

# Firefox Desktop Browser: Improving Import Usability for Users

Senior Design Final Report

California State University, Los Angeles

May 6, 2023

## **Team Members**

Alvin Lew  
Angel Villalobos  
Ani Movesian  
Brian Pham  
Bryan Macoy  
Evan Liang  
Nolan Ishii  
Portia Wu  
Steven Wang  
Zachariah Harris

## **Faculty Advisor**

Dr. Chengyu Sun

## **Project Liaisons**

Mike Conley  
Katherine Patenio  
Niklas Baumgardner  
Neil Deakin  
Gijs Kruitsbosch

## Table of Contents

<b>1. Introduction</b>	<b>3</b>
1.1. Background	3
1.2 Project Goals	3
1.2.1 Redesign the migrator’s user interface	3
1.2.2 Simplify and streamline the import process	4
1.2.3 Add additional features and support for more browsers	4
1.3 Achievements	4
<b>2. Related Work</b>	<b>5</b>
2.1. Old Migration Features	5
2.2. Development Cycle	5
<b>3. System Architecture</b>	<b>7</b>
3.1. Migration	7
3.2. Favicon Importing	8
3.3. User Interface Design	9
<b>4. Conclusions</b>	<b>11</b>
4.1. Results	11
4.1.1 New Data Imports	12
4.1.1.1 Expanded Browser Support	12
4.1.1.2 Favicons	12
4.1.1.3 Credit Card Data and Payment Methods	12
4.1.2 User Interface Results	12
4.1.2.1 Modal Windows and Localization	12
4.1.2.2 Streamlined User Experience	12
4.1.2.3 Testing	12
4.1.2.4 Permission Dialog and Error Handling	12
4.1.3 Other Project Achievements	13
4.1.3.1 ESMification	13
4.2 Impact	13
4.2.1 Enhanced User Experience	13
4.2.2 Expanded Browser Support	13
4.2.3 Favicon Imports	13
4.2.4 Codebase Modernization	14
4.3 Future Work	14
<b>5. References</b>	<b>15</b>

# 1. Introduction

## 1.1. Background

The Mozilla Foundation is an organization focused on developing free and open-source products, one of which is Mozilla Firefox (Firefox), a popular web browser known for its privacy. Although Firefox receives regular updates, certain parts of its code have received lower priority than others. In particular, the browser's migrator component—which allows users to import data from other browsers to Firefox—became outdated. While the migrator was still fully functional, it lacked user friendliness and new features, which could negatively impact user interaction and retention.

To address this issue, the Mozilla Foundation contacted Cal State LA in 2022 to sponsor a project for the university's students. As part of this collaboration, computer science students worked with experienced Mozilla engineers to contribute to Firefox's open-source code throughout the 2022-2023 school year. As a result, most of the project's changes have been released as of Firefox 114, available to both users and developers.

Before collaborating with Cal State LA, Firefox allowed users to import data (such as bookmarks and search history) from other browsers. This process is known as migration, and the component of code implementing this feature is known as the "migrator." However, because Firefox's data importing process had no critical issues, its bugs were considered low priority, with some left untouched for over 15 years. Unfortunately, it led to parts of the migrator needing to be updated. While Firefox's user interface was being updated with new themes, the migration dialogs continued to use default operating system popups. Not only did these OS-specific modals clash with Firefox's Design System, but some were unneeded, increasing users' required steps to follow.

## 1.2 Project Goals

The importance of creating a user-friendly migration process cannot be overstated. Users often need help to switch browsers because they want to avoid losing data and following complicated instructions. In order to improve the old Firefox migrator, three overarching goals were prioritized throughout FIU (Firefox Improving Import Usability):

### 1.2.1 Redesign the migrator's user interface

By implementing new designs matching Firefox's brand and aesthetic, users will see consistency in the browser's theme. Additionally, the interface can stay consistent across different operating systems embedded into the browser.

### 1.2.2 Simplify and streamline the import process

With steps condensed into fewer windows and more settings hidden with default options, users will have less work and experience a smaller cognitive load.

