CS 411

Project 2

BROWSIFY

İlker ÖZGEN 21902719

Onurcan ATAÇ 22002194

13.12.2023

TABLE OF CONTENTS

1 CURRENT WEB BROWSERS IN THE MARKET

1.1 Introduction

In this section, some of the currently available web browsers will be discussed. These web browsers include Edge, Chrome, Safari, Firefox and Opera.

1.2 Problem Definition and Domain

Understanding the problem definition and domain of currently used web browsers is crucial for the development of Browsify. Web browsers operate in a domain in which user experience, security, data privacy, performance, and reliability are critical aspects. A good web browser should enable the users to navigate between different web pages smoothly. Web browsers are created in response to the problem of navigating between URLs. Many browsers also incorporate additional features like bookmarking capability and additional customization options. Some browsers like Safari and Edge focus on the operating system they are running in. Some of them, such as Chrome, give importance to performance and speed, while others prioritize security and data privacy.

1.3 General Evaluation of Browsers, Their Major Technical Problems, and Proposed Solutions

1.3.1 Edge

Since Microsoft Edge utilizes Google's Chromium Engine, it provides close but worse performance in terms of RAM usage and CPU utilization than Google Chrome. In terms of speed, it is close to Google Chrome as well.

A common complaint about Microsoft Edge is the unwanted "Your browser is managed by your organization" label on the browser. The browser can be restricted by other applications, and it is time-consuming to get rid of those restrictions.

Moreover, Microsoft Edge is not open source, which creates privacy concerns among users. Many users change to other browsers since they think Microsoft is not completely transparent with the data they keep from the users.

Microsoft Edge also lacks features such as an inbuilt VPN and adblocker, which users appreciate.

Solutions to the problems of Microsoft Edge can be suggested as optimization in terms of performance, making the browser open-source to gain user trust. Adding reliable and inbuilt VPN, adblocker services might also attract users.

1.3.2 Chrome

Google Chrome uses its own web engine Chromium as well, which has the largest market share among web engines. Chrome presents the best performance in speed, CPU usage, and RAM usage among all of the browsers.

Easy integration of Google accounts with websites and services gives Google Chrome an advantage compared to other browsers.

The largest extension library also belongs to Google Chrome, which greatly helps with increasing the usability and functionality of the browser.

Google Chrome does not provide inbuilt and reliable adblocker or VPN services, which are some of the valuable features that Opera and Brave provide. Therefore, that becomes a drawback for Google Chrome.

Google Chrome is also closed-source, and therefore it raises privacy concerns among the users especially since Google is a “data company”. Just because of that reason people look to switch browsers.

Solutions to the problems of Google Chrome can be suggested as making the browser open-source to gain user trust, adding reliable and inbuilt VPN, and adblocker services.

1.3.3 Safari

Safari outperforms its competition significantly in macOS in terms of CPU usage, RAM usage, and battery conservation. However, it is extremely optimized for macOS and is not available on other platforms. Because of that, the candidate user pool decreases significantly. Consequently, there is a performance and compatibility trade-off as well.

Since it uses Apple’s own WebKit engine, it gains reliability among the community. However, because of the same reason that it does not use Chromium, it lacks many valuable extensions. Moreover, it is also unique in terms of its interface. Since all of the Chromium-based browsers are extremely similar, users may have a hard time getting used to Safari.

The biggest advantage and disadvantage of Safari is that it is only made for Apple users. This enables them to provide better performance, but they lose an important percentage of users. For Apple users, adding some of the significant extensions of Chromium to Safari is important.

1.3.4 Firefox

Firefox presents an option for Windows/Linux users who do not want to use Chromium since it is the only mainstream browser that does not depend on the Chromium web engine. However, it is slower compared to the Chromium-based engines in speed tests. Moreover, website owners usually optimize their web pages according to Chromium because of its market share. Therefore, users report that sometimes web pages do not load properly or come up as “clunky”.

Users also report that changing profiles in Firefox is also an issue, and it is too low-level for an ordinary user. Furthermore, there are features that are seldom used by the users such as Firefox View.

One of the biggest problems of Firefox is hard customization and account management for unknowledgeable users because of the user interface of some pages. These problems can be solved by managing these processes with better user interfaces and more convenient processes.

1.3.5 Opera

Opera is a popular browser nowadays since it has a market strategy towards gamers and therefore young people. It comes with some extremely useful features such as a built-in ad-blocker, built-in VPN, automatic picture-in-picture feature when switching tabs, and a sidebar to easily reach messaging platforms without opening a complete tab. It is also more customizable by the users.

However, users report frequent bugs and glitches while using the browser which isn't the case in most of its competitors. Users also have a privacy concern since Opera uses the Chromium web engine, it is not open-source, and it is owned by a Chinese company.

Opera is up and coming with its unique VPN, adblocker features, and customizability. However, frequent bugs and glitches that are reported by the users need to be solved. The performance optimization for Opera is also crucial.

1.4 Stakeholders

There are various stakeholders for web browsers, each having distinct expectations and contributions. Understanding these stakeholders is essential for developing a browser like Browsify that satisfies stakeholder needs and industry standards.

1.4.1 Users

Users are the end consumers who utilize the application for their specific needs. They prioritize user-friendly interfaces, performance, and security. Users expect a user interface with easy navigation and a good visual experience. They also want a browser with fast performance. Finally, they want a secure web browser that prioritizes data privacy. Users contribute to the project by providing feedback on their experiences, reporting the bugs they encounter, and making suggestions on enhancing the application.

1.4.2 Developers

Developers are the creators of the web browser and also the creators of web content since they also implement web pages. Developers rely on web browsers to have support for the latest standards and technologies so that their web applications can work consistently. They also want web browsers to include various developer tools, such as debugging, for effective web development. Developers also expect extension capabilities in the web browsers for enhanced functionality. They contribute to the project mainly by fixing bugs, updating, and testing the web browser. They may also contribute, as web page developers, by creating extensions and identifying and reporting bugs.

1.4.3 Enterprises

Enterprises are organizations that utilize web browsers across their internal networks. They give importance to security. They desire a secure web browser to protect sensitive data. They also expect compatibility with enterprise applications. They contribute to the project by providing feedback.

