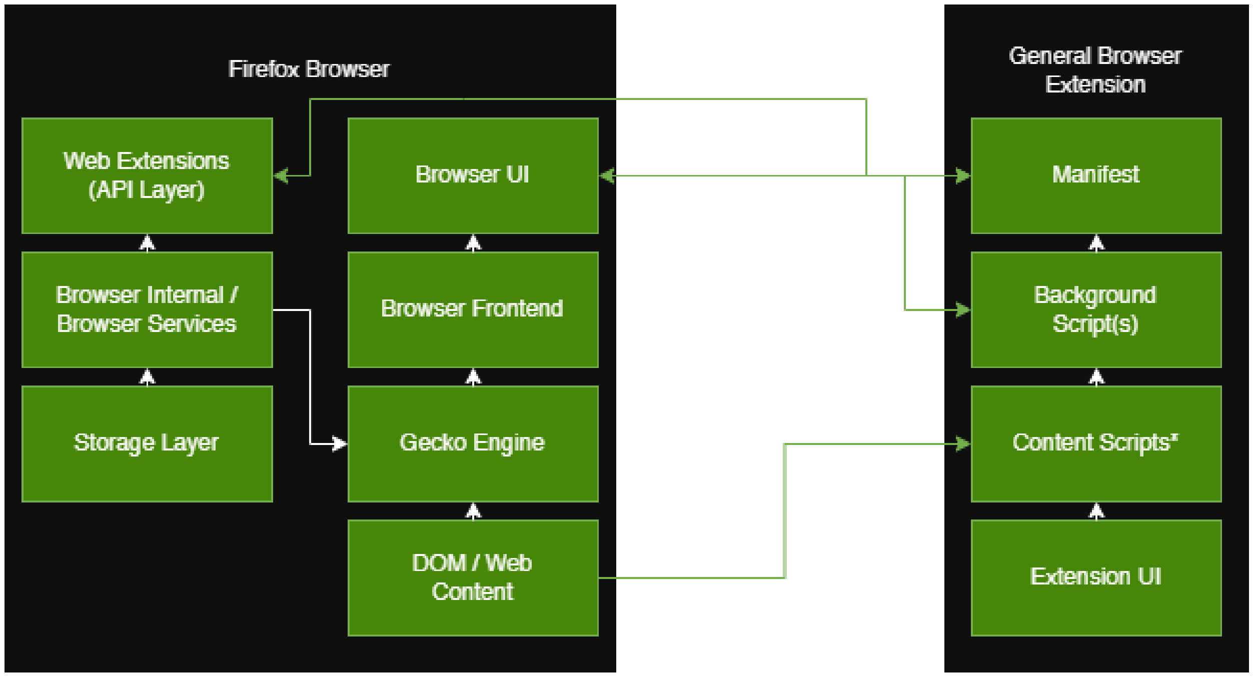


Figure 3.1 - System Design Diagram

Table 3.2.1: Activating Incognito Mode

Name	US-1 Activating Incognito Mode
Description	Allows users to activate a fully incognito browsing session
Rationale	To allow users to browse the internet without having to leave a trace or expose personal information.
Users	All Users
Step-by-step description	1. User clicks on browser 2. System opens a new window with no tracking, cookies, or data collection 3. User is able to browse internet securely 4. System blocks any scripts, trackers, and advertisements while anonymizing requests.
Priority	5
Story Points	8

References	UI-3, FR-2

Table 3.2.2: Enable Ad-Blocker

Name	US-2 Enable Ad-Blocker
Description	Blocks Advertisements and trackers to enhance privacy
Rationale	To protect users from intrusive ads and unnecessary tracking
Users	All Users
Step-by-step description	1. User clicks on browser 2. Built in ad-blocker is activated 3. System blocks advertisements, pop-ups, and tracking scripts across websites. 4. System displays a counter showing the number of ads blocked
Priority	4
Story Points	7
References	UI-7, FR-5

Table 3.2.3: Set Custom Commands

Name	US-3 Set Custom Commands
Description	Allows users to set custom commands to control browser behavior via shortcuts
Rationale	Allows users to navigate the browser more efficiently
Users	All Users, Technical Background
Step-by-step description	1. User accesses custom command settings 2. System allows the user to define specific actions for key combinations 3. User saves the custom command settings. 4. System executes these commands instantly during future use.
Priority	1
Story Points	5

References	UI-5, FR-1

Table 3.2.4: AI-Powered Chat

Name	US-4 AI-Powered Chat
Description	Provides AI-driven search suggestions
Rationale	Allows users to gain more insight into their search results
Users	All Users
Step-by-step description	1. The user activates the AI assistant via the browser's toolbar or command interface. 2. The assistant opens a chat interface embedded in the browser or sidebar. 3. The user enters a query (e.g., "Summarize this page," "Compare alternatives to X," "What is Y?"). 4. The assistant processes the request using a local or secure AI model (e.g., gemma). 5. The assistant responds with helpful insights, links, or summaries while respecting local privacy constraints (no data leaves the browser). 6. Users can optionally ask follow-up questions or issue voice commands, if enabled. 7. The chat window remains accessible while browsing or can be minimized.
Priority	3
Story Points	8
References	UI-5, UI-8, FR-1

Table 3.2.5: Plug-in Support

Name	US-5 Plug-in support
Description	Allow users to install and manage plugins for browser and email
Rationale	To extend the functionality of the browser and allow users to personalize their experience
Users	All Users
Step-by-step description	1. User accesses a plugin management tool 2. System displays a list of available plugins (e.g. email integration) 3. User installs desired plugins. 4. System runs each plugin in a secure sandboxed environment to ensure user privacy.
Priority	4
Story Points	6
References	UI-7, FR-3

Table 3.2.6: Packet Tracing Monitor

Name	US-6 Packet Tracing Monitor
Description	All users to monitor and analyze network traffic.
Rationale	To provide users with insights into their network activity and detect potential security threats
Users	All Users, Technical Background
Step-by-step description	1. User enables network monitoring in the browser settings. 2. System integrates dev tools networking to capture and display network traffic in real-time. 3. System enables this in a window.
Priority	3
Story Points	4
References	UI-6, FR-6

Table 3.2.7: Stable Release

Name	US-7 Stable Release
Description	Ensures stable releases and frequent updates to address performance improvements and security patches.
Rationale	To provide a secure and consistently stable browsing experience.
Users	All Users
Step-by-step description	1. User receives a notification for a new update and clicks on it. 2. System downloads and installs the update. 3. System automatically applies updates and informs the user of improvements.
Priority	1
Story Points	2
References	UI-4

Table 3.2.8: Incorporate Firefox Security Updates

Name	US-8 Incorporate Firefox Security Updates
Description	Integrates firefox security updates to ensure the browser is protected against vulnerabilities and exploits
Rationale	To stay up-to-date with the latest security standards and protect users from known threats.
Users	All Users
Step-by-step description	1. System checks for new Firefox security updates. 2. User is notified of the updates. 3. System applies the security patches automatically. 4. User benefits from improved security without manual intervention.
Priority	5
Story Points	4
References	UI-3, FR-2

Table 3.2.9: Multiplatform Support

Name	US-9 Multiplatform Support
Description	Provides consistent browsing experience across Linux, Windows, and Mac operating systems.
Rationale	To ensure users can access the browser with the same features and performance regardless of the platform.
Users	All Users
Step-by-step description	1. User installs the browser on their chosen platform (Linux, Windows, or Mac). 2. System provides a consistent UI and feature set across all platforms.
Priority	1
Story Points	2
References	FR-8

Table 3.2.10: User Profiles and Preferences.

Name	US-10 User Profiles and preferences.
Description	Provides users with customizable settings in different profiles on the same system.
Rationale	To provide to users the ability to use the browser in the way easiest for them.
Users	All Users
Step-by-step description	1. User opens preference menu. 2. User is presented with several options which change the look and functionality of the browser. 3. After selecting settings, these changes are reflected in further browsing.

Priority	4
Story Points	2
References	UI-1, UI-2, FR-4

Section 3.3 External Interfaces:

The system's interaction with users and external components relies on user interfaces (UIs) and a range of input and output mechanisms. Each interface is carefully designed to enhance usability, functionality, and accessibility. Figure 1-8 corresponds to a UI.

Section 3.3.1 User interfaces:

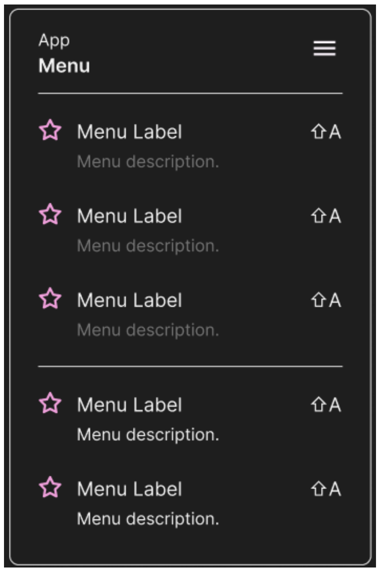


Figure 1:UI-1 Settings Dropdown

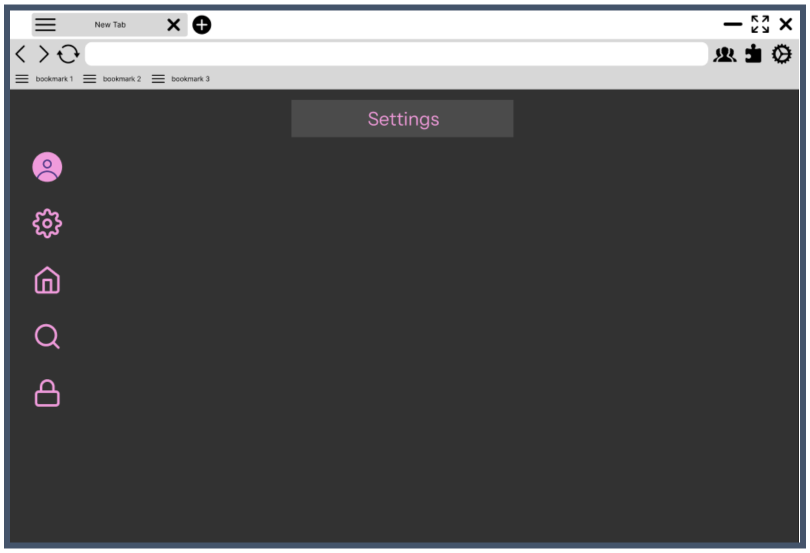


Figure 2: UI-2 Settings Page

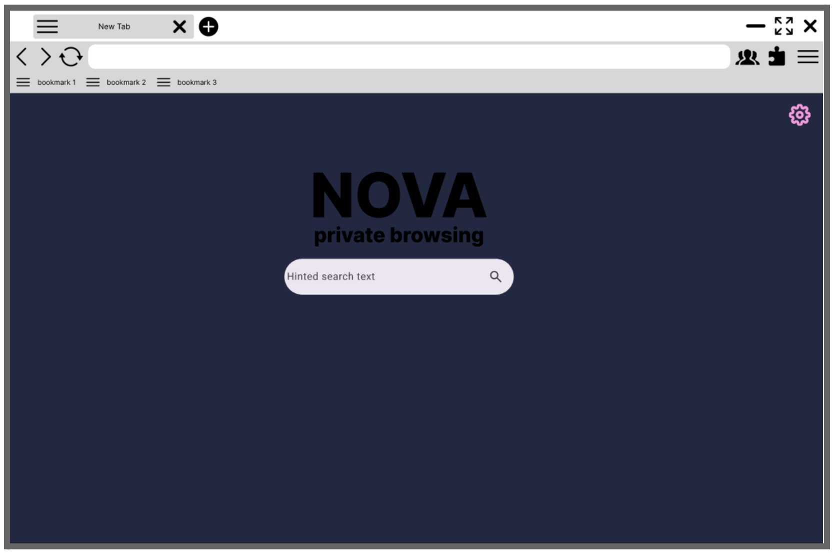


Figure 3: UI-3 Private Browsing Home

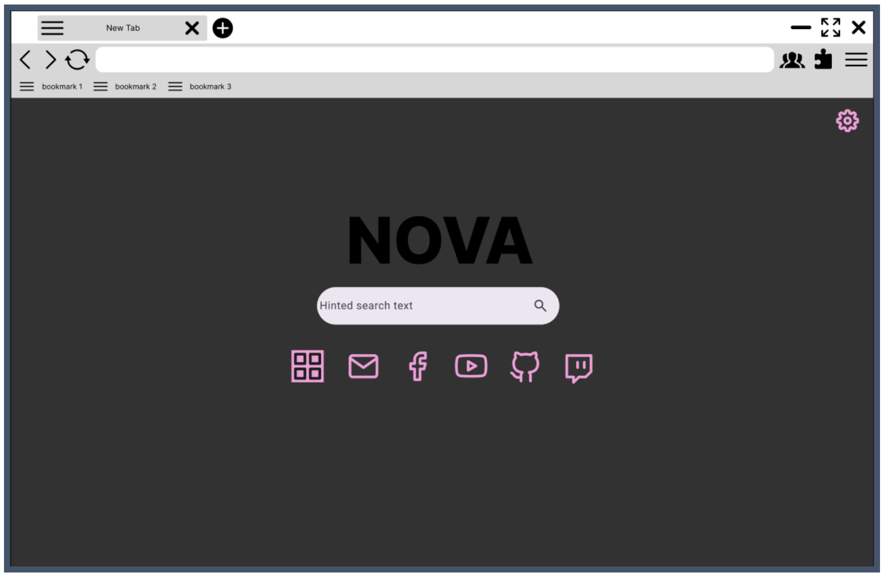


Figure 4: UI-4 Standard Browsing Home

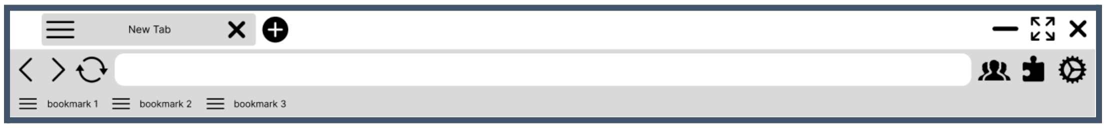
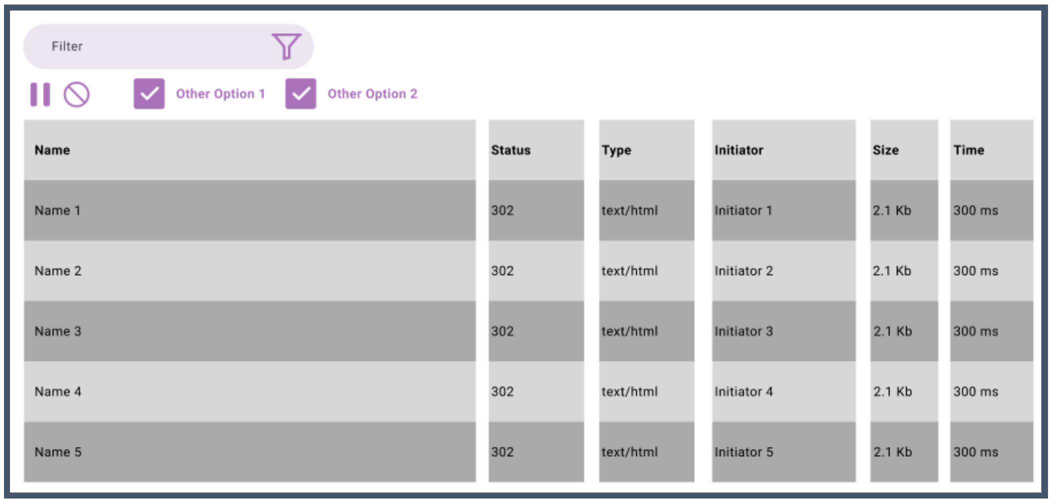


Figure 5:UI-5 Top Tab/Search Bar



Name	Status	Type	Initiator	Size	Time
Name 1	302	text/html	Initiator 1	2.1 Kb	300 ms
Name 2	302	text/html	Initiator 2	2.1 Kb	300 ms
Name 3	302	text/html	Initiator 3	2.1 Kb	300 ms
Name 4	302	text/html	Initiator 4	2.1 Kb	300 ms
Name 5	302	text/html	Initiator 5	2.1 Kb	300 ms

Figure 6: UI-6 Network Traffic Monitor Page

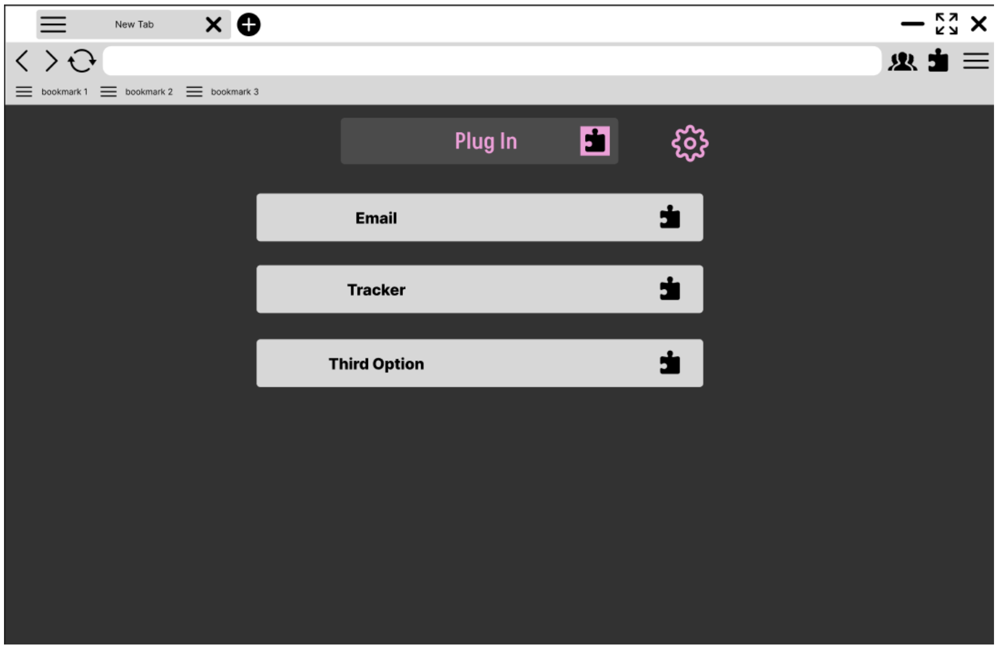
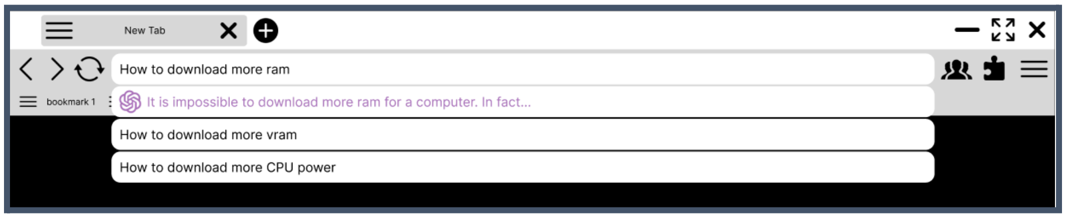


Figure 7: UI-7 Plug-In Enabler



How to download more ram

It is impossible to download more ram for a computer. In fact...

How to download more vram

How to download more CPU power

Figure 8: UI-8 AI Search Option

The table below describes the primary user interfaces, detailing their purpose and connection to relevant user stories.

Table 3.3.1: User Interfaces

UI Name and Number	Short description	Reference user stories
UI-1 Settings Dropdown	Dropdown accessible from any page in the browser, allows users to open settings, preferences, etc.	US-10
UI-2 Setting Page	Allows users to edit the settings of the browser.	US-10

UI-3 Private Browsing Home	Home page for browsing with telemetry and cookies disabled.	US-1, US-8
UI-4 Standard Browsing Home	Home page for standard browsing without security features.	US-7
UI-5 Top Tab / Search Bar	Navigation between pages of the browser.	US-3, US-4
UI-6 Webshark Page	List of incoming / outgoing connections, accessible by the settings dropdown.	US-6
UI-7 Plugin Enabler	Contains a list of installed plug-ins, with switches to enable and disable.	US-2, US-5
UI-8 AI Search Option	When searching, will appear instead of an auto-fill search below the bar.	US-4

Table 3.3.2: Inputs

Input # and name	Source	Description	Units of measure	Data format	References to user stories, etc.
I1- Search	Text Field	Enter search keywords	text length	string	US-1,US-4, UI-3,UI-4, UI-5, UI-8
I2-Back Button	Button	Navigates to previous page	NA	NA	UI-8
I3-Forward Button	Button	Navigates to next page	NA	NA	UI-8
I4-Refresh Button	Button	Reloads selected page	NA	NA	UI-8
I5-Close Tab	Button	Closes the current tab	NA	NA	UI-8
I6-New Tab	Button	Opens a new tab	NA	NA	UI-8
I7- Users Button	Button	Opens up a user selection	NA	NA	US-10,UI-8, UI-2

		menu			
I8-Plug-in Button	Button	Opens the plug-in support page	NA	NA	UI-8, UI-7
I9-Menu	Button	Menu icon, opens settings / preferences dropdown	NA	NA	UI-8
I10- Minimize	Button	Minimizes browser	NA	NA	US-3, UI-8
I11 Maximize	Button	Maximizes browser	NA	NA	US-3, UI-8
I12-Settings	Button	Settings icon, opens settings	NA	NA	UI-8, UI-2
I13-Home	Button	Home Icon, opens home settings	NA	NA	US-10, UI-2
I14-Privacy	Button	Lock Icon, opens privacy settings	NA	NA	US-10, UI-2
I15-Start Trace	Button	Starts Wireshark network tracing	NA	NA	US-6, UI-6
I16-End Trace	Button	Ends Wireshark network tracing	NA	NA	US-6, UI-6

Table 3.3.2: Outputs

Output # and name	Destination	Valid range, accuracy, and/or tolerance	Units of measure	Data formats	References to user stories, etc.
O-1 Search	Search Engine Webpage	< 100 char	# char	text	US-1, UI-3, UI-4, UI-5
O-2 Back	Previous Webpage	<100 milliseconds only works if there was a previous page	milliseconds	Button (icon)	US-3, UI-5
O-3 Forward	Next Webpage	<100 milliseconds	milliseconds	Button (icon)	US-3,UI-5

		only works if there is a next page			
O-4 Refresh	Same Webpage	<100 milliseconds	milliseconds	Button (icon)	US-3,UI-5
O-5 Close Tab	N/A	<100 milliseconds	milliseconds	Button (icon)	US-3, UI-5
O-6 New Tab	a new browsing tab	<100 milliseconds	milliseconds	Button (icon)	US-3, UI-5
O-7 Users	User profile/preferences page	<100 milliseconds	milliseconds	Button (icon)	US-3, US-10, UI-5
O-8 Plug-ins	Plug in page	<100 milliseconds	milliseconds	Button (icon)	US-3, UI-5, UI-7

Section 3.4 Functional Requirements

This section details the functional requirements for the privacy-focused browser project. The tables below organize these requirements, outlining their purpose, behavior, and cross-references to user stories and user interfaces. **Tables 3.1 to 3.8** list the primary functionalities the browser must implement, including rendering web pages, activating incognito mode, managing plugins, and more. Each table provides a concise description of the requirement, its rationale, system behavior, and related system components