### 1.2.3 Add additional features and support for more browsers

More support for different browsers will target a wider audience, and allowing users to import more data will ensure a more seamless transition from their old browsers.

## 1.3 Achievements

FIIU accomplished most of its goals while leaving room for future additions. Its significant contributions to Firefox can be summarized as follows:

- 7 new migration UI elements are implemented
- Data from 3 new browsers can be imported
- Favicons for web pages can be imported for 3 new browsers
- The overall import process is simplified
- Over 50 minor bugs are fixed

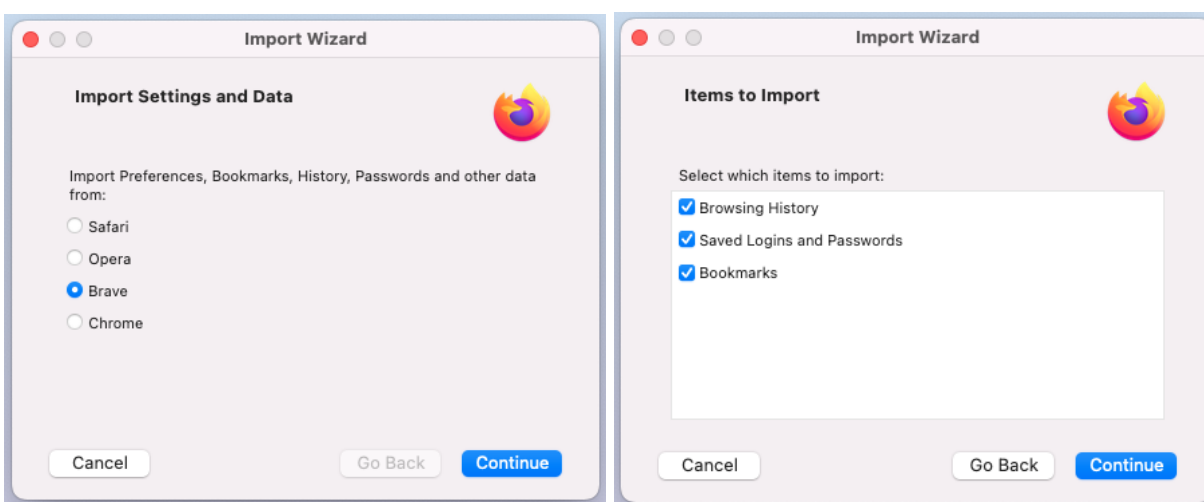
All the project's source code is thoroughly tested to ensure seamless integration with the existing browser and approved by experienced Mozilla engineers using Phabricator, a code review software. Additionally, it can be found online in Firefox's public repositories through code indexing tools such as Searchfox.

The specific bugs that FIIU completed can be found here on Bugzilla, Firefox's bug tracking system: [https://bugzilla.mozilla.org/show\\_bug.cgi?id=1788748](https://bugzilla.mozilla.org/show_bug.cgi?id=1788748).

## 2. Related Work

### 2.1. Old Migration Features

Before FIIU, Firefox already had basic migration features for most major browsers—Safari, Internet Explorer, Edge, Chrome, and Brave. Such features usually included bookmarks and history. However, the migration user interface was never fully implemented by Firefox. Instead, the browser used default operating system popups, leading to an outdated look. For macOS and Windows, the migration menu consisted of radio buttons for selections of browsers to import from. The following window contained a list of checkboxes allowing users to select which data they wanted to import.



