TabX

Khayyon Parker, Miguel Guerrero,
Mishig Davaadorj, Nicole Woch, & Russell Skorina

Overview

TabX is a writing assistant program that makes typing in a browser faster and easier. It is an accessibility tool that functions as a Google Chrome extension. The client is a public library which chooses to invest in the project so that the application can serve as an accessibility tool for its patrons. Given that the library is publically funded, its priority is not pulling in the greatest profit, but rather improving the overall experience of its patrons, many of whom are not very familiar with technology and are slow typers.

The people who often frequent the library either do not have regular access to a computer at home or are seeking information online, but are not sure where or how to access it -- potentially due to lack of experience or because they are not native English speakers. The library wants a product which will help to more efficiently fulfill such people's needs.

The library is also interested in minimizing the amount the amount of time librarians need to invest in assisting patrons with navigating web pages and finding resources, as some of the most frequently asked questions are related to these technological concerns. In addition, the library wants the finished product to be as close to universal as possible; it should be able to function on almost any web page, and especially the most popular ones frequented by its patrons.

Given the intent of the product to be an accessibility tool for patrons, user-friendliness is a priority for the library over speed and efficiency; so long as the application is functional, provides useful suggestions, and is efficient to the extent to which users do not experience noticeable delays in receiving suggestions when typing, seamless functionality of the user interface is most important.

Specification Traceability

The following traceability matrix demonstrates how the client's needs and the user stories are specifically fulfilled through the product features and technical architecture. The full traceability matrix can be found here.

Project Name	**TabX**		**QA Lead**	**Russell Skorina**
Project Workers	**Russell, Miguel, Khayyon, Nicole, Mishig**		**Target Implementation Date**	**Dec 18**
Client Need	**Product Feature**	**Use Stories**	**Architecture**	**Code File**
Help Speed Up User Tying Speed	Add Next Word Prediction	As a layman writer, I want to have suggestions for what the next word would be, so that I can quickly type common phrases.	A file that Does markov Chain prediction that is called by front end	word-prediction.js
	Add Finish Word Prediction	As a user who types slowly, I want to see word completions, so that I can type faster.	A file that finishes word prediction that is called by front end	wordCompletionModel.js
	Add Word Correction Prediction	As a person who makes frequent spelling mistakes, I want to be able to have spelling corrections suggested to me, so that I may type more accurately.	A backend file that returns 3 closest words to front end	wordCompletionModel.js
	Hotkey To automatically Select Next Word	As a user. I want to have hotkeys or mouse click to select the full, so I can quickly type	The Event Handler reesponsible for this is created in TabX. It currently uses only 1,2,3 to select the suggestions available. Modifications to the current event handler or creation of a new event handler would be the only thing needed to implement this feature.	tabx.js

Validation Testing

The client is a public library, so ideally, we will be testing this by asking library patrons to test our application in exchange for their feedback. After receiving the user feedback, we can determine the parts of the system that contributed heavily to some of the negative feedback. We will refactor the code to conform to the client's requirements as needed. We will collect both quantitative and qualitative feedback which will help us assess the accessibility and discoverability of TabX.

The word completion feature uses the partial string in the input field to quickly fill out the words using a keyboard shortcut. This will allow slow typers to quickly complete words to common sentences resulting increased typed speed of the user, expediting the writing process. Since we expect an increase in typing speed, we can use two computers with the same hardware specifications: One with the extension and another without the extension. Then we can use a script to close the browser after 100 words are typed and calculate and compare the typing speed to determine if the difference is non-trivial. This test allows our client to determine whether TabX is helping their patrons write faster, as specified.

The client want to be more inclusive of the writing services provided to another set of patrons: non-native English-speakers. The word prediction feature can provide common English words to the user, which should allow the patrons to form common phrases with relative ease. We can validate the relative ease by using a script that count the number of word prediction calls in a set time frame. This test is significant because it allows us to verify the relative ease through the prediction calls/hour rate to determine how frequently our services are being used. This helps our client validate whether TabX is assisting their non-fluent English speakers in their writing.