Table 3.1: FR-1 Render Webpage

Name	FR-1 Render Webpage
Description	The system retrieves and renders web pages using HTML, CSS, and JavaScript.
Rationale	To provide the user with a display of web content
System Behavior	● User enters a URL in the address bar. ● System sends an HTTP/HTTPS request to the web server ● Server responds with web page data (HTML, CSS, JS). ● System processes and renders the page for the user to view and interact with.
Cross-Reference	UI-4, UI-3, UI-5, US-3, US-4

Table 3.2: FR-2 Incognito Mode Activation

Name	FR-2 Incognito Mode Activation
Description	The system activates a private browsing session where no data like history or cookies is stored.
Rationale	To protect user privacy by ensuring that no browsing data is retained.
System Behavior	● User selects Incognito Mode and a new window is opened ● System disables cookies, history logging, and cache storage. ● All browsing in this session remains private until the session is closed. ● On closing the session, all session data is erased.
Cross-Reference	UI-3, US-1, US-8

Table 3.3: FR-3 Plugin Installation and Management

Name	FR-3 Plugin Installation and Management
Description	The system allows users to install, enable, disable and manage plugins
Rationale	To extend browser functionality
System Behavior	● user accessed plugin management page ● User selects a plug in to install,enable, or disable ● System installs/modifies plugin status
Cross-Reference	US-5 UI-7

Table 3.4: FR-4 Handle User Preferences

Name	FR-4 Handle User Preferences
Description	The system must allow users to customize their experience
Rationale	To provide personalized browsing experiences
System Behavior	● The user creates a profile ● the system stores browsing history, preferences, settings

	● User adjusts appearance and privacy settings
Cross-Reference	US-10, UI-1, UI-2

Table 3.5: FR-5 Ad-Blocker

Name	FR-5 Ad-Blocker
Description	The system blocks ads and tracking scripts by default.
Rationale	To protect user privacy.
System Behavior	● User enables ad-blocking. ● System blocks ads and tracking scripts on all web pages. ● System displays number of ads blocked.
Cross-Reference	US-2, UI-7

Table 3.6: FR-6 Monitor Network Traffic (Webshark Integration)

Name	FR-6 Monitor Network Traffic (Webshark Integration)
Description	The system tracks and logs network traffic going in and out of the browser
Rationale	To provide information for users interested in cataloging network traffic.
System Behavior	● User opens networking tab. ● System prompts for a refresh to get network traffic from loading the site. ● Network traffic appears in a list, detailing status, type, time, etc.
Cross-Reference	US-6, UI-6

Table 3.7: FR-7 Bookmark Management

Name	FR-7 Bookmark Management
Description	The system allows users to save and catalogue links to websites

Rationale	To give users ease of access to browsing commonly viewed sites
System Behavior	● user saves webpage as a bookmark ● system organizes bookmarks into folder
Cross-Reference	UI-5

Table 3.8: FR-8 Multi-Platform Support

Name	FR-8 Multi-Platform Support
Description	Multiple systems are able to run the application natively.
Rationale	To allow a multitude of users to use the browser.
System Behavior	● User installs system on their preferred platform ● User launches browser ● System runs natively on platform
Cross-Reference	US-9

Section 3.5. Non-functional Requirements

Section 3.5.1. Performance requirements

PR1: Page Load Time

- Description: The browser must load standard web pages within 5 seconds over a standard broadband connection (5 Mbps) .
- Measurable: Pages will load 95% of the time within 5 seconds on a stable internet connection.

PR2: Number of Simultaneous Open Tabs

- Description: The browser must handle at least 15 simultaneous tabs open without significant performance degradation (less than 10% increase in response time).
- Measurable: Browser response time and memory usage will be monitored when 15 tabs are open.

PR3: Ad-Blocking Performance

- Description: The ad-blocker should block 95% of intrusive ads and tracking scripts with less than 200 ms delay in page load time.
- Measurable: The system will measure the delay caused by ad-blocker and verify blocked content.

PR4: AI Chat Response Time

- Description: The AI-powered chat bar should return results within 12 seconds of user input.
- Measurable: The time between entering a query and receiving search suggestions will be tested.

Section 3.5.2. Logical database requirements

DR1: User Preferences Storage

- Description: The system must store user preferences (e.g., themes, privacy settings, search engine preferences) locally and sync them across devices.
- Frequency of Use: Every time the user updates preferences or when syncing occurs.
- Data Entities: User profile, preferences, bookmarks, browsing history.

DR2: Bookmark and History Storage

- Description: The system must store user bookmarks and browsing history with a capacity of 10 entries without performance degradation.
- Accessing Capabilities: The user should be able to retrieve bookmarks and browsing history within 5 seconds.

Section 3.5.3. Engineering requirements

ER1: Platform Compatibility

- Description: The browser must be compatible with Windows, macOS, and Linux platforms.
- Software Requirements: Compatible with OS versions (e.g., Windows 10+, macOS 10.14+, Ubuntu 18.04+).

ER2: Minimum Network Speed

- Description: The browser must function properly on internet speeds of 1 Mbps and above.

ER3: Hardware Requirements

- Description: The browser should use no more than 2 GB of VRAM when utilizing AI Chat, and no more than 2 GB of RAM.

Section 3.5.4. Security requirements

Confidentiality

- Encryption: All sensitive data, such as passwords and cookies, will be encrypted both at rest and in transit. Data in transit will utilize SSL/TLS protocols to ensure secure communication, while locally stored data will use robust encryption methods like AES or scrypt hashing.
- Masking: Sensitive information displayed to users, such as partially obscured passwords, will ensure that critical data remains hidden from unauthorized individuals.

Integrity

- Data Reliability and Accuracy: The browser will maintain data integrity by ensuring all modifications are conducted by authorized users. This will be achieved through hashing mechanisms and digital signatures to verify data accuracy and authenticity.
- Testing for Vulnerabilities: Regular functionality tests and audits will verify the reliability of the browser's data handling features.

Availability

- Protection Against Service Disruption: Measures will be implemented to protect the browser from unwanted downtime. This includes defining a Maximum Tolerable Downtime (MTD) and setting a Recovery Time Objective (RTO) to restore service in the event of disruptions.

Authentication, Authorization, and Accountability (AAA)

- Authentication: The browser will ensure the legitimacy of user identities through multi-factor authentication methods such as two-factor authentication (2FA), biometric verification, or single sign-on (SSO).
- Authorization: Permissions will be assigned based on roles, ensuring only authorized users can access sensitive data or functionality.
- Accountability: User actions will be logged to detect unauthorized access or changes. These logs will be auditable to maintain transparency and security compliance.

Input Validation

- Sanitization Against Threats: The browser will validate all user inputs to prevent attacks such as SQL injection and other malicious input-based exploits. Input validation will be tested through penetration testing to ensure robust security.

Specific Security Requirements

1. SR1: SSL/TLS Encryption
    - Description: All communication between the browser and websites must use SSL/TLS encryption to secure data in transit.

- Measurable: HTTPS connections will be enforced, and users will be warned about insecure HTTP connections.

2. SR2: Data Encryption
   - Description: All sensitive data, such as passwords and cookies, stored locally must be encrypted.
   - Measurable: Encryption methods will undergo testing and audits to ensure robustness and compliance with standards.
3. SR3: Input Validation
   - Description: The browser must validate all user inputs to prevent vulnerabilities like SQL injection and related attacks.
   - Measurable: Input validation mechanisms will be rigorously tested using automated and manual penetration tests.

Section 3.5.5. Reliability Requirements

RelR1: System Uptime

- Description: The browser must maintain 99.9% uptime, ensuring it is available for use at all times except during updates.
- Measurable: Uptime will be tracked and should not fall below this threshold over any 30-day period.

RelR2: Crash Recovery

- Description: The browser must recover all open tabs and session data in the event of a crash.
- Measurable: In the case of a system failure, session recovery should occur within 5 seconds of restarting the browser.

Section 3.5.6. Robustness

RobR1: Error Handling

- Description: The browser must handle errors (e.g., page load failure, network issues) gracefully by providing informative feedback to the user and attempting automatic recovery.
- Measurable: Errors must be presented to the user within 1 second of occurrence, and automatic retries should be attempted for network errors.

RobR2: Resource Management

- Description: The browser should handle high memory usage (e.g., multiple tabs, plugins) without crashing or significantly degrading performance.
- Measurable: When memory usage reaches 80%, the browser will warn users and automatically manage resources (e.g., suspend inactive tabs).

Chapter 4: Standards and Constraints

This chapter outlines the engineering standards and constraints that guide the development of the browser. The browser must adhere to privacy, legal, ethical, and design principles to deliver a secure, user-friendly experience. The following sections elaborate on the standards and constraints impacting the project.

Section 4.1 Standards:

Section 4.1.1 Security Standards

As our browser seeks to implement true privacy and security, we must adhere to strict security standards. To protect user data, the browser will implement TLS 1.3 for encrypted communication and hashing for password security. The design will adhere to IEEE Standard 1363.2-2008 for cryptographic protocols ensuring secure data transmission [1], [2].

Section 4.1.2 Ethical Standards

The browser will adhere to ethical guidelines to protect user privacy and data security. This includes compliance with the IEEE Code of Ethics and ACM Code of Ethics, ensuring transparency and fairness in data handling. The browser will not collect, sell, or share user data without explicit consent, prioritizing privacy as a core value. Ethical considerations also shape the design of features like incognito mode, encryption, and data handling practices to ensure users' trust [3], [4].

Section 4.1.3 Legal Standards

The browser must comply with data protection regulations such as GDPR (Europe) and CCPA (California). These laws require that users are informed about data collection and allowed to control their data. Additionally, If the browser uses or builds upon open-source software, it must comply with the relevant licenses (e.g., GPL, MIT) and ensure no violations of intellectual property laws. Legal compliance ensures secure, transparent, and lawful data processing practices [5], [6].

Section 4.1.4 Ergonomic and Aesthetic Standards

The browser must be intuitive, with clear controls and accessible features. The layout of buttons, menus, and tabs should follow ergonomic principles for ease of use. The browser must have customizable themes and a modern design to appeal to users, while also supporting accessibility features like high contrast and large text [7].

Section 4.1.5 Development Standards

The browser's code will comply with W3C HTML5 Standards for web content rendering and follow ISO/IEC 25010 guidelines for system quality attributes. These standards ensure compatibility, reliability, and performance in the browser's core functionalities [8], [9].

Section 4.2 Constraints

Section 4.2.1 Ethical Constraints

Privacy is paramount; no unauthorized data collection or sharing will occur. Explicit user consent will be required for any data-handling operations. Ethical constraints directly shape privacy controls, encryption methods, and the incognito mode [4], [6].

Section 4.2.2 Legal Constraints

Compliance with GDPR, CCPA, and other regional data laws influences the browser's data storage and notification features. Open-source licensing agreements are adhered to strictly to avoid legal repercussions [6], [7].

Section 4.2.3 Economic Constraints

There is a time constraint as the project must be completed by May 2025.

Section 4.2.4 Ergonomic and Aesthetic Constraints

The browser prioritizes user-friendly design, featuring intuitive controls and customizable themes. Accessibility options ensure usability for all users, including those with disabilities [10].

Section 4.2.5 Social and Political Constraints

The removal of telemetry enhances privacy but may limit user experience on platforms like Facebook, which rely on tailored data. The browser must comply with data sovereignty laws to ensure secure operation in different regions [7].

Section 4.2.6 Environmental and Sustainability Constraints

Energy-efficient coding practices are adopted to minimize resource usage, particularly on mobile devices. Sustainability is enhanced through the use of open-source tools, ensuring updates can be implemented with minimal resource expenditure.

Section 4.2.7 Health Constraints

There are no health constraints relevant to this project.

