

Software Requirements Specification (SRS)

Project X-Browser

Team: Team 6

Authors: Alkevicius Aurimas, Ho Stephen, Hunter Stevens, Tran Phuc, Wang Xuhao

Customer: Front-end Web Developers

Instructor: Professor Daly

Table of Contents

1 Introduction	3
1.1 Purpose	3
1.2 Scope	3
1.3 Definitions, acronyms, and abbreviations	3
1.4 Organization	4
2 Overall Description	5
2.1 Product Perspective	5
2.2 Product Functions	6
2.3 User Characteristics	6
2.4 Constraints	7
2.5 Assumptions and Dependencies	8
2.6 Apportioning of Requirements	9
3 Specific Requirements	10
4 Modeling Requirements	11
5 Prototype	12
5.1 How to Run Prototype	12
5.2 Sample Scenarios	13
6 References	13
7 Point of Contact	13

1 Introduction

This document will establish a framework for the proposed software product, X-Browser, a web development tool that assists the user by displaying code simultaneously in multiple web browsers in real-time. This solution increases web design efficiency since it eliminates the need for switching between browsers to test the code.

The first section of this document contains brief descriptions of the purpose, scope, and goals of the X-Browser system. It will also outline the high-level specifications of the system's perspective, functionality, and constraints. The specific requirements of a successful solution will follow next, as well as a series of various diagrams. A description of the prototype follows as well as sample scenarios of using the prototype. Finally, the document will conclude with a list of references and points of contact.

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to fully document the specifications and requirements for the cross-web browser package in an open-source text editor, Atom. The intended audience of this SRS is the developers of this software and users who would like to know more about this product.

1.2 Scope

The scope of this project is to create X-Browser as a package for an open-source text editor Atom. The benefit of this package includes increased efficiency of the web page design process. X-Browser will display the real-time code edits in several different web browsers at the same time. In turn, it will allow web developers to immediately inspect any compatibility issues across various web browsers. Compatibility issues occur as a result of inconsistent support of web programming languages across various web browsers.

X-Browser package will be available to install using Atom packages Installer. The web designers will have an option to launch various locally supported web browsers. The code typed in Atom will be immediately displayed in launched browsers.

1.3 Definitions, acronyms, and abbreviations

Asynchronous: using asynchronous requests processing allows the browser continue to work as normal while request is being handled.

Atom: open-source text editor.

Front-end: front-end web development is the practice of converting data to a graphical interface.

Hotkey: an assigned key or sequence of keys programmed to execute a command or perform a specific task in a software application.

HTML: Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser.

Open-source: is a source code that is made freely available for possible modification and redistribution.

Package: a software collection that provides certain functionality as part of a larger system.

Plugin: is a software component that adds a specific feature to an existing computer program.

Real-time: Terminology that will be used to describe the feature that allows users to see changes on the browsers as it is modified

Software application: a type of computer program that performs a specific function.

Split panel: the screen is split vertically in half. The left side displays the source code (working file) in Atom. The right side is split horizontally equally by the selected browsers.

Web programming language: a formal language, e.g., HTML comprising a set of instructions involved in Web development.

Web browser: a software application, e.g., Chrome, Firefox, Safari, MS Edge, and IE used to access information on the World Wide Web.

URL: A Uniform Resource Locator also known as a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

1.4 Organization

The rest of the SRS document is organized as follows:

- **Section 2** is an overall description of the project. This includes all properties, functions, constraints, assumptions, dependencies, and requirements of the product.
- **Section 3** cites the specific requirements for the X-Browser package.
- **Section 4** provides models and diagrams, e.g., case, class, and sequence, of the product.
- **Section 5** describes a prototype of the project. It outlines functionality using various example scenarios. As well as the instructions to download and launch the prototype.
- **Section 6** lists all references used in the creation of the product.

- **Section 7** lists points of contact.

2 Overall Description

This section will address information on how to use X-Browser. This information will also include the context in which X-Browser is created and used. It will also explain the interface of X-Browser and its functions. This section will also cover the minimum requirements to use the product and the constraints of X-Browser.