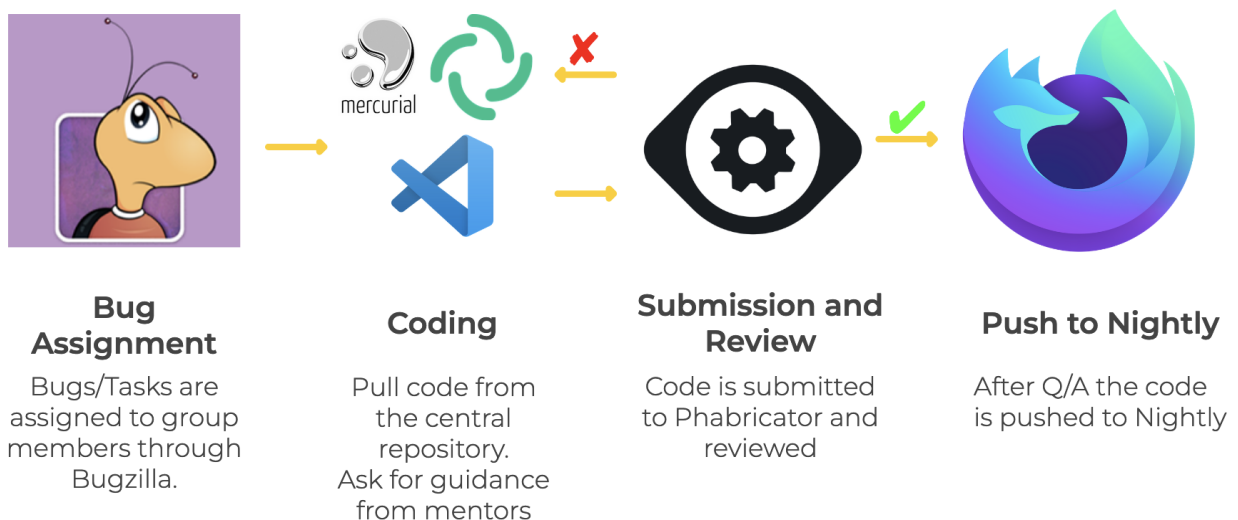
*Old Migration Process (macOS)*

Since the old migration process worked, it saved users time compared to manually transferring data. However, it lacked support for newer browsers and data types, such as OperaGX and Autoform data. Additionally, its design did not fit the general Firefox theme and included steps that could have been shortened.

### 2.2. Development Cycle

It is worth noting that FIIU differed from traditional Senior Design projects. Typically, student teams work on relatively small projects and can be flexible in their design choices and technologies. It allows more freedom to decide on collaboration techniques and workflows, where students can use what they learn from classes directly. FIIU, on the other hand, worked on top of a large-scale, established, distributed open-source project. Firefox had over 200 million users worldwide and thousands of developers contributing code during the project. Thus, FIIU utilized Firefox's existing best practices for development.

1. Anyone may file bugs inside the Bugzilla system with a free online account. Each bug represents a task which can include feature requests, issues found by users, code defects, and more.
2. Experienced Mozilla engineers organize bugs with hierarchies and higher-level objectives called “meta-bugs.” Meta-bugs are groups of related normal bugs, and one was created specifically for FIIU.
3. The normal bugs under FIIU’s meta-bug were assigned to team members. However, since Firefox is open-source, other engineers would occasionally help complete tasks under FIIU’s meta-bug.
4. FIIU’s team members worked on their assigned bugs using various development tools. On average, 6-8 hours of work per week was expected.
5. Throughout the process, team members communicated with Mozilla mentors using Zoom, Element, and email. When students ran into development-related issues, Mozilla mentors provided generous help, depending on which mentor was available.
6. After a bug was completed, team members submitted the changes to Phabricator—there, senior Mozilla developers reviewed the code and provided feedback. It ranged from FIIU’s designated liaisons to quality assurance teams and other engineers at Mozilla.
7. Based on the code review, team members would continue working on their bug (repeating step 4) or have their submission accepted. Larger, more complicated bugs typically require multiple rounds of code review.
8. Once a patch (the code for a bug) was accepted, it was marked as complete and merged into the central Firefox repository. Initially, the code is integrated into Firefox Nightly, which is an experimental version of Firefox that precedes the public release version by two versions. After approximately two months, the new features introduced by the FIIU project will be incorporated into the official Firefox release.