Section 4.2.8 Welfare Constraints

There are no welfare constraints relevant to this project.

Section 4.2.9 Global Constraints

There are no global constraints relevant to this project.

Section 4.2.10 Cultural Constraints

There are no cultural constraints relevant to this project.

Section 4.2.11 Manufacturing Constraints

There are no manufacturing constraints relevant to this project.

Chapter 5: Project Design

Our system design follows a modular and layered architecture to ensure flexibility, scalability, and maintainability. Each component is designed as an independent module with clearly defined responsibilities, such as the rendering engine for visualizing web pages, the JavaScript engine for executing dynamic content, and the networking module for handling HTTP/HTTPS requests. These modules interact through well-defined APIs, allowing updates or replacements without affecting the entire system. The layered architecture organizes the browser into hierarchical layers, with the User Interface at the top, enabling interaction with users, and core layers like the browser engine, rendering engine, and Data Storage working below to process requests and render outputs. This separation of concerns ensures that each layer performs specific tasks while communicating seamlessly with others, providing a robust, efficient, and secure foundation for modern web browsing.

Section 5.1. Overview of System Components

This section introduces the system components of the NOVA Browser. It outlines the modular architecture and the layered design principles employed to ensure scalability, flexibility, and maintainability. Each system component, ranging from core functionalities like incognito mode to advanced features like AI-driven chat suggestions and network traffic monitoring, is described with its purpose and cross-references to relevant user stories. This section provides a clear foundation for understanding how the browser's features integrate to deliver a secure, efficient, and user-friendly experience.

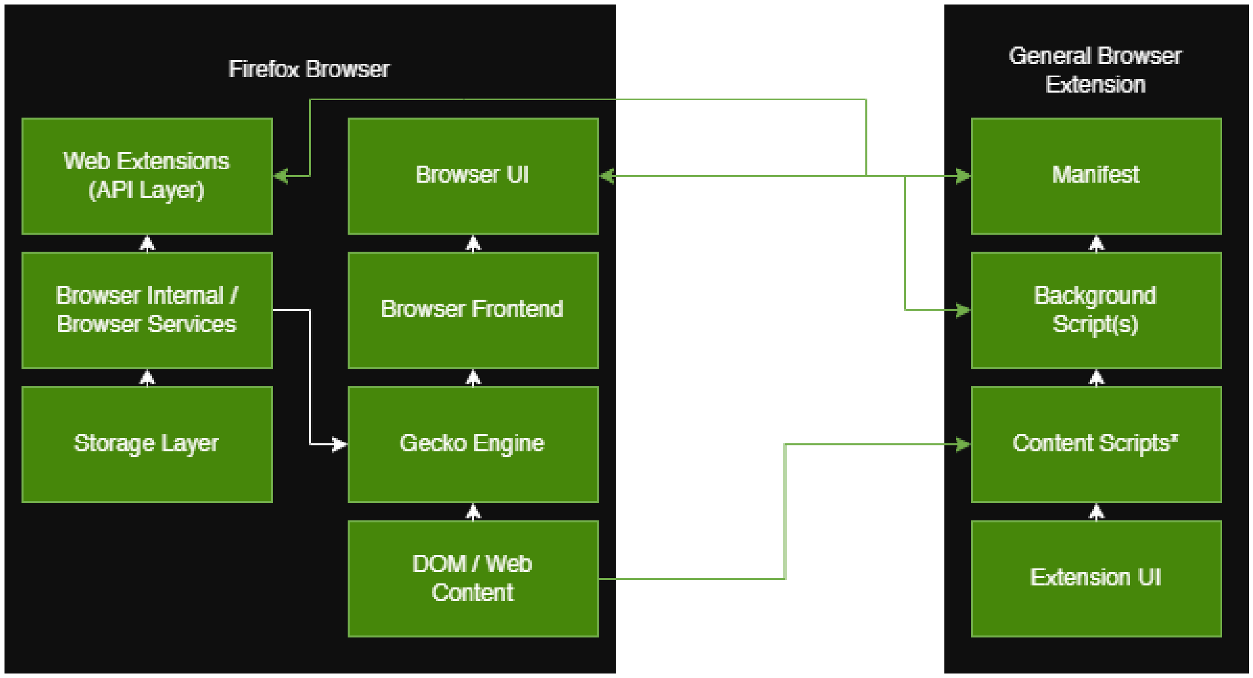


Figure 5.1 - System Design Diagram

Table 5.1.1: System Components (Extensions)

System component #	Name	Description	Cross-reference
SC-1	Incognito Mode	Enables private browsing without storing history, cookies, or user data.	US-1, T1, UI-3, FR-2
SC-2	Ad-Blocker	Blocks ads and tracking scripts to enhance privacy and improve load times.	US-2, FR-5, UI-7, NFR: Ad-Blocking Performance
SC-3	Command Processor	Allows users to set and execute custom commands for shortcuts and browser automation.	US-3, FR-3, UI-5, NFR: Customization
SC-4	AI Chat	Provides context-aware chat suggestions in real time.	US-4, FR-4, UI-8, NFR: Privacy, NFR: Search Response Time
SC-5	Plug-In Manager	Allows users to install, enable, disable, and manage plugins for enhanced browser functionality.	US-5, FR-3, UI-7, NFR: Customization
SC-6	Network Traffic Monitor	Monitors and displays network traffic in real-time within a web browser, allowing users to inspect HTTP/S requests, API calls, and resource loads for debugging and analysis.	US-6, FR-6, UI-6, NFR: Performance, NFR: Transparency
SC-7	User Profile/Preferences Manager	Manages user profiles and preferences, allowing users to customize browser settings, switch between profiles, and save personalized configurations	US-7, FR-4, UI-1, UI-2, NFR: Customization, NFR: Usability

Section 5.2. Structure and Relationship

Section 4.2 delves into the structural organization and interrelationships among the various components of the NOVA Browser. It highlights the layered architecture that ensures clear separation of responsibilities while promoting scalability and maintainability. This section explains how the user interface interacts with core modules like the browser engine, rendering engine, and backend database to deliver a seamless and efficient browsing experience. By exploring the relationships between components, such as how the networking module supports the rendering engine or how local storage ensures personalized user settings, Section 4.2 provides a cohesive view of the browser's system architecture and its ability to support both user-friendly and advanced functionalities.

Section 5.2.1. User Interface

The highest level of the browser, the user interface allows users to interact with the system, sending inputs to the browser engine, which processes the inputs through modules, such as the command manager.

Section 5.2.2. Browser Engine

The browser engine processes inputs and commands from the UI, and communicates with the rendering engine for displaying content. Accesses local storage for saving and retrieving user preferences and security-related information.

Section 5.2.3. Rendering Engine

The rendering engine converts and renders web content based on instructions from the browser engine. It interacts with other backend components like networking and the JavaScript interpreter to fetch and execute web resources.

Section 5.2.4. Networking

The networking module handles communication with the internet, fetching data and passing it to the rendering engine for display.

Section 5.2.5. JavaScript Interpreter

Collaborates with the rendering engine to execute JavaScript code embedded in web pages, ensuring interactive functionality.

Section 5.2.6. Backend Database

Supports the system by managing plug-ins and other backend data needed for browser operation. It communicates with the rendering engine to provide data essential for rendering content.

Section 5.2.7. Local Storage

Works closely with the browser engine to store and retrieve user data securely, ensuring consistent and personalized experiences.

Section 5.3. Detailed Component Description

Below is a detailed description of each component.

Section 5.3.1. Incognito Mode

Identification	SC-1 Incognito Mode
Type	Core System Component (C++/JavaScript)
Purpose	Enables private browsing without storing history, cookies, or user data.
Inputs	User action to activate incognito mode from the UI.
Outputs	A private browsing session with no data retention.
Data	**Session Data**: Temporary data required to render pages during the session, stored in memory only (e.g., cookies and cache), discarded after the session. **Incognito Status**: A flag indicating whether incognito mode is active, affecting data storage policies. **Blocked History/Tracking Data**: Incognito mode prevents storage of browsing history, cookies, cache, and form inputs, ensuring no trace remains after the session.
Internal Structure	Contains a session handler that controls data storage, cookies and cache settings for the private browsing session.
Processing	**S1**: Incognito mode is activated via the UI. **S2**: The session handler disables data storage for history, cookies, and cache. **S3**: All session data is stored temporarily in memory and discarded on session end.
Dependencies	Relies on the security layer to enforce data privacy and the data storage component to ensure no persistent data is stored.
Resources	Memory management libraries for temporary session storage.

Cross-Reference	US-1, T1, UI-3, FR-2

Section 5.3.2. Ad-Blocker Module

Identification	SC-2 Ad-Blocker Module
Type	JavaScript Module
Purpose	Blocks ads and tracking scripts to enhance privacy and improve load times.
Inputs	Page content data from network requests and user ad-blocking preferences.
Outputs	Ad-free and tracker-free page content.
Data	**Ad Signatures and Tracking URLs**: A library of known ad sources, tracking scripts, and URLs, used to identify and block ads on websites. **Blocked Element Data**: Information on elements that were blocked (e.g., ad count, blocked domains), displayed to the user. **User Ad-Blocking Preferences**: User-defined settings for ad-blocking behavior, such as site exceptions or aggressive blocking modes.
Internal Structure	Comprises a script scanner that identifies and removes ad scripts and trackers.
Processing	**S1**: Ad-blocker scans the page content for ad scripts and trackers. **S2**: Blocks or removes identified ads and tracking elements. **S3**: Notifies the user about the number of blocked ads through the UI.
Dependencies	Depends on the security layer to manage blocking rules and UI for displaying ad-blocking status.
Resources	Ad-blocking libraries and a database of ad/tracker signatures.
Cross-Reference	US-2, FR-5, UI-7, NFR: Ad-Blocking Performance

Section 5.3.3. Command Processor

Identification	SC-3 Command Processor
Type	Core System Component (C++/JavaScript)
Purpose	Allows users to set and execute custom commands for shortcuts and browser automation.
Inputs	User-defined custom commands from the UI.
Outputs	Executes the specified browser action, like opening a tab or navigating to a URL.

Data	**User-Defined Commands**: Custom commands and shortcuts that users create, mapping specific keystrokes to browser actions. **Command Actions Mapping**: A predefined list of browser actions (e.g., open tab, refresh page) linked to user-created shortcuts. **Command Execution History**: A log of executed commands, which may be helpful for troubleshooting or future customization.
Internal Structure	Contains a command parser and an action executor that maps commands to browser functions.
Processing	**S1**: Command Processor receives a custom command input. **S2**: The command parser matches the input with predefined browser actions. **S3**: Executes the associated browser action.
Dependencies	Requires data storage to retain user-defined commands.
Resources	Command-mapping database.
Cross-Reference	US-3, FR-3, UI-5, NFR: Customization

Section 5.3.4. AI Chat Module

Identification	SC-4 AI Chat Module
Type	Machine Learning Module
Purpose	Provides content-aware chat suggestions in real-time.
Inputs	User query input from search bar.
Outputs	Real-time, relevant search suggestions displayed to the user.
Data	**User Query Data**: The current search query input by the user, used to generate real-time suggestions. **Search Suggestion Cache**: Cached results and popular searches, allowing faster response to common queries. **AI Model Data**: Weights and configurations for the machine learning model that powers search predictions, updated periodically.
Internal Structure	Includes a query processor and AI model.
Processing	**S1**: User types a query, which the query processor receives. **S2**: The AI model processes the query to generate relevant responses. **S3**: Responses are displayed in real-time as the user queries.
Dependencies	Requires access to LLAma 3.1 Language Model.
Resources	Machine learning libraries and algorithms for generating content aware suggestions.