1.4.4 Regulatory Bodies

Regulatory bodies are entities that set standards and regulations for online privacy and security. They expect the web browsers to comply with their data protection regulations. They contribute to the project by enhancing user security and privacy by setting these regulations.

1.5 Pros and Cons of the Products

1.5.1 Edge

Pros:

- ➢ Integration with Microsoft services
- ➢ Advanced PDF Customization and Voiceover Options

Cons:

- ➢ Worse performance compared to Chrome
- ➢ Poor collection of extensions
- ➢ No inbuilt VPN or ad-blocker
- ➢ Not open source, privacy concerns

1.5.2 Chrome

Pros:

- ➢ Integration with Google account
- ➢ Extensive collection of extensions
- ➢ Good tab management system
- ➢ Fast performance

Cons:

- ➢ Considerable consumption of system resources
- ➢ Poor privacy features

- ➢ No inbuilt VPN or ad-blocker
- ➢ Not open source, privacy concerns

1.5.3 Safari

Pros:

- ➢ Integration with Apple services
- ➢ Good privacy features
- ➢ Fast performance on Macs

Cons:

- ➢ Limited customizability
- ➢ Only available for Apple products
- ➢ Lacks extensions

1.5.4 Firefox

Pros:

- ➢ Customizability for knowledgeable users
- ➢ Synchronization features
- ➢ More trust from the user in terms of security, since it is open source and Mozilla is not a data company

Cons:

- ➢ Slow performance, high RAM usage
- ➢ Non-Chromium based, websites can seem and load “clunky”
- ➢ Non-used Features such as Firefox View
- ➢ Hard to customize for unknowledgeable users

1.5.5 Opera

Pros:

- ➢ Customizability
- ➢ Built-in VPN and ad-blocker
- ➢ Sidebar for messaging platforms

- ➢ Automatic picture-in-picture when tabs are switched while a video is playing
- ➢ Synchronization features

Cons:

- ➢ Relatively slower performance
- ➢ Bugs and glitches are reported by the users while loading pages

2 SOFTWARE ARCHITECTURE OF BROWSIFY

2.1 Introduction

Browsify is a new web browser designed to address the limitations of other web browsers, particularly in bookmarking capabilities. It prioritizes easy navigation to provide a good user experience to its users. Browsify matches most of the navigating ability of its competitors, while also providing advanced bookmarking features. Besides being a reliable and private application, it provides seamless navigation and high-echelon performance.

Browsify utilizes the PyQT5 library to reach and use the Chromium web engine and for most of its GUI components.

2.2 Quality Attributes and Main Requirements

As indicated in the project document, the main quality attributes for Browsify are security, data privacy, high performance, and reliability.

Data Privacy and Security: In order to clear off data privacy concerns, Browsify uses a mostly open-source library in PyQT5 and an open-source web engine in Chromium. As most of the data privacy concerns originate from the browsers and/or their web engines being closed-source, Browsify takes this approach. Moreover, Browsify stores data only in the local storage. Therefore, users do not have to rely upon outside servers for their data. They can completely control the privacy of their data in their local storage. Browsify also makes sure to open web pages with the “https” protocol if available to ensure privacy.

High Performance: Since Browsify implicitly uses Chromium as its web engine, and Chromium is one of the fastest and widely-used web engines, Browsify is fast, and it navigates seamlessly on web pages. The library PyQT5 does not add a significant overhead on Chromium, and therefore, Browsify stays high performance.

Reliability: Browsify is tested on all of its features, and it does not cause any bugs or inconsistencies. Even though it does not provide functionality to download data from the internet, it is highly reliable considering its features.

Main Requirements: Main requirements are recognized as in the project document:

- Multiple tabs concurrently open.
- Back and forward buttons.

- A reload and a stop button.
- A home button.
- An address bar that takes a URL as input and shows the URL of the webpage when visited.
- Personalized and better bookmarking capability.

Browsify meets all of these criteria and has all of the features implemented.

2.3 High-Level Architecture

Overall, the high-level architecture of Browsify is best seen through the High-Level Modular Architecture Diagram. The architecture of Browsify utilizes various views as below:

Module Styles:

- Uses Style
- Decomposition Style
- Layered Style

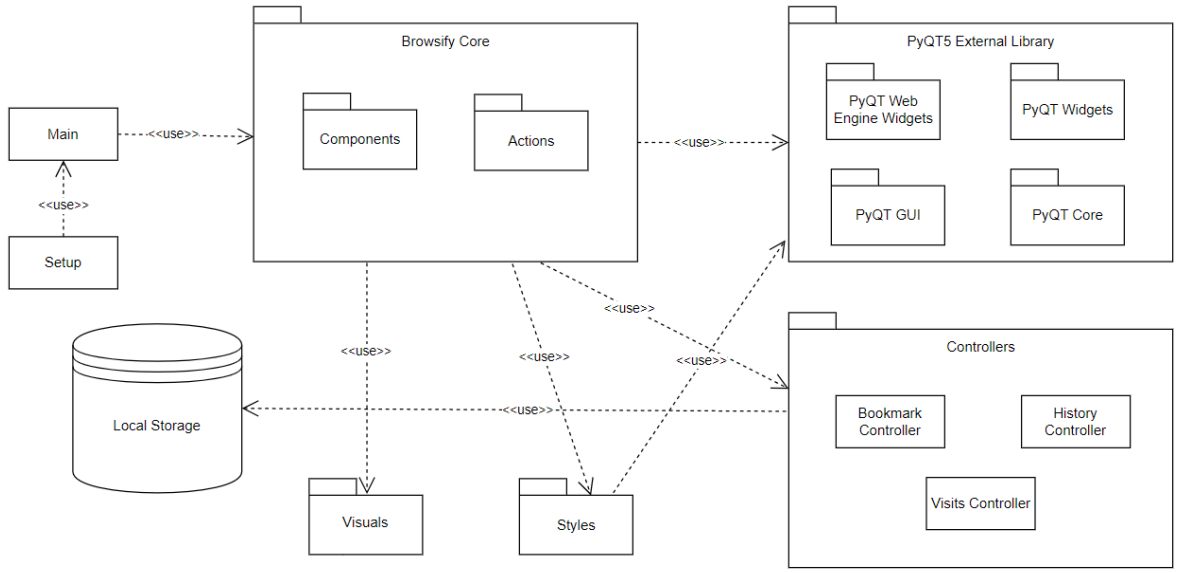
C&C Styles:

- Event-Based Style

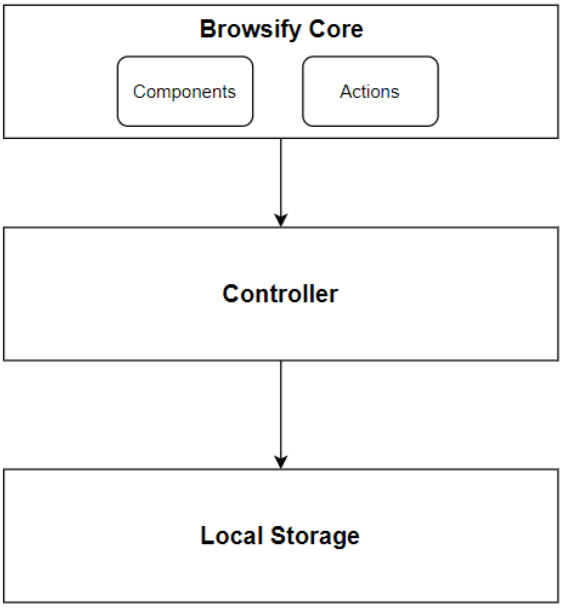
Allocation Styles:

- Install Style

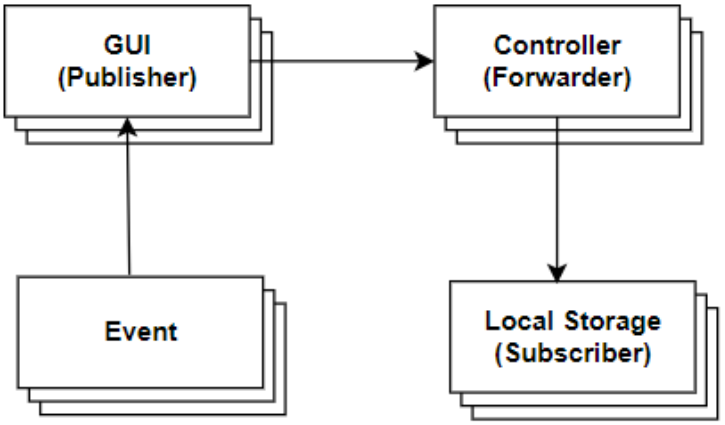
2.3.1 High-Level Modular Architecture Diagram



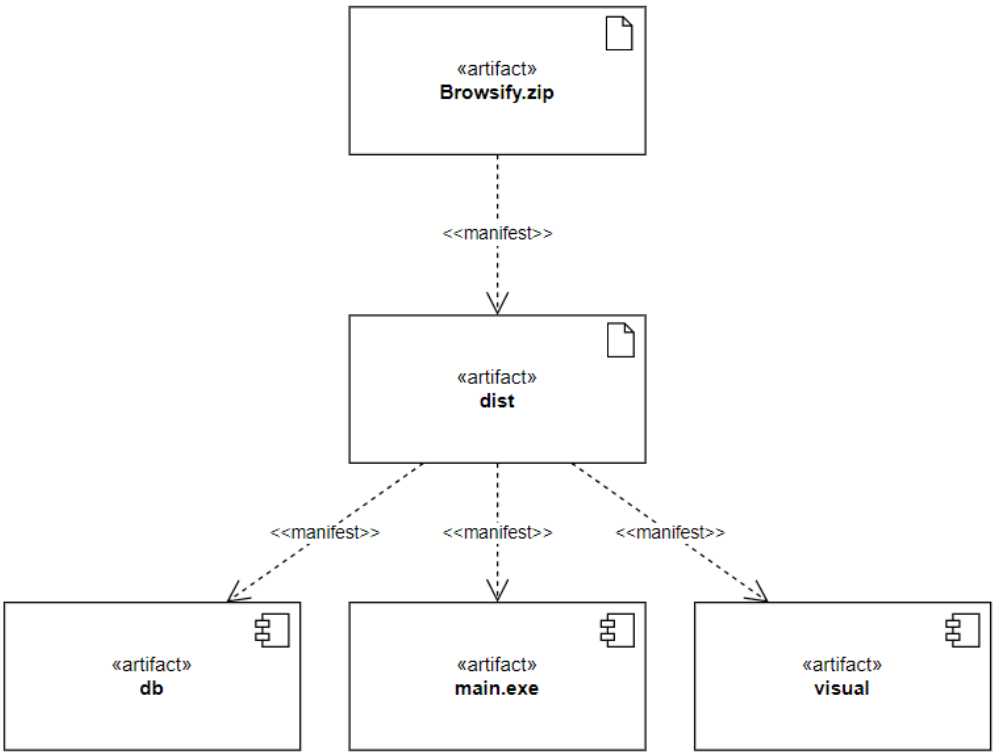
2.3.2 Layered Architecture Diagram



2.3.3 Button/Actions Event Based C&C Architecture Diagram - Pub/Sub Style



2.3.4 Allocation Diagram - Install Style



2.4 The Rationale Behind the Architecture

Browsify aims to meet user needs and quality requirements of the users while keeping a maintainable, minimal and intuitive application. Therefore, the application aims to solve the problems that are presented in efficient ways.

2.5 Architectural Patterns

Two of the architectural patterns discussed in class are used in Browsify.

2.5.1 MVC (Modal View Controller) Pattern

The MVC design pattern is used in Browsify since it makes it easier and simpler to implement, debug, and maintain the application. Therefore, we separated the front-end components, controllers, and data of our application into separate containers. This approach has helped us locate and fix the problems much easier. MVC also enables us to utilize parallel development.