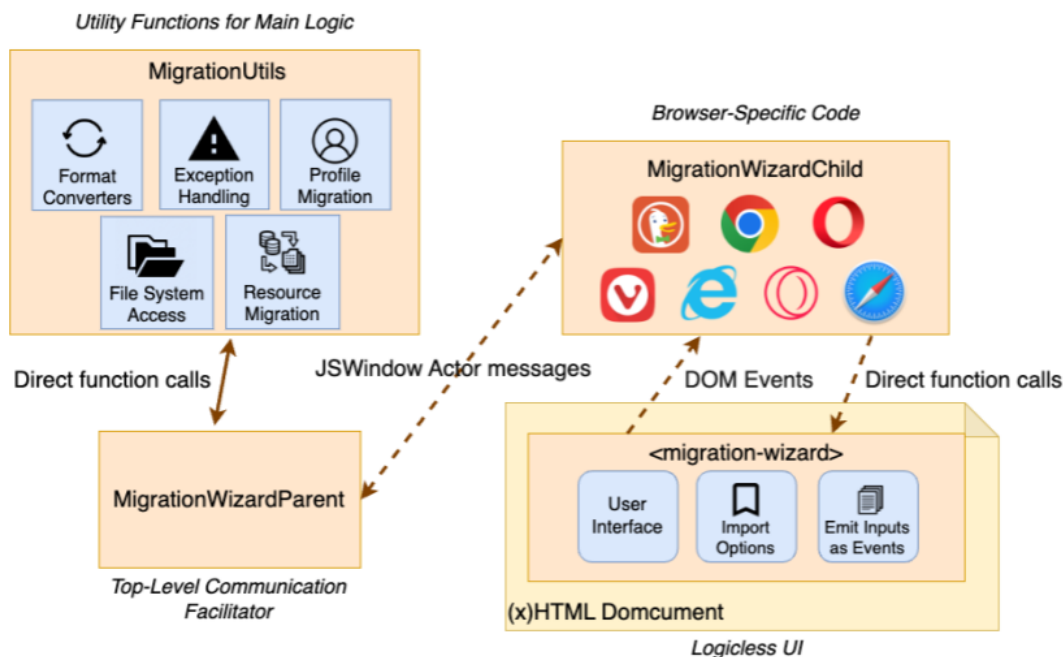


## 3. System Architecture

### 3.1. Migration

The top level of Firefox's migrator is the *MigrationWizardParent* component. This parent facilitates communication between an actual browser-specific migrator and the *MigrationUtils* module. The corresponding *MigrationUtils* class contains most of the logic for importing data and useful functions reused among browser-specific migration modules. Meanwhile, browser-specific migrators extend the shared prototype class *MigratorBase*. This ensures that each migrator will contain certain things, such as a unique key, a localization ID, and a *getResources* function to return an array of migration resource objects representing user data. When all necessary functions are implemented, the *MigrationWizardParent* can execute the migration process while utilizing logic from *MigrationUtils*.

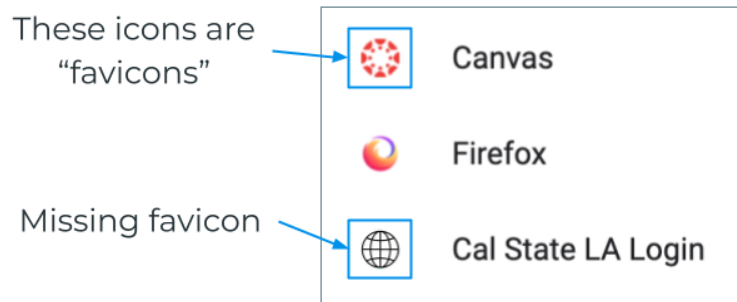
To fetch user data from browsers, developers must first discover where it is stored. While this differs for each browser, most major ones (including Google Chrome and Safari) keep information in files on a user's local device. Each browser-specific migrator would read from these files and convert the data into JavaScript objects. Functions from *MigrationUtils* can then populate the data into Firefox's own data storage mechanisms.