Cross-Reference	US-4, FR-4, UI-8, NFR: Privacy, NFR: Search Response Time

Section 5.3.5. Plug-In Manager

Identification	SC-5 Plug-In Manager
Type	Core System Component (C++/JavaScript)
Purpose	Allows users to install, enable, disable and manage plugins for enhanced browser functionality.
Inputs	User requests for plug-in management from the UI.
Outputs	Installation status and a list of active or inactive plug-ins.
Data	**Plugin Metadata**: Details about each installed plugin, including name, version, permissions, and settings. **Plugin Status**: Active or inactive status of each plugin, determining if the plugin is enabled or disabled in the current session. **User Plugin Preferences**: User-specific settings for plugins, such as configuration data or usage limits for certain plugins. **Sandboxed Plugin Data**: Data generated by plugins, stored in a secure, isolated environment to prevent unauthorized access.
Internal Structure	Contains a plug-in registry, sandbox manager, and plug-in settings handler.
Processing	**S1**: User initiates plugin installation or management through the UI. **S2**: Plugin Manager updates the plugin registry and activates the plugin in a sandboxed environment. **S3**: Plugin Manager updates the **Data Storage** with plugin settings and status.
Dependencies	Relies on data storage for managing plug-in data and security layer for sandboxing.
Resources	Plug-in management APIs, sandboxing libraries.
Cross-Reference	US-5, FR-3, UI-7, NFR: Customization

Section 5.3.6. Network Traffic Monitor

Identification	SC-6 Network Traffic Monitor
Type	JavaScript Web Application
Purpose	Monitors and displays network traffic in real-time within web browser, allowing users to inspect HTTP/S requests, API calls, and resource loads for debugging and analysis.

Inputs	Network traffic from browser's internal APIs (i.e., Fetch API, XMLHttpRequest, WebSocket data), user-specified filters, and headers.
Outputs	Network traffic logs, performance metrics, and decoded request/response payloads, displayed in a tabular format with options for export.
Data	**Request Metadata**: URL, method (GET, POST, etc.), status codes, request/response headers, and timestamps. **Response Payloads**: Decoded content such as JSON, HTML, or binary data. **Filters**: User-defined rules for filtering requests by URL, status, or method. **Performance Metrics**: Load times, transfer size, and latency of individual requests.
Internal Structure	JavaScript modules handling HTTP/S requests. UI for filtering, searching, and visualizing traffic logs. Backend integration for secure data processing if required. Data storage mechanism for temporary session logs.
Processing	**S1:** Captures network events using browser APIs like Fetch, XMLHttpRequest, or WebSockets. **S2:** Filters requests based on user-defined criteria (e.g., URL, method). **S3:** Decodes and parses request/response data for display. **S4:** Provides performance insights like load time and latency for each request.
Dependencies	Optional backend for advanced features like saving session logs.
Resources	Packet capture and analysis tools such as tcpdump and Zeek.
Cross-Reference	US-6, FR-6, UI-6, NFR: Performance, NFR: Transparency

Section 5.3.7. Profile and Preferences Manager

Identification	SC-7 Profile and Preferences Manager
Type	Core System Component (C++/JavaScript)
Purpose	Manages user profiles and preferences, allowing users to customize browser settings, switch between profiles, and save personalized configurations.
Inputs	- User commands to create, delete, or switch profiles - User preferences (e.g., themes, privacy settings, extensions) - Sync data for preferences (if account-based profiles are used)
Outputs	- Active user profile applied - Saved preferences reflected in the browser - Confirmation messages for profile changes
Data	**User Profile Data**: Unique settings for each profile, including extensions, bookmarks, and history. **Preferences Data**: Theme, language, privacy settings, and other customizations.

	Session Sync Data: Synchronization data for profiles across devices (optional).
Internal Structure	**Profile Manager Module**: Handles creation, deletion, and switching of profiles. **Preferences Module**: Stores and applies user preferences like themes, privacy settings, and extensions. **Data Storage**: A database or file system for saving profile and preferences data persistently **Sync Integration**: Optional module to sync profiles and preferences across devices (e.g., via a user account).
Processing	**S1:** Accepts user commands for managing profiles and preferences. **S2:** Applies active profile settings to the browser. **S3:** Saves user preferences and updates storage. **S4:** Retrieves and syncs profile data as required.
Dependencies	File system or database for persistent storage. Browser UI for user interaction.
Resources	Mozilla XPCOM framework for managing components. SQLite for preference storage.
Cross-Reference	US-7, FR-4, UI-1, UI-2, NFR: Customization, NFR: Usability

Section 5.4. Methodologies

This section outlines the research and identification of tools and methods for implementing the project, along with considerations for alternative designs, resource allocation, and skill requirements.

Section 5.4.1. Research New Tools and Methods

Section 5.4.1 focuses on the research and identification of tools and methodologies essential for implementing the NOVA Browser. It examines various technologies, frameworks, and approaches to ensure alignment with the project's goals of privacy, performance, and security.

Section 5.4.1.1. Research Tools

Chromium [2]
Chromium is an open-source browser engine widely used as the foundation for browsers like Google Chrome and Microsoft Edge. Its extensive support for modern web technologies, developer tools, and plugin compatibility makes it a strong candidate for web-based projects. However, its inclusion of telemetry and tracking features conflicts with privacy-focused objectives. Additionally, its complexity and resource requirements make it less suitable for a lightweight browser. As such, Gecko Engine, with its emphasis on privacy and streamlined integration, is better aligned with the project's goals.

Gecko Engine [8]
The Gecko Engine is an open-source rendering engine developed by Mozilla, designed to support modern web technologies like HTML5, CSS3, and JavaScript. It is particularly well-suited for privacy-focused projects due to its robust architecture, lack of telemetry, and active open-source development community. Its flexibility enables seamless integration with browser components such as the ad-blocker and security layers. Although it may lack certain proprietary features found in competing engines like Chromium, its alignment with the project's privacy goals makes it an ideal choice.

SQLite [5]
SQLite is a lightweight embedded database management system widely recognized for its simplicity and efficiency. It is particularly effective for local data storage in applications that do not require distributed database systems. In this project, SQLite serves as the backbone for storing user preferences, bookmarks, and session data, ensuring quick data retrieval and minimal resource usage. Its open-source nature and compatibility with resource-constrained environments make it an optimal choice for the project's requirements.

MySQL [13]
MySQL is a robust relational database system designed for large-scale, multi-user applications. It supports complex queries and transactions, making it a powerful choice for data-heavy projects. However, it requires more setup and resources, which could be an unnecessary overhead for a browser focused on local data storage. SQLite's simplicity and lightweight nature make it the preferred choice for this project’s requirements.

uBlockOrigin [3]
uBlock Origin is a lightweight and powerful ad-blocker that enhances privacy and improves user experience by blocking intrusive advertisements and trackers. Its low resource consumption and comprehensive filtering capabilities make it a preferred tool for projects requiring efficient ad-blocking functionality. Being open-source, it integrates well into the browser, maintaining consistency with the project's emphasis on transparency and accessibility. Its proven reliability and minimal system impact ensure a distraction-free and secure browsing experience.

AdGuard [14]
AdGuard is an advanced ad-blocking tool offering features like DNS-level blocking and parental controls. It is available as a standalone app and browser extension, providing cross-platform compatibility. While effective, AdGuard is heavier on system resources and includes paid features, making it less ideal for this project’s focus on open-source, lightweight solutions. uBlock Origin, with its efficiency and strong ad-blocking capabilities, offers a better fit for enhancing user privacy and performance.

Wireshark [15]
Wireshark is a leading packet analysis tool that provides in-depth insights into network traffic. It enables users to monitor real-time network activity, analyze HTTP/S requests, and diagnose security vulnerabilities. Its open-source availability ensures cost-free access to advanced diagnostic features, making it suitable for privacy-conscious users and developers. Although it requires a steeper learning curve, its comprehensive capabilities align well with the project's goal of empowering users with transparency and control over their network data.

Fiddler [16]
Fiddler is a user-friendly web debugging tool that specializes in monitoring HTTP/S traffic and analyzing session performance. Its streamlined interface makes it ideal for developers working on web applications. However, it lacks the low-level packet analysis capabilities that Wireshark provides, which are crucial for advanced diagnostics and network monitoring in this project. Wireshark's comprehensive features make it a better fit for addressing the security-conscious goals of the system.

Section 5.4.1.2. Research Methods

Machine Learning Algorithms
Machine learning algorithms are used to provide real-time, context-aware chat suggestions personalized to the user's behavior and preferences. These algorithms dynamically adapt to user input, creating a richer browsing experience. By leveraging techniques like natural language processing, the AI module can generate relevant and timely suggestions, enhancing usability. While computationally intensive, the benefits of adaptive learning and enhanced interactivity make machine learning a cornerstone for the project's innovative features.

Rule-Based System
Rule-based systems rely on predefined logic and conditions to process user input and generate responses. These systems are easy to implement and less resource-intensive compared to machine learning models. However, they lack adaptability and cannot personalize responses based on user behavior, resulting in a less dynamic user experience. For this project, machine learning algorithms are preferred as they offer the sophistication needed for context-aware, real-time chat suggestions.

Input Validation and Sanitization
Input validation and sanitization are essential security measures that protect the system from malicious attacks like SQL injection and cross-site scripting (XSS). These methods ensure that only valid and safe inputs are processed by the system, mitigating risks and maintaining the

integrity of user data. The project employs comprehensive frameworks to check and clean all user inputs, offering a reliable defense against potential exploits and reinforcing system security.

Web Application Firewall
A web application firewall offers automated protection against malicious inputs like SQL injection and XSS attacks. While scalable and easy to implement, WAFs often add cost and complexity to a system and provide less customization compared to in-house input validation methods. In this project, robust input validation and sanitization frameworks are more appropriate, offering fine-grained control and cost-effectiveness for securing the browser.

Crash Recovery Mechanisms
Crash recovery mechanisms are designed to preserve session data and ensure seamless restoration after unexpected browser shutdowns. These mechanisms periodically save session states, such as open tabs and active data, allowing users to resume work without loss. By integrating real-time session management tools, the project ensures that data remains secure and accessible, delivering a smooth and reliable user experience even in the event of crashes or errors.

Periodic Checkpointing
Periodic checkpointing involves saving snapshots of session data at regular intervals. This method is simpler to implement and uses less memory compared to real-time crash recovery. However, it risks losing data between checkpoints and results in slower recovery times. Real-time crash recovery mechanisms, as planned for this project, ensure a smoother user experience by preserving and restoring session data more effectively after unexpected crashes.

Section 5.4.2. Identify New Tools and Methods

This section identifies the tools and methods selected for NOVA Browser's development, emphasizing their role in achieving the project's objectives. It highlights the rationale behind these choices, focusing on their ability to enhance functionality, security, and user experience while maintaining the browser's privacy-focused design principles.

Section 5.4.2.1. Identify Tools

Based on extensive research, we carefully selected a suite of tools to support the project’s goals of privacy, security, and functionality. Each tool was chosen for its proven reliability, compatibility with the system's objectives, and ability to enhance user experience while maintaining strict privacy standards. Below is a detailed explanation of the selected tools and their significance in achieving project goals.

Gecko Engine [8]
The Gecko Engine was chosen as the core rendering engine for its robust open-source architecture, emphasis on privacy, and alignment with the goals of the project. The engine will handle web page rendering, ensuring compatibility with modern web standards such as HTML5, CSS3, and JavaScript. Its flexibility also allows seamless integration with other browser components, such as the ad-blocker and security layers, enabling a unified, privacy-focused user experience.

uBlock Origin [3]
uBlock Origin is a lightweight and highly effective ad-blocker that excels at preventing intrusive advertisements and trackers. It offers superior performance compared to many alternatives, consuming fewer system resources while maintaining comprehensive blocking capabilities. Integrated as a native ad-blocker within the browser, uBlock Origin will provide users with a secure and distraction-free browsing experience. Its ability to block both visible and hidden trackers will complement the browser's incognito mode and privacy-focused features.