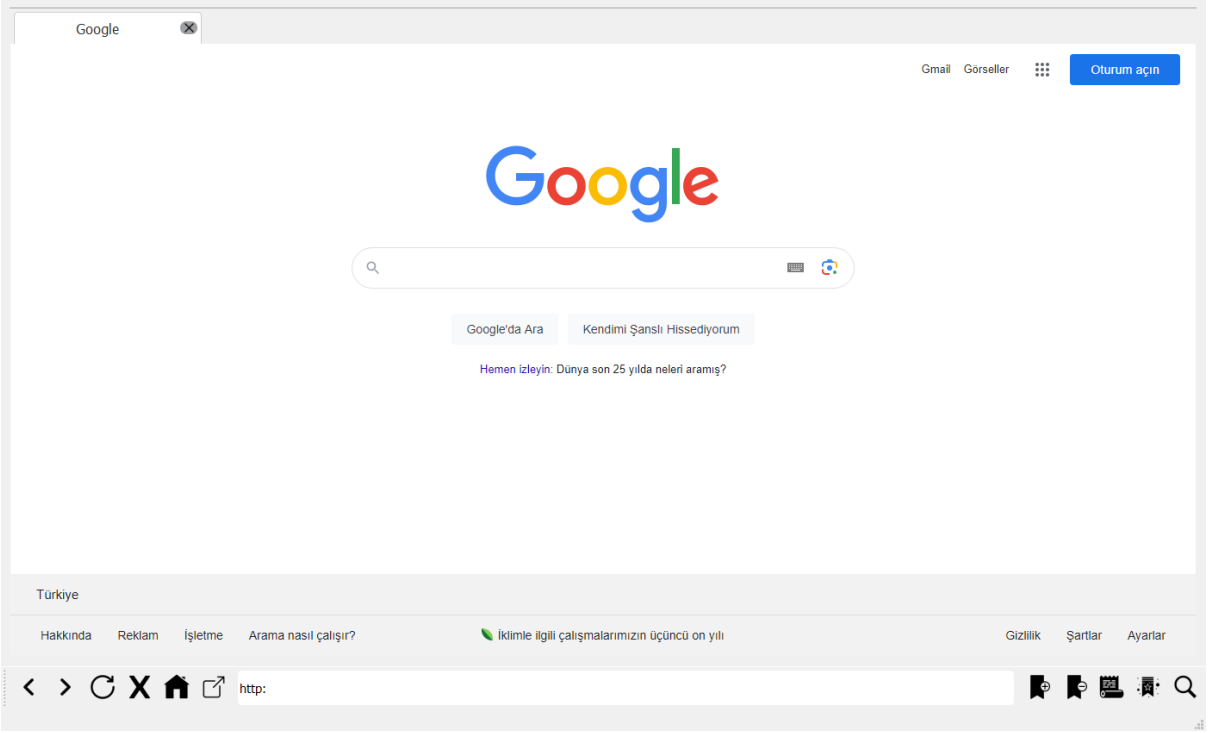
2.5.2 Monolithic Architectural Pattern

The monolithic architectural pattern is an architectural pattern where an application is built as a single, integrated unit. An instance of the monolithic architectural pattern is used in Browsify since it enables the coding process to become fast and simple. Monolithic architecture is widely used in smaller and simpler applications such as Browsify, where there aren’t many units to separate, and there isn’t a large team of developers working on the

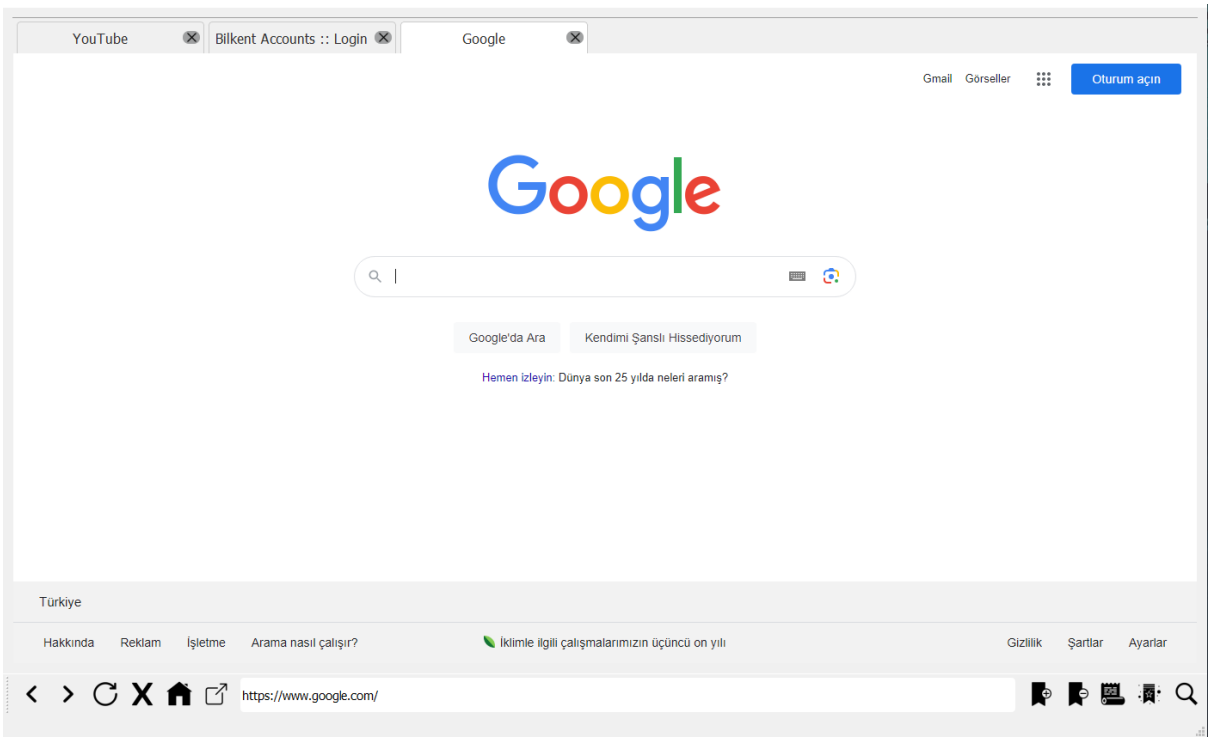
project. Monolithic architecture also enables better performance. Since one of the quality attributes of our project is high performance, monolithic architecture was the best fit. Since we have a team of 2 people, the easiness of debugging and testing a monolithic application was also a crucial factor.