*Firefox's Migration Architecture*

## 3.2. Favicon Importing

Before FIIU, Firefox’s migrator supported the importing of bookmark and browser history data from the most popular browsers: Google Chrome, Internet Explorer, and Safari. On these browsers, websites usually have an icon (typically a small, square image) associated with them, known as a “favicon” (short for “favorite icon”). These favicons show up on tabs and next to bookmarked pages. However, when importing bookmarks from other browsers to Firefox, all favicons are lost, making web pages harder to recognize.

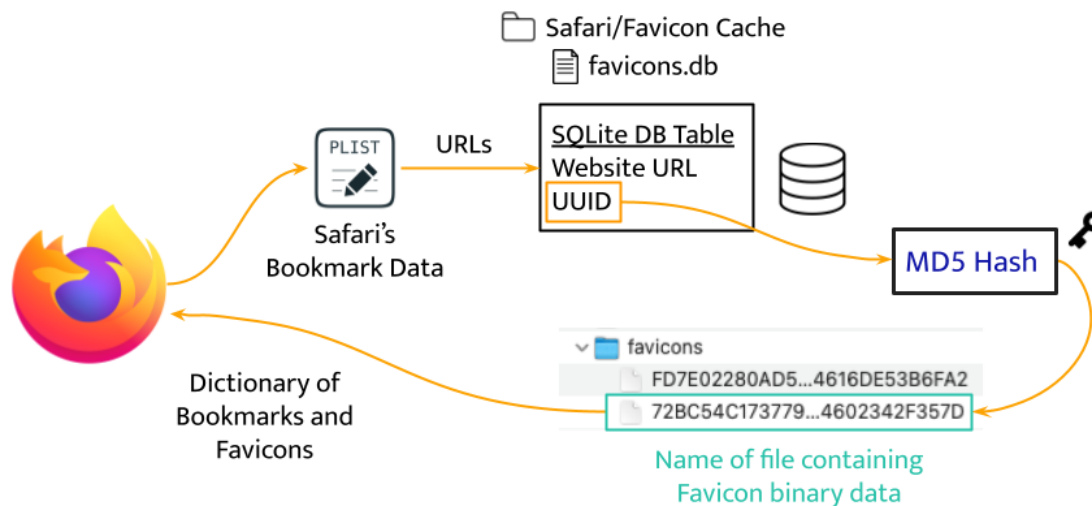


The FIIU team has added support for migrating favicons from Google Chrome, Safari, and Internet Explorer. Additionally, favicon imports are supported for Opera, OperaGX, and Vivaldi by extension, as they are Chromium-based browsers. New unit tests have also been added to ensure that favicon migration works properly.

Each major browser stored its users’ favicons slightly differently, but it typically involved files with key-value pairs, SQLite databases, and binary data for the favicon image. Just like bookmark data, these files are also stored on users’ local machines and can be read by Firefox. To reuse key logic, a team member wrote an *insertManyFavicons* function to populate Firefox’s own favicon database given a dictionary object mapping bookmark URLs to favicon binary data. This function was placed inside *MigrationUtils* and called by all browser-specific migrators.

As an example, Safari’s favicon data was stored separately in “~/Safari/Favicon Cache.” Each favicon can be found as ICO (an image file format for computer icons) files inside the folder. However, there is a rather complex mapping from these files to bookmark URLs. Safari maintains a “favicons.sqlite” file containing a table with rows corresponding to bookmarks and their favicons. Each row has a bookmark URL column, along with a Universal Unique Identifier (UUID) for it. After hashing this UUID with the Message Digest Method 5 (MD5) algorithm, the resulting string gives the name of the ICO file inside the favicons folder. With this, the corresponding ICO file’s binary data can be read and appropriately matched up with its bookmark URL. Lastly, calling the new *insertManyFavicons* function inserts all this data into Firefox.