Wireshark [15]
Wireshark was included to cater to advanced users who require network monitoring and diagnostics. As a leading packet analysis tool, Wireshark provides detailed insights into network traffic and potential security vulnerabilities. Wireshark will enable users to monitor network activity in real-time, analyze packet-level data, and diagnose potential security threats. This feature aligns with the project’s goal of providing transparency and control to security-conscious users.

Section 5.4.2.2. Identify Methods

Key methods for the project have been carefully chosen to ensure seamless functionality, robust security, and optimized performance.

Machine Learning Algorithms
The quantized model of Gemma3, utilizing Ollama, is designed to enhance user experience by delivering intelligent, context-aware features such as real-time chat suggestions and personalized interactions. These algorithms leverage advanced techniques like natural language processing (NLP) and adaptive learning to understand user inputs, predict intent, and generate relevant responses dynamically. By processing data locally to prioritize privacy, the algorithms ensure seamless integration with the browser's functionality while maintaining high performance and minimizing resource usage. This approach combines the power of AI with the project's commitment to security and user autonomy.

Data Encryption and Masking
We will be securing sensitive user data, such as passwords and browsing history, during storage and transmission. In order to implement this we will adopt encryption standards like AES-256 for data at rest and TLS 1.3 for secure data in transit. Data masking techniques will be used to obscure sensitive information displayed on the user interface, such as partially hidden passwords.

Resource Management
We plan on optimizing memory and CPU usage to maintain stability and performance, even with multiple open tabs and resource-intensive plugins. To implement this we will be utilizing Gecko's memory management tools to prioritize active tabs while suspending inactive ones. Implement dynamic resource allocation algorithms to adjust memory usage based on the system's capabilities and browser load.

Input Validation and Sanitization
We will be protecting against malicious inputs and attacks such as SQL injection or cross-site scripting (XSS). We will be using robust input validation frameworks to check and sanitize user inputs. Regular penetration tests will be conducted to evaluate the system's resilience against such exploits

Crash Recovery Mechanisms
We seek to preserve session data and ensure the browser can recover gracefully from unexpected crashes or shutdowns in regular browsing mode. We will develop session management features that periodically save tab states and active data. Upon relaunch, the system will retrieve saved states and restore the session seamlessly.

Network Monitoring and Analysis
We seek to provide tools for advanced users to monitor network traffic for diagnostics and security analysis. We will integrate Wireshark’s API for real time packet analysis. User-friendly visualizations for presenting network data, with details on active connections, bandwidth usage, and potential threats.

Plugin Sandboxing
We will ensure the secure execution of plugins without compromising core browser functionality. We plan on isolating plugin operations from the main browser processes using sandboxing techniques. This prevents rogue plugins from accessing sensitive data or interfering with the browser's performance.

Section 5.5. Design Alternatives

When designing the browser, we evaluated several alternative approaches to balance privacy, functionality, and performance. Each alternative was considered for its advantages, limitations, and alignment with our goals

Chromium [2]
Chromium, as the base for popular browsers like Chrome and Edge, offers extensive support for web technologies, developer tools, and wide-ranging plugin compatibility. It also has a robust ecosystem with an active developer community. However, Chromium's architecture includes telemetry features and tracking capabilities that counter our project's privacy-focused mission. The complexity of removing these built-in features would outweigh the benefits. We ultimately chose Gecko for its privacy-first design, active open-source development community, and compatibility with our goals of eliminating telemetry and providing enhanced user security.

Developing Our Own Ad-Blocker Instead of Using uBlock Origin
Creating a proprietary ad-blocker would allow complete customization and seamless integration with the browser’s architecture. It would enable full control over blocking rules and offer unique features not available in existing solutions, potentially differentiating our browser in the market. However, developing an ad-blocker from scratch is resource-intensive and would require significant development time, especially to achieve the same level of efficiency and effectiveness as established tools like uBlock Origin. Additionally, maintaining an in-house ad-blocker would involve continuous updates to keep up with new ad formats and tracking techniques, adding to long-term development costs. We opted to integrate uBlock Origin, a proven and highly effective ad-blocking library, to save development time while ensuring ad-blocking functionality. This decision allows us to focus on other critical features like incognito mode and AI-powered tools.

Cloud-Based AI vs. Local AI
Leveraging cloud-based AI would provide access to state-of-the-art computational resources, enabling highly accurate and advanced AI functionalities. Pre-trained models can be updated frequently without requiring changes on the client side. However, the use of a cloud-based AI introduces privacy concerns, as user data needs to be transmitted to external servers for processing. It also relies on consistent internet connectivity, which could degrade performance for users in low-bandwidth environments.

Local AI ensures complete privacy by processing data entirely on the user's device, aligning with our project’s privacy-first mission. It also functions without internet connectivity, offering independence and faster response times. However, local AI models are resource-intensive and may perform poorly on devices with limited computational power. Developing efficient models requires significant optimization and testing to balance performance with resource usage.

We selected local AI to prioritize user privacy and autonomy. Optimizing AI models to run efficiently on user devices ensures alignment with our core mission of delivering a secure and private browsing experience.

Section 5.6. Reuse and Relationship With Other Products

The project builds upon the open-source Firefox browser, leveraging its design and Firefox compatibility. Tools like uBlock Origin and Hugging Face are integrated to enhance features while ensuring compliance with open-source licenses.

Section 5.7. Resource List

The table below outlines the resources required for the development of NOVA Browser. It details the tools, technologies, and team contributions necessary to ensure the successful implementation and delivery of the project.

Table 5.7.1. Resource List

Resources	Description	Availability	Cost
Braden Norum	Developer working on the project	10 hours per week	free
Roshni Varkey	Developer working on the project	10 hours per week	free
Aman Luthra	Faculty Advisor	1 hour per week	free
Gecko Engine	Web Rendering Engine	Open-Source 24/7	free
Wireshark	Network Monitoring Tool	Open-Source 24/7	free
uBlock Origin	Ad-blocker Library	Open-Source 24/7	free
Gemma 3 Model	Local AI model for AI chat function	Open-Source 24/7	free

Section 5.8. Resource Skill List

Section 4.8 highlights the skills and expertise needed to develop the NOVA Browser. It identifies the technical proficiencies and knowledge areas essential for implementing core functionalities, security measures, and advanced features.

Table 5.8.1. Resource Skill List

Skill	Application	Reference
Software Engineering	Developing core browser functionalities	User Stories, Functional Requirements
Machine Learning	Implementing AI chat and navigation	AI chat, LLAma 3.1 Language Model
Web Development	Designing and optimizing the browser UI	HTML/CSS, JavaScript
Security Engineering	Encrypting data, managing session security	Security Requirements
Network Analysis	Monitoring and analyzing network packets	Network Traffic Monitor

Chapter 6: Test Plan

Section 6.1. Test Plan

We will conduct unit testing for individual components (e.g., the Incognito Mode Handler and Ad-blocker Module). Integration testing will be used to verify interactions between components for example those between the Browser UI and Rendering Engine. Functional testing will validate that the browser performs as expected under user stories and functional requirements. Additionally, non-functional testing such as performance (page load times), security (sandboxing plugins), and robustness (crash recovery) will be performed. A combination of manual testing for user-centric features and automated testing for repetitive tasks like regression and load testing will be used.

Unit tests will validate the functionality of individual design components and subcomponents, such as the Ad-blocker Module, AI Search Module, and Incognito Mode Handler. Integration tests will be used to verify interactions between components, ensuring smooth communication and data flow like the Browser UI with the Rendering Engine. Functional tests will validate that features like the implementation of user stories and functional requirements work correctly. Validation tests will confirm that entire user stories (e.g. Enable incognito mode) are fully implemented and behave as intended. Non-functional tests will evaluate performance, security, reliability, and robustness.

Table 6.1.1 Test Cases

Test Case number and name	**Test type (unit, functional, etc.)**	**Short Description**	**Cross-references**
T1: Enable Incognito	Functional	Validate incognito mode activation.	UC-1.1, FR-1.1
T2: Ad-blocker Activation	Functional	Verify ads and trackers are blocked on activation.	UC-1.2, FR-1.2
T3: AI Chat	Functional	Confirm real-time AI chat.	UC-1.5, FR-1.5
T4: Plugin Installation	Integration	Test plugin installation and sandboxing.	UC-1.6, FR-1.6
T5: Page Load Time	Performance	Verify page loads within 5 seconds.	NFR: Performance
T6: Security Validation	Security	Ensure incognito mode prevents persistent storage.	NFR: Security, FR-1.1

Test Case number and name	Test type (unit, functional, etc.)	Short Description	Cross-references
T1: Enable Incognito	Functional	Validate incognito mode activation.	UC-1.1, FR-1.1
T2: Ad-blocker Activation	Functional	Verify ads and trackers are blocked on activation.	UC-1.2, FR-1.2
T3: AI Chat	Functional	Confirm real-time AI chat.	UC-1.5, FR-1.5
T4: Plugin Installation	Integration	Test plugin installation and sandboxing.	UC-1.6, FR-1.6
T7: Crash Recovery	Robustness	Verify recovery of tabs after a crash.	NFR: Robustness

Section 6.2. Test Cases

Section 6.2 provides detailed descriptions of the test cases designed to validate the functionality, performance, and security of the NOVA Browser. Each test case outlines specific objectives, preconditions, and expected outcomes, ensuring the browser meets its requirements and delivers a reliable user experience.

Table 6.2.1. Enable Incognito Mode

Name	T1: Enable Incognito Mode
Type	Functional
Description	Validate that incognito mode is activated and prevents persistent data storage.
Preconditions	Browser is installed and functional; user is logged in.
Basic Course of Events	1. Open the browser. 2. Enable incognito mode through the Browser UI. 3. Navigate to a website. 4. Check that no cookies, history, or cache data are stored after closing the session.
Expected Results	Incognito mode activates, and no persistent data is saved after session termination.
Acceptance Criteria	The browser must not store history, cookies, or cache during incognito sessions.

Cross-references	UC-1.1, FR-1.1, NFR: Security

Table 6.2.2. Ad-blocker Activation

Name	T2: Ad-blocker Activation
Type	Functional
Description	Verify that enabling the ad-blocker prevents ads and trackers from loading.
Preconditions	Browser is running; a test website with ads is available.
Basic Course of Events	1. Open the browser. 2. Navigate to a test website with known ads and trackers. 3. Enable the ad-blocker via the Browser UI. 4. Refresh the test website.
Expected Results	Ads and trackers are blocked, and the browser displays the number of blocked elements.
Acceptance Criteria	No ads or trackers are visible or loaded in the network traffic logs.
Cross-references	UC-1.2, FR-1.2, NFR: Security

Table 6.2.3. AI Chat

Name	T3: AI Chat
Type	Functional
Description	Confirm that the AI Chat Module provides real-time, relevant chat.
Preconditions	Browser is running; AI bar is accessible.
Basic Course of Events	1. Open the browser. 2. Click on the AI Chat bar. 3. Enter a partial query (e.g., "weather"). 4. Observe the answers displayed by the search bar.
Expected Results	Context-aware, relevant answers appear dynamically as the user types.
Acceptance Criteria	Response must be relevant to the input and displayed within 4 seconds.
Cross-references	UC-1.4, FR-1.4, NFR: Performance

Table 6.2.4. Plug-In Installation

Name	T4: Plug-In Installation
Type	Integration
Description	Test that plugins can be installed, activated, and sandboxed securely.
Preconditions	Browser is running; plugin repository is accessible.
Basic Course of Events	1. Open the browser. 2. Navigate to the plugin management page in the Browser UI. 3. Select and install a plugin. 4. Activate the installed plugin and verify its functionality. 5. Check the Security Layer logs for sandboxing validation.
Expected Results	Plugin installs and activates successfully, runs securely in a sandbox.
Acceptance Criteria	Plugin functionality works as expected, with no unauthorized data access.
Cross-references	UC-1.5, FR-1.5, NFR: Security, Robustness

Table 6.2.5. Page Load Time

Name	T5: Page Load Time
Type	Performance
Description	Verify that web pages load within the acceptable time frame (5 seconds).
Preconditions	Stable network connection (5 Mbps); browser is running.
Basic Course of Events	1. Open the browser. 2. Navigate to a set of test websites with varying content sizes. 3. Measure the time taken to fully load each page.
Expected Results	Pages load within 5 seconds for 95% of the test cases.
Acceptance Criteria	The page load time must not exceed the specified threshold under normal network conditions.
Cross-references	NFR: Performance

Table 6.2.6. Security Validation

Name	T6: Security Validation
Type	Security
Description	Ensure that no persistent data is stored during incognito sessions.