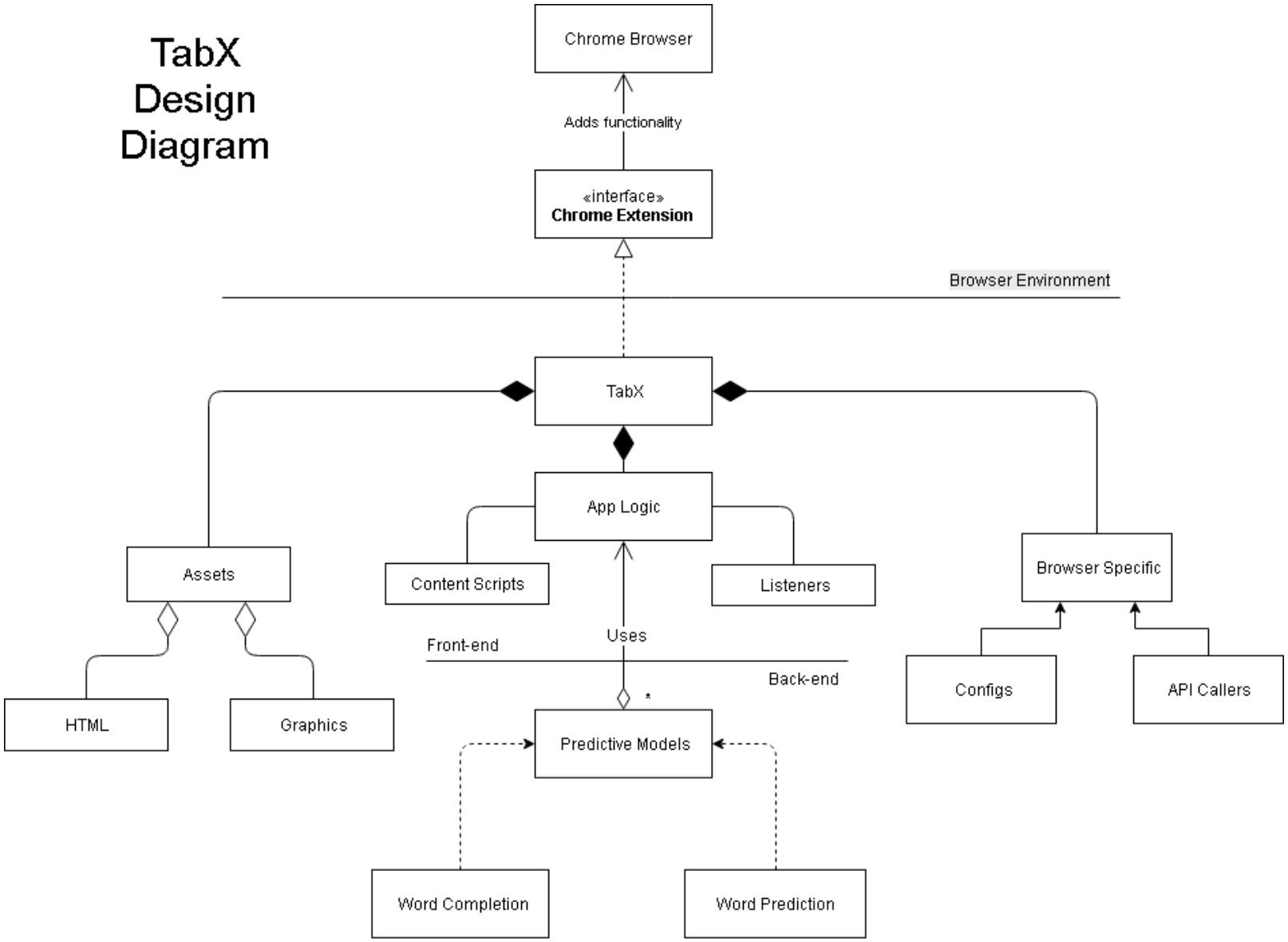
The client would like to provide accessibility to the writing services provided for their less-abled patrons. Specifically, the client is more concerned with patrons that are color blind or visually impaired. TabX will provide font size customization and color schemes that most people can see. We can validate the accessibility features by taking the user asking if they can see the suggestions within the application. For a less obtrusive experience, we also plan to observe for key behaviors that would suggest a user is struggling to interact with TabX. For example, if a visually impaired user leaned in closer to their monitor or if a color-blind user could not identify the colors in the UI. This test helps our client validate whether TabX is serviceable for their less-abled patrons.