*Safari's favicon import process*

The general idea for obtaining favicon data from the other two browsers looks like the following:

- Google Chrome
  - Create a set to store all bookmark URLs
  - Join multiple SQLite tables containing favicon IDs and URLs to obtain favicon binary data from the database files
  - Using the SQLite table data and the bookmark URL set, create an object that maps bookmark URLs to their favicon data
- Internet Explorer
  - Iterate through a folder containing files with the .url extension, where each file contains the information of a bookmarked website
  - For each file, read its metadata to get the favicon corresponding with the bookmark
  - This is done in the same loop as reading the bookmark data, so simply match the bookmarked URLs and favicons based on their file names

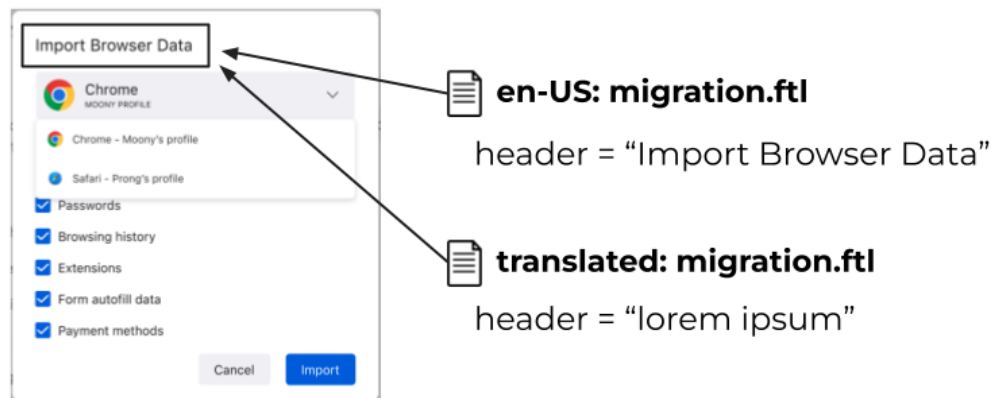
### 3.3. User Interface Design

Experts at Mozilla created new user interface prototypes on Figma for the FIIU team to implement. Team members developed new modal windows with HTML and CSS following these designs. These windows form the basis of the Firefox migrator's new user interface, and they contain fewer steps for migration than before. Their themes are consistent with the browser's overall color design and will look the same across different operating systems. It dramatically improves the migrator's aesthetics compared to the old native OS popup windows, which will no

longer be used. Mozilla's engineers designed and implemented the new migrator's general code structure. As a result, FIIU mainly focused on the individual migration window components.

Because Firefox was available in around 100 languages, the main challenge was considering English words written in new UI elements. Mozilla had a dedicated team reviewing FIIU's code to ensure that sentences written in English could be easily localized (localization is the process of translating a product's language and adapting its characteristics for a specific country or region). Once the new UI elements were properly designed for en-US (English – United States), localization teams would translate them with online volunteers worldwide. From a software engineering perspective, several design strategies were necessary to simplify the process for translators.

Although UI elements were created with HTML, no actual text displayed to users was in the code. Instead, the text was stored inside FreeMarker Template (FTL) files, commonly used with the Model-View-Controller (MVC) design pattern. By separating the English text from actual code, designers and translators have a much easier time with localization. The FTL files, which consist of uniquely identified lines of English text, are placed inside the “locales/en-US/browser” folder. Note that this is separate from where the migration code is written. With this design, translators could copy the English FTL file into the folder corresponding to their language and translate each line of text. Then, by updating language settings in Firefox, the content from a certain language's FTL files would be injected successfully into the browser's UI. Therefore, no additional code changes are needed from translators for everything to work.