Preconditions	Browser is running in incognito mode.
Basic Course of Events	1. Open the browser in incognito mode. 2. Navigate to multiple websites. 3. Close the browser. 4. Check the system for cookies, history, or cache traces.
Expected Results	No data related to the session is found on the system.
Acceptance Criteria	Incognito mode must leave no persistent data after the session ends.
Cross-references	FR-1.1, NFR: Security

Table 6.2.7. Crash Recovery

Name	T7: Crash Recovery
Type	Robustness
Description	Verify that the browser restores tabs and session data after a crash.
Preconditions	Browser is running with multiple tabs open.
Basic Course of Events	1. Open the browser and load multiple tabs. 2. Simulate a crash (e.g., force close the application). 3. Reopen the browser. 4. Check that all tabs and session data are restored.
Expected Results	The browser restores all tabs and session data without errors.
Acceptance Criteria	Tabs and session data must be recoverable after a crash.
Cross-references	NFR: Robustness

Chapter 7: Project Plan

Chapter 7 outlines the comprehensive plan for the development of the NOVA Browser, detailing the timeline, tasks, risks, and resources required to ensure the project's success. This chapter provides a structured approach to managing the project, from initial development phases to final deployment.

Section 7.1. Sprints

This section focuses on the sprint-based development methodology adopted for the NOVA Browser project. Each sprint is designed to achieve specific milestones, ensuring steady progress and incremental delivery of functional components. The subsections detail the goals, timelines, and tasks for each sprint phase.

Section 7.1.1. Sprint 1

This subsection covers the initial sprint, which focuses on establishing core functionalities such as incognito mode, ad-blocker integration, and basic UI elements. The sprint aims to deliver a foundational prototype for further development.

Sprint goal: Implement basic functionality, including incognito mode activation, ad-blocker integration, and the browser's core user interface.

Sprint demo goal: Deliver a working demo that demonstrates incognito mode functionality, ad-blocker activation, and a functional rendering engine.

Starting date and period: February 1st to February 28th (4 weeks)

Sprint master: Braden Norum

Table 7.1.1.1. Sprint 1 Plan

#	Backlog	Hardware/Software resources	Possible team members	Estimated hours	Points/ Difficulty	Cross-references to requirements, design components
1	Install, Troubleshoot, and Initial Build of Browser Framework	Browser Framework, Python, Rust, C++	Braden Roshni	2	1	

2	Design and implement UI for incognito mode toggle	Browser framework (Pulse, Gecko), IDE	Braden Roshni	10	3	UC-1.1, FR-1.1, IM-1
3	Develop Incognito Mode Handler	Gecko Engine, Memory Management Tools	Braden Roshni	15	5	FR-1.1, IM-1, Security Layer Design
4	Integrate Ad-blocker Module	uBlock Origin, Gecko API	Braden Roshni	20	6	UC-1.2, FR-1.2, IM-2

Section 7.1.2. Sprint 2

The second sprint emphasizes implementing advanced security features, plugin support, and network monitoring tools. It ensures the removal of telemetry and strengthens the browser's privacy capabilities.

Sprint goal: Implement advanced security details, like removal of telemetry, any tracking blocking not covered in previous sprint, packet tracing, and plug-in support.

Sprint demo goal: Deliver a working demo which demonstrates the packet tracer, and shows evidence of the removal of telemetry and tracing.
Starting date and period: March 1st to March 30th (30 Days)

Sprint master: Roshni Varkey

Table 7.1.2.1. Sprint 2 Plan

#	Backlog	Hardware/Software resources	Possible team members	Estimated hours	Points/ Difficulty	Cross-references to requirements, design components
1	Install, Troubleshoot, and Initial Build of Browser Framework	Browser Framework, Python, Rust, C++	Braden Roshni	2	1	
2	Design and	Browser	Braden	10	3	UC-1.1,

	implement UI for incognito mode toggle	framework (Pulse, Gecko), IDE	Roshni			FR-1.1, IM-1
3	Develop Incognito Mode Handler	Gecko Engine, Memory Management Tools	Braden Roshni	15	5	FR-1.1, IM-1, Security Layer Design
4	Integrate Ad-blocker Module	uBlock Origin, Gecko API	Braden Roshni	20	6	UC-1.2, FR-1.2, IM-2

Section 7.1.3. Sprint 3

The final sprint targets the completion of remaining functionalities, including AI chat integration and custom commands. This phase culminates in the delivery of a fully functional and tested browser.

Sprint goal: Finish any remaining goals, focusing on AI chat within browser search bar, and custom commands.

Sprint demo goal: Deliver a fully functioning build of the browser.

Starting date and period: April 2nd to April 30th (29 Days)

Sprint master: Braden Norum

Table 7.1.3.1. Sprint 3 Plan

#	**Backlog**	**Hardware/Software resources**	**Possible team members**	**Estimated hours**	**Points/ Difficulty**	**Cross-references to requirements, design components**
1	Develop AI Chat	LLaMa 3.1, Sandboxed Execution Environment, Local Auth. Mechanism, Storage Frameworks	Braden Roshni	40	10	UC-1.5, FR-1.5, IM-4
2	Develop Method of	Custom Encryption	Braden Roshni	15	5	UC-1.3, FR-1.3, IM-3

	Creating Custom Commands	Libraries, Command Parser, Secure Backend APIs				
3	Develop Method of Executing Custom Commands	Execution Sandboxes, Secure Memory Management Tools, API Gateways	Braden Roshni	12	4	UC-1.4, FR-1.3, IM-3
4	Develop User Preferences and Profiles Functionality and UI	Gecko API, Secure Session Management Libraries	Braden Roshni	20	6	UC-1.8, FR-1.8, IM-7

Section 7.2. Risk Plan

Section 6.2 outlines the potential risks associated with the development of the NOVA Browser and the mitigation strategies to address them. It identifies challenges such as AI integration, security vulnerabilities, and performance issues, detailing proactive measures to ensure project stability, data protection, and user satisfaction throughout the development process.

Risk: AI Integration and Testing
Integrating AI-powered features like chat and navigation presents challenges in ensuring smooth functionality and compatibility with the browser. Errors in integration could result in reduced user experience or unstable features. To mitigate this risk we have decided to dedicate additional time during the first sprint and every sprint after to thoroughly test and debug AI integration, ensuring it operates seamlessly with other components.

Risk: Potential Data Breaches or Input Validation Failures
Without robust security measures, the browser could be vulnerable to data breaches or attacks like SQL injection. These vulnerabilities could compromise user data and damage trust. To mitigate this risk we have decided to regularly perform penetration testing and implement strict input validation to identify and eliminate vulnerabilities early in development. Routine security audits will ensure ongoing protection.

Risk: Slow Page Loads or Crashes with Multiple Tabs
Heavy resource usage from multiple open tabs or unoptimized processes can lead to slow performance or browser crashes, frustrating users. To mitigate this risk we have decided to

Optimize resource allocation and memory management, and conduct stress tests to ensure the browser remains stable and responsive even under heavy loads. Regular performance testing will highlight and resolve bottlenecks.

Section 7.3. Estimated Financial Budget

This section provides an overview of the financial resources required for the NOVA Browser project. It includes a breakdown of estimated costs for essential hardware, software, and tools to ensure the successful development and deployment of the browser within budget constraints.

Item	**Estimated Cost**
Inland TN450 500GB 3D TLC NAND PCIe Gen 4 x4 NVMe M.2 2280 Internal SSD	$38.99
Total Cost	**$38.99**

Section 7.4. Teamwork Plan

NOVA will be developed using Visual Studio Code and tools like the Gecko Engine, and uBlock Origin. Tasks will be managed with Notion, while GitHub will host the repository, with each team member working on separate branches to develop specific modules. The sprint master will oversee code integration and resolve conflicts.

Weekly meetings via Zoom will ensure progress updates and collaboration, with additional guidance from the faculty advisor. The project will be completed over three sprints: core functionalities in the first, advanced security and plugin features in the second, and AI chat and user profile management in the final sprint. Testing at each stage will ensure reliability, performance, and security. This plan ensures efficient collaboration and project success.

Chapter 8: Project Methods

Section 8.1: Technical Methods

The algorithms used for NOVA power many of the extensions used in the system.

Section 8.1.1 AI Chat

To power the AI-driven chat functionality, we developed a local query suggestion algorithm integrated with the Gemma 3 language model. This algorithm generates intelligent, context-aware suggestions in real-time as the user types in the browser's extension tab. Unlike traditional query suggestions which rely on remote servers and compromise user privacy, our implementation runs entirely locally to ensure data never leaves the device. The algorithm uses natural language processing (NLP) techniques to predict the user's intent and retrieve or generate relevant completions. This was implemented as a JavaScript interface connected to a locally running instance of the model via the Ollama API. To ensure performance, the model is optimized to respond in under 12 seconds and outputs are streamed and rendered incrementally in markdown format.

Section 8.1.1.1 AI Chat Processing Steps

1. User input is captured dynamically from the search bar as they type.

2. The query string is passed to a local server running Gemma 3 via Ollama.

3. The model analyzes the query using NLP and returns relevant suggestions.

4. Suggestions are streamed into the UI using markdown parsing.

5. The system updates the display in real-time with low latency.

Section 8.1.2 Plugin Sandboxing Security Method

To protect users from malicious or overly-permissive plugins, we implemented a plugin sandboxing method that isolates each plugin's functionality from the core browser environment. This method ensures that plugins operate within strict boundaries and cannot access sensitive browser data unless explicitly allowed by the user. We used Firefox's extension architecture and content security policies to create controlled communication channels between plugins and browser components. The system defines permissions at installation time and enforces them during runtime using background scripts and manifest controls. This sandboxing approach guarantees that even if a plugin is compromised, it cannot affect core browser functions or access user data. All plugin data is stored in isolated storage and any access to core APIs is tightly regulated through message passing and permission verification.

Section 8.1.2.1 Plugin Sandboxing Processing Steps

1. When a plugin is installed, the browser creates a separate sandboxed environment for it using Firefox's content and background script model.

2. Plugin permissions are defined in the manifest and checked before execution.

3. During runtime, plugin actions are monitored and restricted to their sandbox, using controlled message passing to interact with the browser.

4. Any attempt to access protected APIs requires explicit user consent.

5. All plugin-generated data is stored separately and wiped upon plugin deactivation.

Section 8.2: Tools

NOVA AI chat utilizes Marked.js to display markdown formatted strings [17]. Raymond Hill's uBlock Origin is included as part of the NOVA suite of add-ons [3].

Section 8.3: Programming Languages

Extensions utilize Javascript for interfacing with the browser. On top of this, extension UI utilizes HTML and CSS. Rust source code was edited to enable Ollama server and CORS proxy at launch, for interfacing with AI chat extension.