Design Documentation

There are two major considerations that went into TabX's design:

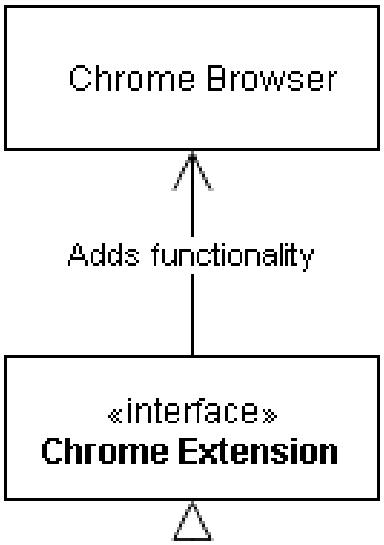
1. Meeting the client's requirements in providing several writing assistant features
2. Abiding by Chrome's extension interfaces and API

As a result, TabX is being designed such that it allows developers to add new features such was word completion and word prediction without too much modification to the code and allows TabX to operate within a browser environment. This was done by separating the concerns of the application into small subsystems that are responsible for one task.



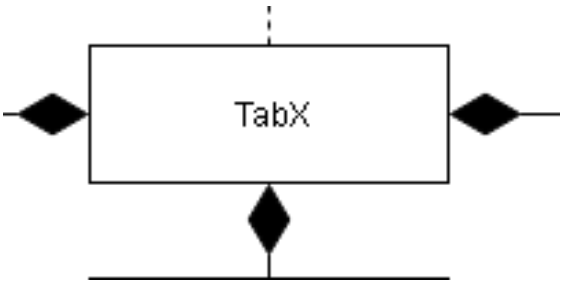
Subsystems

Chrome Extension Interface



By abiding by this interface, TabX "contractually" agreed to follow Chrome's specifications. This puts technical constraints on the rest of TabX's subsystem. This mainly includes that code must run in a browser environment and follows standard security protocols (e.g. CORS).

TabX

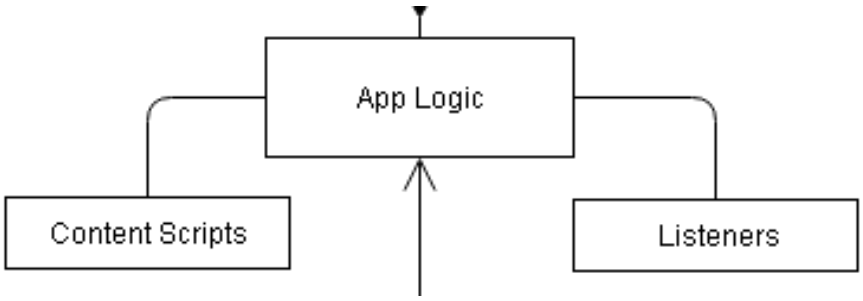


TabX is the point-of-contact to Chrome. It is composed of three main subsystems:

1. The application logic
2. Assets and static resources
3. Browser specs

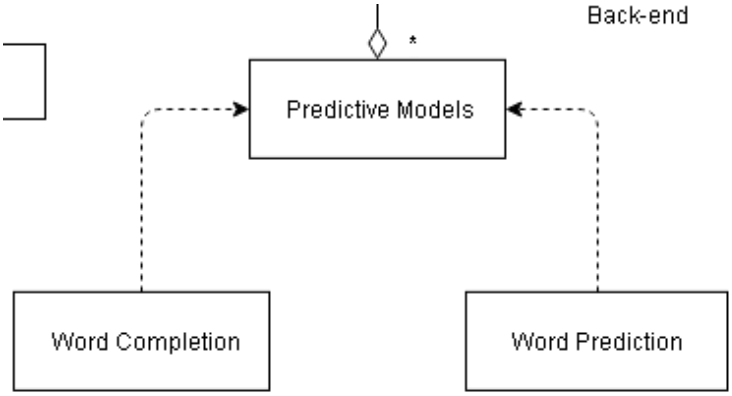
TabX provides several interfaces by which these subsystems can abide. These interfaces ensure that TabX follows Chrome's specifications, as well as abstract away implementation details for flexible system design.

Application Logic



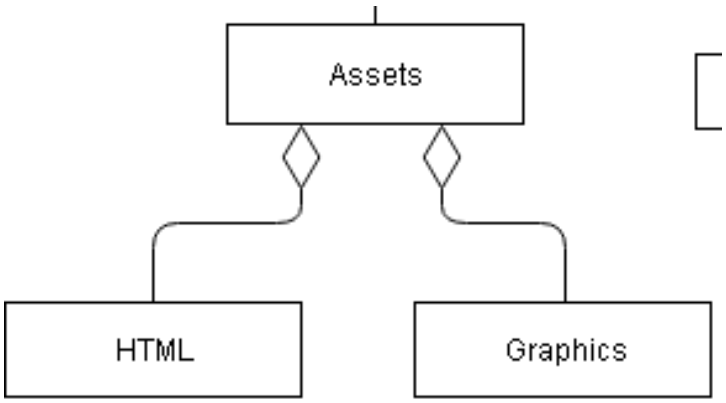
This layer of the TabX system is responsible for scripts that can run webpages and allows users to access writing assisting services.