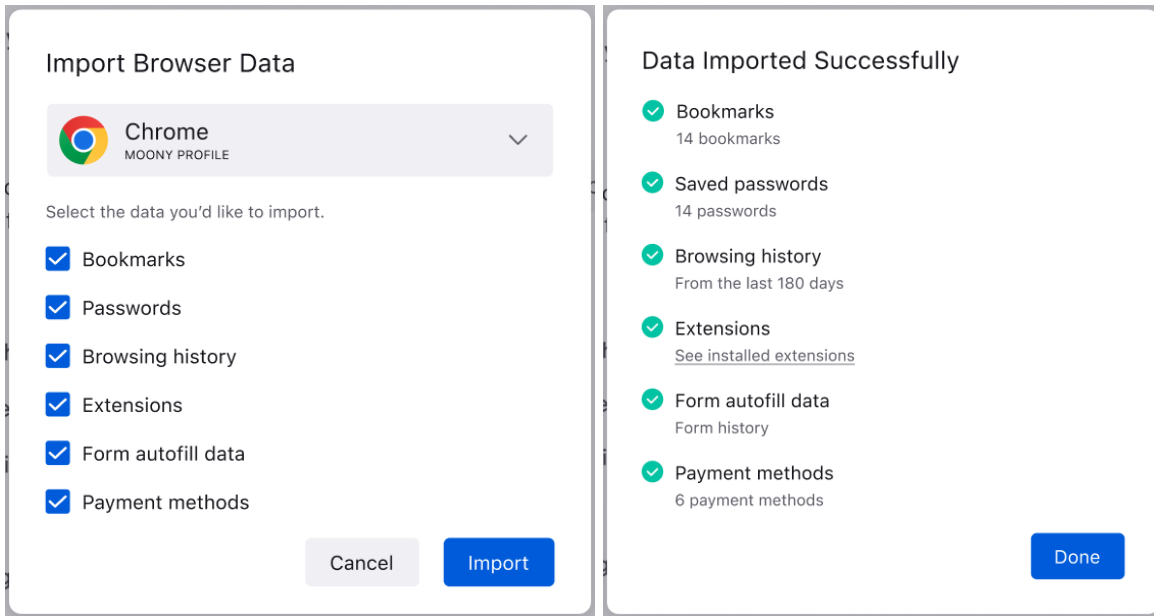


*How FTL files provide UI text*























## 4. Conclusions



### 4.1. Results

FIIU offers a handful of improvements, from new UI designs to more data support.



*New Migration Process (2 steps)*

Browser Migrator	 Safari	 Chrome	 Chromium Snap	 Opera	 OperaGX	 Internet Explorer	 Vivaldi
Import Bookmarks and History							
Import Favicons							
Import Credit Cards							

 : Existing     : Added

*Summary of FIIU's Work*

## 4.1.1 New Data Imports

### 4.1.1.1 Expanded Browser Support

The improved migration tool has extended data migration support to include Opera, OperaGX, and Vivaldi, all of which are Chromium-based browsers. This expanded browser support broadens Firefox's appeal to a larger user base.

### 4.1.1.2 Favicons

The project has added support for migrating favicons from Google Chrome, Safari, and Internet Explorer, detailed in Section 3.2.

### 4.1.1.3 Credit Card Data and Payment Methods

The team successfully implemented a process to import credit card data and payment methods from Google Chrome. It involved decrypting the SQLite database stored locally and inserting the data into Firefox's database. This achievement opens the pathway for new AutoForm import options, such as addresses, and expands the range of data that can be migrated to Firefox.

## 4.1.2 User Interface Results

### 4.1.2.1 Modal Windows and Localization

The FIU team developed seven new modal windows, designed with translation for 100+ languages in mind. This allows Firefox to cater to a global user base.

### 4.1.2.2 Streamlined User Experience

The team improved the migration wizard's user experience by reducing the number of steps, updating the theme, and filtering out disabled migrators.

### 4.1.2.3 Testing

The team modified and added unit and integration tests to ensure the stability and reliability of the new features.

### 4.1.2.4 Permission Dialog and Error Handling

The team developed a permission dialog page tailored explicitly for Safari users, requesting permission to access data during the migration process. Additionally, the team implemented an error message display to notify users when no importable resources are detected, ensuring a smooth and transparent user experience during the migration process.