Section 8.4: Use of Open Source Code

This project makes use of the source code of the development build of Mozilla Firefox, called Nightly [6]. NOVA Webshark extension leverages a fork of Webshark, a browser-based packet tracer meant to mirror the functionality of Wireshark [18]. To enhance backend packet capture capabilities, the project also integrates tcpdump and libpcap which enable low-level network packet collection and filtering directly from the host system[20].

Section 8.5: New Tools and Knowledge

Name	Type	User	How it was learned	Where was the tool applied
Webshark	Tool	Roshni Varkey	Read documentation and integrated via API with browser settings	Packet Tracing Monitor feature
Gemma 3	Machine Learning Model	Braden Norum	Reviewed Hugging Face documentation and experimented with local inference	AI Chat Module
Plugin Sandboxing	Knowledge	Braden Norum and Roshni Varkey	Researching browser architecture and security guidelines	Plugin Manager Module

Chapter 9: Project Implementation and Results

Section 9.1: Project Goal Changes

During the course of the project, several key goal changes were made to adapt to technical limitations, user needs, and performance considerations:

- **Shift from Full Browser Build to Extension-Based Model**:
  Initially, the plan was to develop a standalone browser from the ground up. However, due to the complexity of building and maintaining a full browser engine, we shifted to developing a privacy-centric browser as an extension on top of Firefox, leveraging its Gecko engine and existing security infrastructure.
- **Integration of WebShark through Docker Instead of Native Embedding**:
  Instead of integrating WebShark packet analysis tools natively into the browser, we opted to run WebShark in a Docker container and access it through a browser extension. This allowed us to preserve the functionality of advanced network analysis while simplifying cross-platform deployment.

- **AI Assistant Optimized for Local Execution**: Originally, we considered server-based AI interactions, but switched to a locally processed AI assistant using gemma models to ensure full user data privacy and offline functionality.

Section 9.2: Sprints

Section 9.2 details the breakdown of goals and tasks for each team member during each Sprint.

Section 9.2.1 Sprint 1

Sprint goal: Implement basic functionality, including incognito mode activation, ad-blocker integration, and the browser's core user interface.
Sprint demo goal: Deliver a working demo that demonstrates incognito mode functionality, ad-blocker activation, and a functional rendering engine.
Starting date and period:January 28th-February 25th
Sprint master: Braden Norum

Table 9.2.1: Sprint 1 Backlog - Braden

Number	Task	Original Estimate	Total actual hours
1	Set up version control and repository (Github)	2	5
2	Install, troubleshoot, and build the browser framework	10	4
3	Develop initial UI structure (tabs, search bar, basic navigation)	4	5
4	Implement session isolation to prevent cookies and history tracking	5	6
5	Implement a privacy-focused startup configuration	4	7

Table 9.2.2: Sprint 1 Backlog - Roshni

Number	Task	Original Estimate	Total actual hours
1	Set up Jira page	2	2

2	Install, troubleshoot, and build the browser framework	10	5
3	Implement settings dropdown and settings page UI	3	3
4	Develop Incognito mode UI and Toggle Button	3	3
5	Ensure cache, cookies, and local storage are disabled in incognito mode	5	4

Section 9.2.2 Sprint 2

Original sprint goal:implement advanced security details, like removal of telemetry, any tracking blocking not covered in previous sprint, packet tracing, and plug-in support.
Original demo goal:Deliver a working demo which demonstrates the packet tracer, and shows evidence of the removal of telemetry and tracing.
Starting date and period:March 2nd to March 30th
Sprint master: Roshni Varkey

Table 9.2.3: Sprint 2 Backlog - Braden

Number	Task	Original Estimate	Total actual hours
1	Research and begin implementing telemetry removal	5	5
2	Replace Base Firefox UI and Branding with NOVA branding	1	2
3	Complete telemetry removal	7	8
4	Start UI development for traffic monitoring tool	5	5
5	Begin AI chat research	4	4
	Finish traffic monitoring UI	5	5

Table 9.2.4: Sprint 2 Backlog - Roshni Varkey

Number	Task	Original Estimate	Total actual hours
1	Research best approaches for blocking advanced tracking scripts	5	5
2	Implement & test tracking script blocking	3	3
3	Develop & integrate packet tracing functionality	7	9
4	Begin AI chat research	4	4
5	Develop initial plug-in support	4	4
	Prepare sprint demo	1	1

Section 9.2.3 Sprint 3

Original sprint goal:Finish any remaining goals, focusing on AI chat and custom commands.
Original demo goal:Deliver a fully functioning build of the browser with all planned features.
Starting date and period:April 2nd-April 28th
Sprint master: Braden Norum

Table 9.2.3: Sprint 3 Backlog - Braden

Number	Task	Original Estimate	Total actual hours
1	integrate gemma base for AI chat	10	5
2	finish AI chat ui	5	4
3	Develop Method of Creating Custom Commands	5	5
4	Develop User Preferences and Profiles Functionality and UI	5	6
5	Testing and Debugging	5	7

Table 9.2.3: Sprint 3 Backlog - Roshni

Number	Task	Original Estimate	Total actual hours

1	Initial UI mockups for AI chat	5	3
2	finish integrating LLaMa base for AI chat	5	6
3	Develop Method of Executing Custom Commands	5	5
4	Develop User Preferences and Profiles Functionality and UI	5	5
5	Testing and Debugging	5	6
6	Polish UI and prepare sprint demo	2	2

The results of each Sprint were evaluated through sprint demos that demonstrated the fully functional features completed during each development cycle. All sprint goals were successfully achieved, and the demos reflected 100% completion for each planned task. Across all sprints, progress remained steady and goals were consistently met. The project resulted in a feature-complete browser that met its original specifications.

Section 9.3: Testing Results

Section 9.3 describes the results of each test case.

Test Case number and name	**Test type**	**Pass/Fail**
T1: Enable Incognito	Functional	Pass
T2: Ad-blocker Activation	Functional	Pass
T3: AI Chat	Functional	Pass
T4: Plugin Installation	Integration	Pass
T5: Page Load Time	Performance	Pass
T6: Security Validation	Security	Pass
T7: Crash Recovery	Robustness	Pass

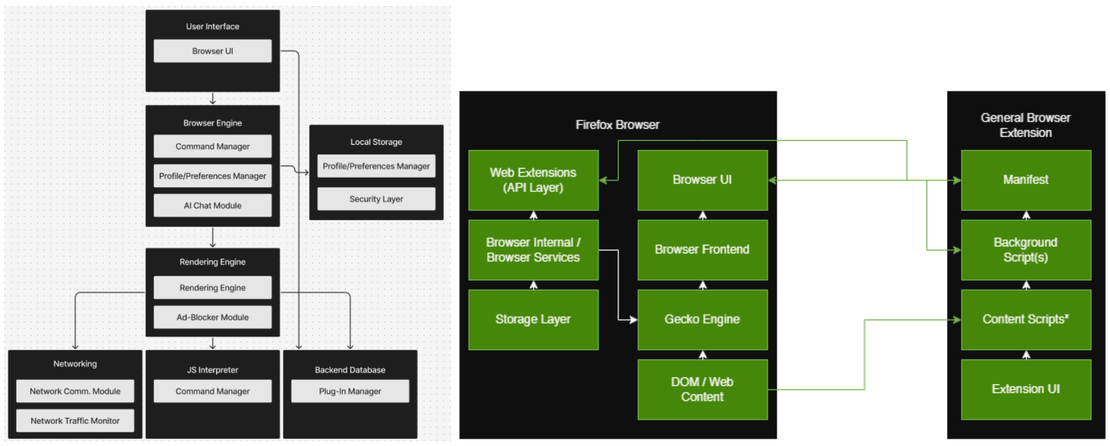
Chapter 10: Problems and Changes

Section 10.1: Project Goal Changes

While planning the project, we anticipated that a strong approach would be to implement many of the planned functionalities directly into the browser source code itself. Ultimately, this was changed in order to use an extension-based architecture, where each component is added using an extension of the browser. This allowed for fast, iterable implementation, and for fast and simple unit testing.

NOVA was initially planned to have a local history saving functionality, but it was cancelled in favor of other features. Mozilla's privacy policy states that no data is harvested from your search history, so the importance of the task was significantly diminished [19].

Section 10.2: Design Changes



Above is the initial project design (left), and the current project design (right) diagrams. Functionally, these two diagrams are the same, but less focus is displayed on the browser framework, because it was directly edited only minorly. The new system design goes into detail about the relationship between the browser and a generic extension used to build on top of the browser source code, rather than directly edit it.

Chapter 11: Financial Budget

NOVA did not require any purchases or expenses.

Chapter 12: Conclusions

The primary goal of our senior design project was to develop a privacy-centric, multi-platform browser built on top of Firefox, with a focus on user empowerment, security, and modern convenience. We aimed to go beyond traditional "incognito" modes by introducing a true private browsing experience, one that actively removes telemetry, blocks trackers, and minimizes data retention. In addition, we sought to integrate cutting-edge features, such as a customizable AI-powered assistant, network traffic analysis and plugin management all within a secure, stable, and user-friendly environment.

Over the course of the year, we successfully implemented a working browser extension with these core features. Our project utilized a wide range of technical tools, including JavaScript, Rust, Python, Docker, and Mozilla's WebExtension APIs. We applied agile development strategies across multiple sprints, adapting to challenges like hardware limitations, cross-platform compatibility, and asynchronous integration of advanced tools.

This year-long experience not only allowed us to apply and deepen the skills we acquired throughout our degree such as full-stack development, data communication and networking, and cybersecurity but also pushed us to explore new technologies and problem-solving approaches in a real-world team environment. Most importantly, we learned how to scope and deliver a technically ambitious project from the ground up. This project has been a meaningful capstone to our academic journey and a foundation for future work in web and mobile engineering.

Reference List:

1. "How Fast Should a Website Load in 2024?" Pingdom Tools. Available: https://www.pingdom.com
2. A. Balachandran et al., "Resource Monitoring in Browsers," Google Chrome Dev, 2021. Available: https://www.google.com/chrome/
3. "uBlock Origin: An Efficient Ad-blocker," GitHub Project Documentation. Available: https://github.com/gorhill/uBlock
4. TensorFlow.js Overview, "Real-Time AI Model Execution in Browsers," TensorFlow Documentation. Available: https://www.tensorflow.org/js
5. SQLite Documentation, "Database Storage Solutions for Lightweight Applications." Available: https://www.sqlite.org
6. Firefox Developer Documentation, "Cross-Platform Development Practices." Available: https://firefox-source-docs.mozilla.org
7. "Minimum Network Speeds for Web Applications," SpeedTest Global Index. Available: https://www.speedtest.net
8. Mozilla Developer Network, "Optimizing Browser Memory Usage." Available: https://developer.mozilla.org
9. "Encryption Standards: TLS 1.3 Overview," Cloudflare Documentation. Available: https://www.cloudflare.com/learning
10. "Data Integrity in Modern Web Applications," OWASP. Available: https://owasp.org
11. "Best Practices in Service Availability," ITIL Framework Overview. Available: https://www.axelos.com
12. "Securing User Authentication," NIST Digital Identity Guidelines. Available: https://www.nist.gov
13. MySQL Overview, "MySQL." Available: https://www.mysql.com/
14. Adguard AdBlocker, "AdGuard – The world's most advanced ad blocker!" Available: https://adguard.com/
15. Wireshark Homepage, "About Wireshark." Available: https://www.wireshark.org/
16. Telerik Fiddler, "Fiddler Everywhere." Available: https://www.telerik.com/fiddler
17. Christopher Jeffrey, "Marked.js." Available: https://marked.js.org/
18. QXIP, "Webshark." Available: https://github.com/QXIP/webshark
19. Mozilla, "Privacy Policy." Available: https://www.mozilla.org/en-US/privacy/
20. tcpdump and libpcap. Available: https://www.tcpdump.org/