Application Logic: Predictive Models



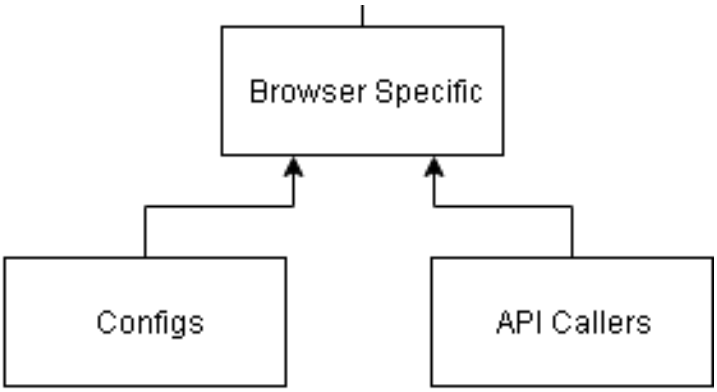
The application logic includes a backend component that is responsible for processing user input and providing suggestions on what words the user might find helpful. The word completion/correction model makes suggestions by traversing a trie and retrieving the words nearest to the input word with respect to matching characters. The word prediction model makes suggestions by using a Markov chain algorithm and suggesting the top three words with the highest probabilities relative to the input word. These models contribute to meeting the client's expectation of a writing assistant that is both fast and accurate in its predictions.

Assets



The assets subsystem contains resources that are used in displaying information to the user and separating the presentation aspect of TabX from the application logic subsystem.

Browser Spec Subsystem



Finally, this subsystem is responsible for configuring TabX for use as a Chrome extension and abstracting away how TabX interacts with browser APIs such as persisting data across browser sessions.

Progress Summary

Feature	Description	Expected Date of Completion	Assigned
Word completion	Trie	Complete	Russell
Word correction	Trie - if word is misspelled	Complete	Russell
Word prediction	Modified Markov chain	Complete	Nicole & Mishig
Customizable word/text training sets	For word prediction - results in interchangeable trained backend models	Complete	Nicole
Register user input from text fields	Grab user input from text fields to be passed to backend prediction models	Complete	Khayyon & Miguel
Hotkey to select next word	Press 1 to complete with first suggestion, 2 for second suggestions, and 3 for third suggestion	Complete	Khayyon & Miguel
Chrome extension	Function in the browser as a Chrome extension	Complete	Khayyon & Miguel
Consistent display of suggestion table	Vertical suggestion table already displays, but needs to be made to persist across all webpages constantly	Saturday 12/8	Khayyon
Abbreviation expansion	Backend model to expand common abbreviations	Friday 12/14	Nicole
Profanity filter	Do not provide suggestions for profanity	Tuesday 12/11	Nicole
Ignore form/sensitive info text fields	Do not register input from/provide suggestions for password fields or other fields that may contain sensitive information, such as in forms	Tuesday 12/11	Mishig
Backend validation tests	Unit tests to make sure backend models are functioning as expected	Tuesday 12/11	Russell
User testing metrics	Metrics to quantify feedback from user testing	Saturday 12/8	Russell
Settings tab display	Ability to toggle on/off writing assistant features and services in the display, but not functional with the application yet	Complete	Khayyon & Miguel
Integrate settings tab with TabX functionality	Settings display tab already exists, but needs to be integrated with the corresponding functioning features in the application	Saturday 12/8	Miguel
Setting: toggle between horizontal suggestion display bar and vertical display table	Implement horizontal suggestion display bar as well as setting to allow users to choose a suggestion display method	Tuesday 12/11	Miguel

Setting: configure display options such as font size, color, etc.	Setting to configure user displays options to improve readability, accessibility, and overall user experience	Tuesday 12/11	Khayyon
Setting: toggle between models/word sets used for training	Setting to switch between different types of the same prediction model so that the training word set used by the application better fits the needs of library patrons with respect to usefulness of word suggestions	Tuesday 12/11	Mishig