### 4.1.3 Other Project Achievements

#### 4.1.3.1 ESMification

FIIU has successfully performed ESMification on over 25 JavaScript modules. This process converts files from an old, Mozilla-specific ESMification, which have been integrated into the public release version of Firefox. It marked an essential step in the ongoing transition from JSM to ESM modules. However, it is essential to note that this process was independent of the expanded browser support and data migration improvements.

## 4.2 Impact

FIIU was a unique Senior Design Project for students, providing valuable software engineering experience. During the 2022-2023 school year, no other Senior Design Project worked on such a large-scale project that impacted so many people. Since FIIU's contributions to Firefox's migrator will be publicly released in Firefox, the project will benefit over 200 million existing users worldwide. In addition, all new users will be asked whether they would like to import their data upon downloading Firefox. This would result in even more opportunities to interact with FIIU's work.

### 4.2.1 Enhanced User Experience

The FIIU project introduced significant improvements to Firefox's migrator. The new user interface, which aligns more consistently with the Firefox theme, provides an intuitive and uniform migration experience across all operating systems. Furthermore, users of previously unsupported browsers can now seamlessly import their data, including previously lost data like favicons. Several online articles have even been written in anticipation of Firefox's new features due to the project [4, 5].

### 4.2.2 Expanded Browser Support

The enhancements to the migration tool have expanded Firefox's compatibility with a wider variety of browsers and data types. This expanded support enhances Firefox's appeal to a more diverse user base, offering a seamless and flexible transition to Firefox from any platform.

### 4.2.3 Favicon Imports

The introduction of favicon support plays a crucial role in users' recognition of their bookmarks. Favicons provide visual continuity, helping users quickly locate and recognize their bookmarked and frequently visited sites. This enhancement supports users in maintaining their browsing habits and preferences, contributing to Firefox's overall usability.

#### 4.2.4 Codebase Modernization

FIIU's contributions have modernized part of Firefox's codebase through ESMification. The new modules follow current JavaScript standards, allowing for better long-term maintainability of the codebase.

### 4.3 Future Work

As FIIU's code is now part of Firefox, others will inevitably continue building on top of it. Currently, FIIU has only worked on a subset of migration features for the major browsers at the time of writing; much more work can be done. Furthermore, the project's contributions continue to open the door for importing even more data. For example, by allowing Google Chrome's credit card data to be migrated, related data stored in similar formats may soon be supported.

Some suggestions for future work to build upon FIIU are as follows:

1. Complete ESMification of Firefox's codebase

As the FIIU team learned about Firefox's software development process, students performed ESMification on over 25 modules. As of May 2023, over 75% of Mozilla's 1400+ JSM files have been converted into ESM. Finishing this process would be beneficial to keep Firefox's codebase modern.

2. Continue exploring data that can be imported

As FIIU searched for more browser data to import, it discovered additional possibilities for what could be migrated. For instance, autocomplete data on forms can be imported to Firefox.

3. Enhance user interface based on feedback

The new migration windows have been designed with two different variations in mind. To decide which is better, Mozilla engineers plan to test the efficacy based on user data and feedback. Afterward, further user interface enhancements may be necessary since the migration windows are entirely redesigned.

## 5. References

[1] Cal State LA, “Capstone Senior Design Program,” calstatela.edu.

<https://www.calstatela.edu/ecst/seniordesign>

[2] Mozilla, “History of the Mozilla Project,” mozilla.org.

<https://www.mozilla.org/en-US/about/history/>

[3] Mozilla, “Firefox FAQ,” mozilla.org. <https://www.mozilla.org/en-US/firefox/faq/>

[4] Nestor, M. (2023a). Firefox 110 Will Finally Let You Import Data from Opera and Vivaldi.

9to5Linux.

<https://9to5linux.com/firefox-110-will-finally-let-you-import-data-from-opera-and-vivaldi>

i

[5] Nestor, M. (2023b). Firefox 112 Will Let Ubuntu Users Import Browser Data from the

Chromium Snap. 9to5Linux.

<https://9to5linux.com/firefox-112-will-let-ubuntu-users-import-browser-data-from-the-chromium-snap>

romium-snap