2.6 Browser Implementation

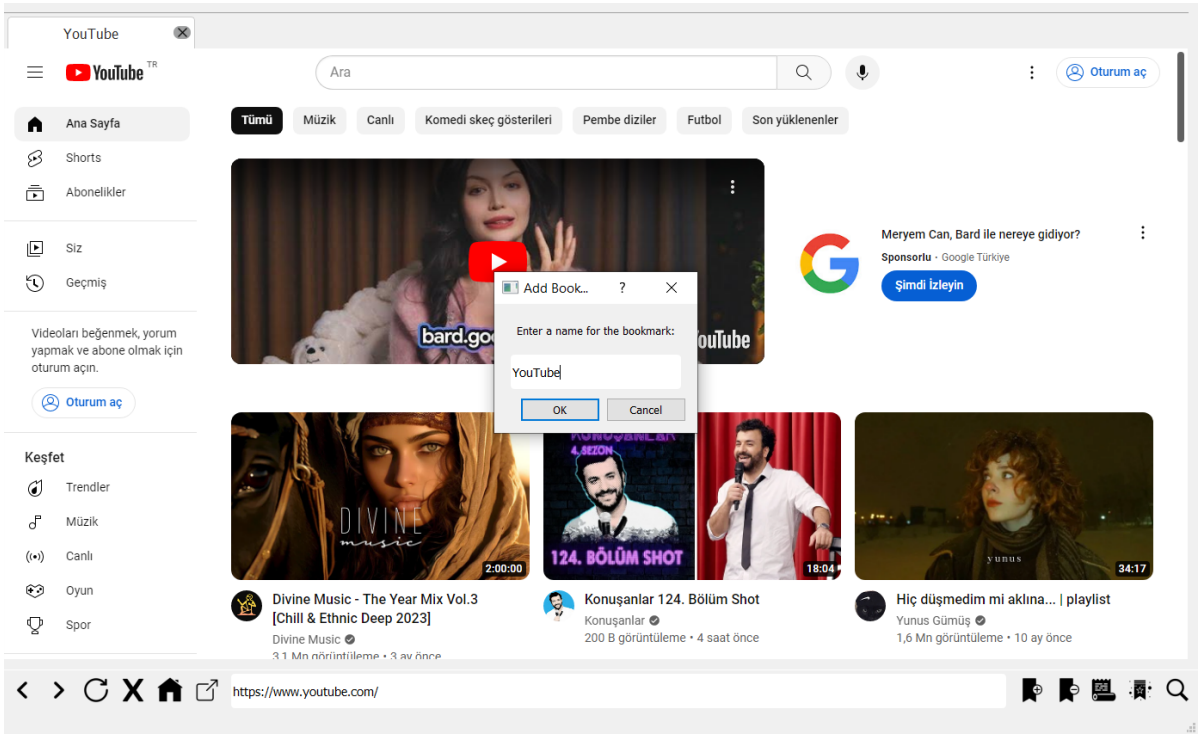
2.6.1 Application Screenshots



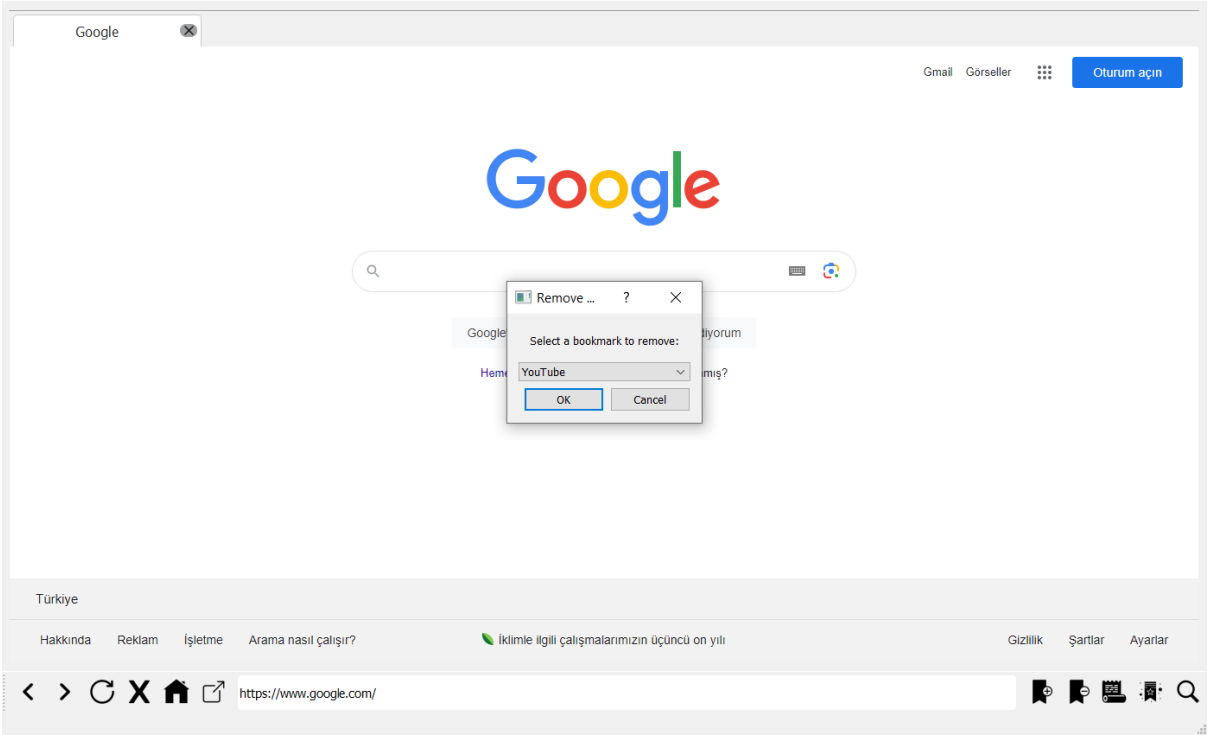
Main Page



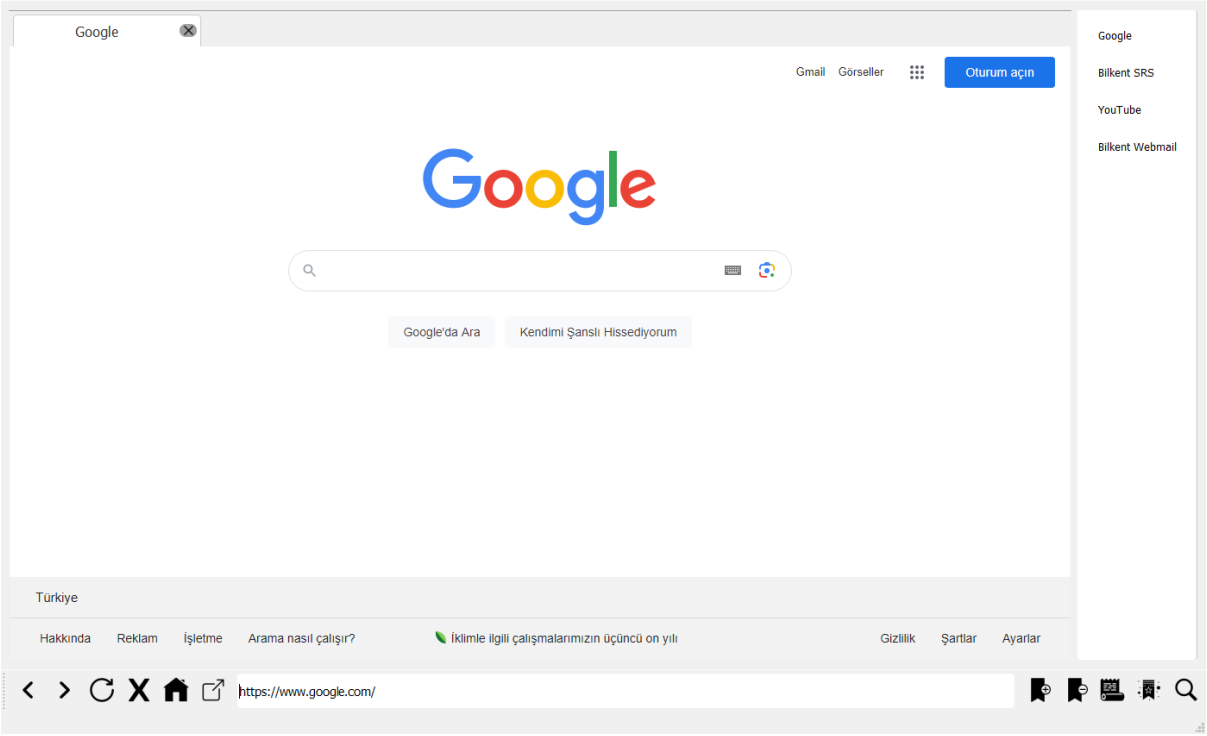
Multiple Tabs Open at the Same Time



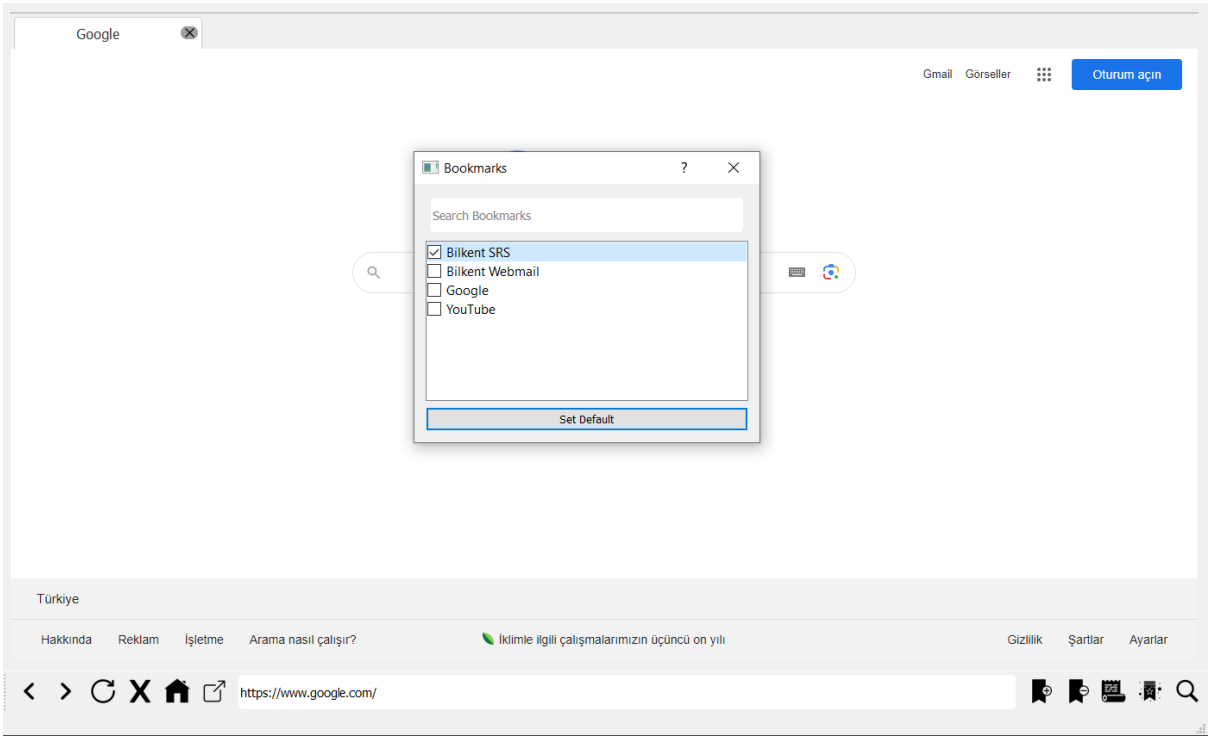
Add Bookmark



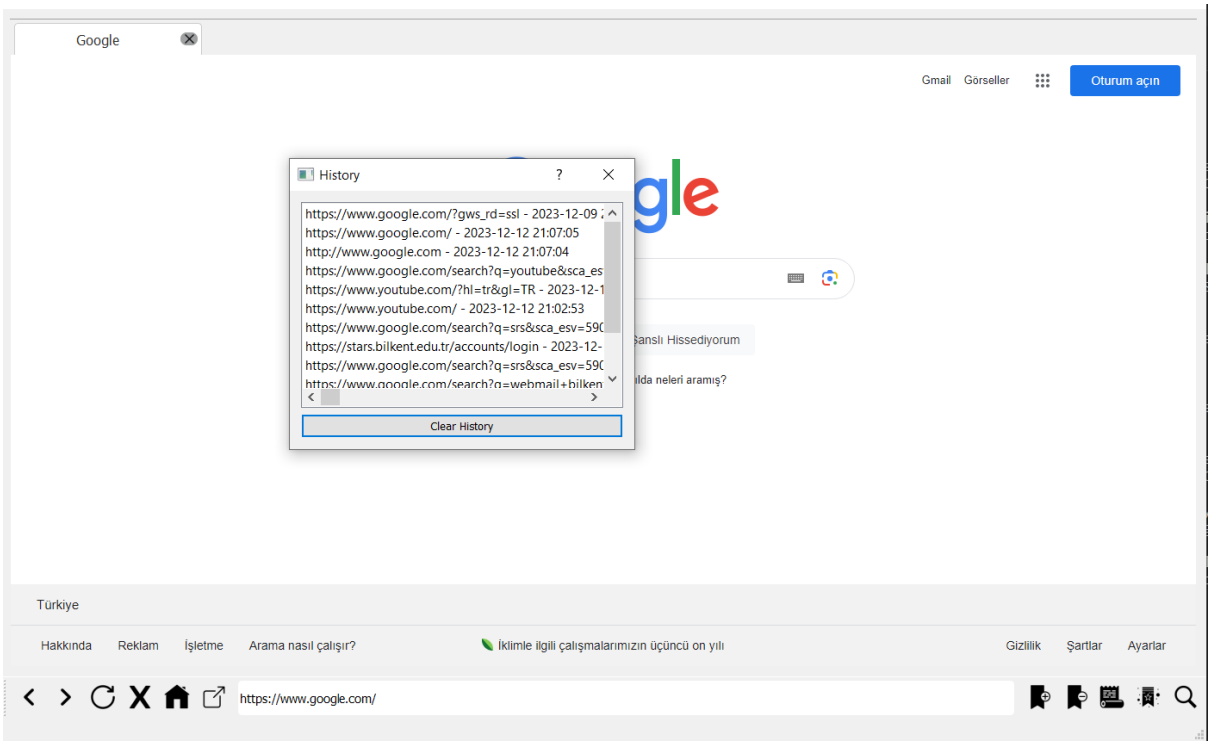
Remove Bookmark



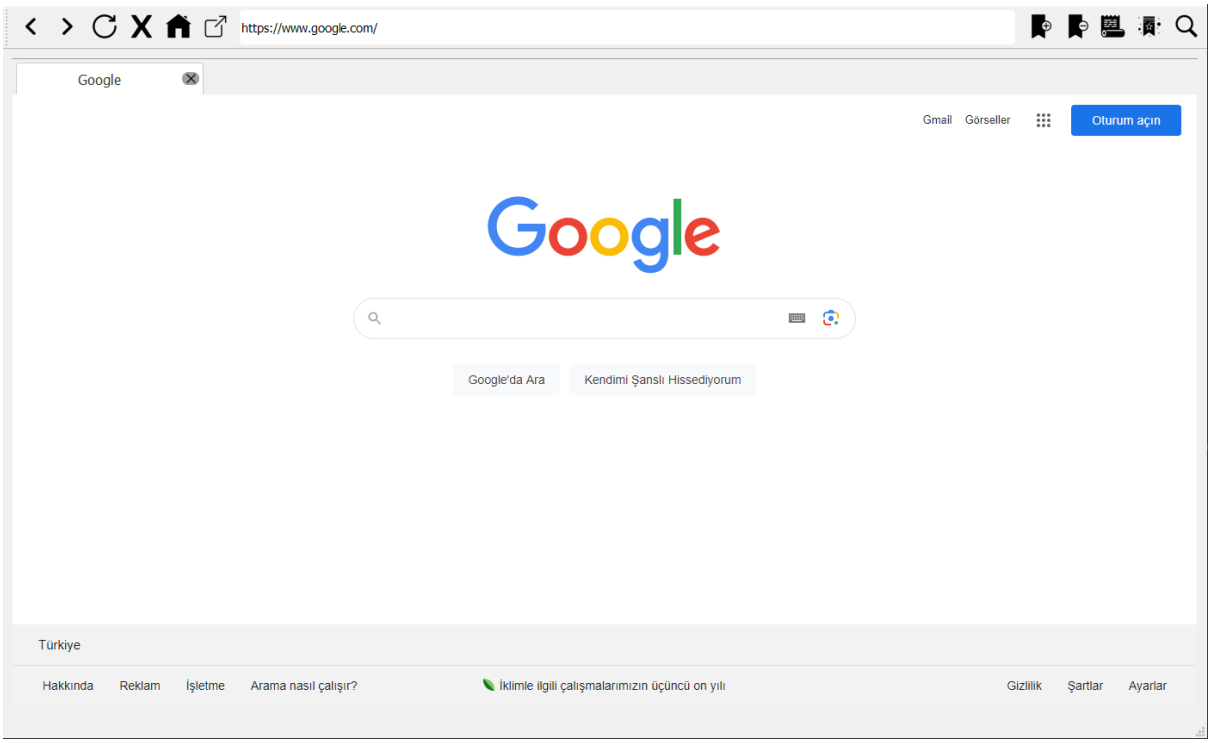
Bookmarks Sidebar



Search Bookmark



Search History



Movable Navbar for Customization

2.6.2 Implementation Screenshots

```python
import sys
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from PyQt5.QtWebEngineWidgets import *

from core.Browsify import Browsify
from controller import ControllerScript as control

def main():
    app = QApplication(sys.argv)
    QApplication.setApplicationName("Browsify")
    window = Browsify()
    control.load_bookmarks_from_file(window)
    window.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

Driver script of Browsify

```python
from setuptools import setup
import sys

# Check if the script is being run on Windows
if sys.platform == 'win32':
    sys.argv.append('pyinstaller')

setup(
    name='Browsify',
    version='1.0',
    console=['main.py'],  # Use 'console' for console applications and 'windows' for GUI applications
    options={
        'pyinstaller': {
            'packages': ['PyQt5.QtCore', 'PyQt5.QtGui', 'PyQt5.QtWidgets', 'PyQt5.QtWebEngineWidgets'],
            'exclude': ['venv'],  # Exclude virtual environment directory
        }
    },
    entry_points={
        'console_scripts': [
            'browsify = main:main',  # Change 'main' to the actual name of your main module
        ],
    },
)
```

Setup script of Browsify

```python
import unittest
from PyQt5.QtWidgets import QApplication
from main import main

class TestBrowsifyApplication(unittest.TestCase):
    def setUp(self):
        # Initialize the application instance for testing
        self.app = QApplication([])

    def tearDown(self):
        # Clean up resources after each test
        del self.app

    def test_application_runs(self):
        # Test if the application runs without errors
        main()

if __name__ == '__main__':
    unittest.main()
```

Test script of Browsify

```python
# Function to save bookmarks
def save_bookmarks_to_file(brwsr, filename='db/bookmarks.json'):
    with open(filename, 'w') as file:
        json.dump(brwsr.bookmarks, file)

# Function to load bookmarks
def load_bookmarks_from_file(brwsr, filename='db/bookmarks.json'):
    try:
        with open(filename, 'r') as file:
            brwsr.bookmarks = json.load(file)
    except FileNotFoundError:
        pass

# Function to save the history
def save_history_to_file(brwsr, filename='db/history.json'):
    with open(filename, 'w') as file:
        json.dump(brwsr.history, file)

# Function to save the visit counts
def save_visits_to_file(brwsr, filename='db/visits.json'):
    with open(filename, 'w') as file:
        json.dump(brwsr.visits, file)

# Function to save checked bookmarks state
def save_checked_state(checked_state):
    with open('db/checked_bookmarks.json', 'w') as file:
        json.dump(checked_state, file)
```

A part of the controller script of Browsify

```python
import json
from PyQt5.QtCore import *
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import *
from PyQt5.QtWebEngineWidgets import *
from PyQt5.QtCore import QDateTime, Qt

from styles.Styles import BrowsifyStyles
from controller import ControllerScript as control
```

The imports of the core module of Browsify

```
        # Add Bookmark Button
        add_bookmark_btn = QAction(QIcon('visual/icons/add.png'), 'Add Bookmark', self)
        add_bookmark_btn.setStatusTip('Bookmark current page')
        add_bookmark_btn.triggered.connect(self.add_bookmark)
        navbar.addAction(add_bookmark_btn)

        # Remove Bookmark Button
        remove_add_bookmark_btn = QAction(QIcon('visual/icons/remove.png'), 'Remove Bookmark', self)
        remove_add_bookmark_btn.setStatusTip('Remove selected bookmark')
        remove_add_bookmark_btn.triggered.connect(self.remove_bookmark)
        navbar.addAction(remove_add_bookmark_btn)

        # Add History Button
        history_btn = QAction(QIcon('visual/icons/history.png'), 'Search History', self)
        history_btn.setStatusTip('View Search History')
        history_btn.triggered.connect(self.show_history_popup)
        navbar.addAction(history_btn)

        # Add Bookmarks Sidebar Button
        toggle_sidebar_btn = QAction(QIcon('visual/icons/bookmarks.png'), 'My Bookmarks', self)
        toggle_sidebar_btn.setStatusTip('My Bookmarks')
        toggle_sidebar_btn.triggered.connect(self.toggle_sidebar)
        navbar.addAction(toggle_sidebar_btn)

        # Search Bookmarks Button
        bookmarks_btn = QAction(QIcon('visual/icons/search.png'), 'Search Bookmarks', self)
        bookmarks_btn.setStatusTip('Search Bookmarks')
        bookmarks_btn.triggered.connect(self.show_bookmarks_popup)
        navbar.addAction(bookmarks_btn)
```

A part of the UI elements of Browsify

```
    ################################
    ### Browser Utiliy Functions ###
    ################################
    # Function to return the current browser
    def current_browser(self):
        return self.tabs.currentWidget()

    # Function to update toolbar
    def update_urlbar(self, q, browser=None):
        if browser and browser == self.current_browser():
            self.url_bar.setText(q.toString())
            self.url_bar.setCursorPosition(0)
            self.add_to_history(q.toString())

    # Function to update URL bar when the current tab changes
    def update_urlbar_on_tab_change(self, index):
        current_browser = self.tabs.widget(index)
        if current_browser:
            self.update_urlbar(current_browser.url(), current_browser)

    # Function to get the URL of a bookmark
    def get_url_from_bookmark_name(self, bookmark_name):
        for url, name in self.bookmarks.items():
            if name == bookmark_name:
                return url
        return ""
```

A part of the browser utility function of the core module of Browsify

2.6.3 Application Testing

The Browsify application is tested specifically in terms of its required main features. After these features are determined to be working correctly, the application is used by us for daily tasks to observe if any unexpected crashes occur.

2.6.4 Comparison of the Application with Existing Browsers

Browsify matches the main functionalities of the existing browsers, such as going back and forward, returning to the home page, reloading, and opening new tabs.

In Browsify, a history log pop-up is also presented for the user to see the URLs of the websites that he/she visited and the times when they were visited. This feature is implemented so that a similar feature to the history feature in the existing browsers is achieved.

In Browsify, the header (the part that includes the URL bar and the main buttons), can be easily moved along the screen, up, down, right, left, or anywhere the user wants to.

Browsify shows the bookmarks in a side menu that is triggered by a button. Bookmarks can be added and removed easily by buttons. In those regards, it is similar to the bookmark systems of existing browsers such as Chrome, Edge, and Opera. However, the information on visit numbers of the webpages is kept in the local storage and is used in sorting the bookmarks by their number of visits in real time. This unique feature helps the user to reach the most visited bookmarks of his/hers easier.

In Browsify, bookmarks can easily be searched by their name using a pop-up that emerges when the search button is clicked. This feature is present in Chrome, Edge, and Opera. However, only Opera provides easy access to this feature like Browsify.

Browsify does not provide the directory structure for the bookmarks that Microsoft Edge presents. However, Browsify uniquely provides the option of setting default webpages that automatically open when the browser starts to the user. The user can therefore prepare his/her automatic workspace whenever he/sHe opens the browser. Changing these webpages that are determined as default is really easy as well. The user also has the option to not use any default web pages.

REFERENCES

[1] E. L, “The Pros and cons of popular web browsers: Firefox, Chrome, Safari, and edge,” The European Business Review, https://www.europeanbusinessreview.com/the-pros-and-cons-of-popular-web-browsers-firefox-chrome-safari-and-edge/ (accessed Dec. 13, 2023).

[2] S. Abdul, “Which browser uses the least ram and CPU on Windows, macos, and Chromeos?,” MUO, https://www.makeuseof.com/browser-windows-macos-chromeos-uses-least-ram-cpu/ (accessed Dec. 13, 2023).