2.1 Product Perspective

While 70% of the world's desktop internet browser users use Chrome, the other 30% of browser users are split between Safari, Firefox, Edge, Opera, and Microsoft Internet Explorer [2]. This makes it necessary for web developers to support and test features on each of these browsers, which can slow down development and testing.

X-Browser is an extension for the Atom text editor. It aims to give front-end web developers the ability to work more efficiently by avoiding switching between browsers when testing website features. Instead of having to test each feature on each browser, X-Browser displays and updates a website on multiple browsers simultaneously. After opening a HTML file in Atom, a developer can open the X-Browser menu in Atom and select which browsers to launch. The launched browsers will display the specified file or URL, and update automatically whenever the site is updated or on a timer. The browser windows will be able to be stacked or tiled via drag-and-drop or shortcut keys. With this method front-end developers will save time testing and visualizing source code behavior on the different browsers.

An internet connection will not be required since a local web server is provided. X-Browser will only support major browsers such as Chrome, Firefox, Safari, Opera, Microsoft Edge, and Microsoft Internet Explorer.

2.2 Product Functions

The major functions of X-Browser are listed below:

- Add the browsers available on the local machine to the menu of the Select browser.
 - Search local machine for available browsers
 - Save file paths of web browser executables to [] directory
 - Populate available web browsers to the menu
- Allow users to select browsers they want to work on.
- Implement a split panel interface that allows users to visualize their website on different browsers simultaneously.

- Refresh browsers every 3 seconds, for real-time editing and testing.

The diagram below shows the developing goal of X-Browsers. The rectangle shape represents the ultimate goal that needs to be achieved by X-Browser. The oval shape represents each task required to accomplish to fulfill the ultimate goal.

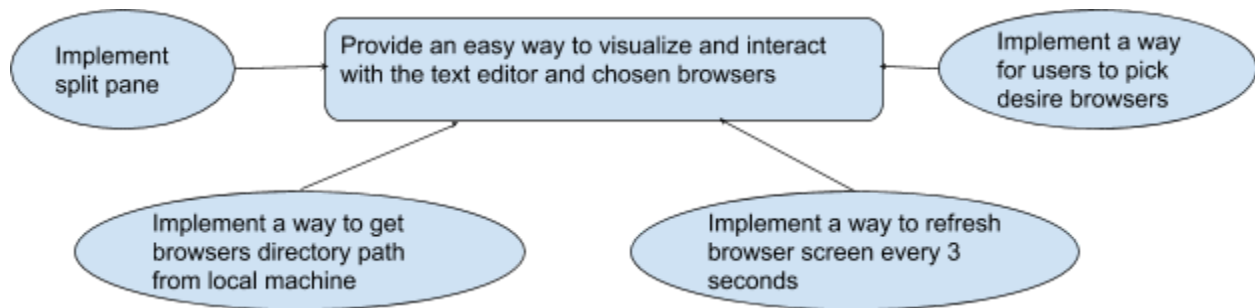


Figure 2.2

2.3 User Characteristics

The users for X-Browser are expected to be front-end web developers who need to test their website on multiple browsers. The user is expected to have working knowledge of the Atom interface, including installing Atom packages.

2.4 Constraints

Some prior user experience with Atom is necessary to install the X-Browser package. The user might also have to set file paths to the browser executables manually in case it is not done so automatically as intended during installation.

The X-Browser Atom plugin will be developed with JavaScript. Since it is an Atom plugin, it requires a system to be able to install and run Atom. Atom can be run on Windows, Linux or MacOS operating systems.

2.5 Assumptions and Dependencies

As Atom plugins are written in JavaScript, they can be installed and used on any system that supports the Atom editor. Atom runs on OS X 10.10 or later, Windows 7 or later, RedHat Linux, and Ubuntu Linux [3].

It's assumed that users will have prior experience with Atom, so a tutorial will not be implemented. Only a brief explanation will be available on Github or Atom Install package page.

2.6 Apportioning of Requirements

This version of X-Browser is an Atom plugin, as the main features of X-Browser work well as an addition to an existing text editor. If needed, X-Browser could be remade as a standalone text editor. It would currently not be a good use of resources to create our own editor, as X-Browser cannot be used to test websites as they would run on smartphones and tablets. A X-Browser smartphone application could be developed to enable real-time testing on smartphones or smartphone emulators.

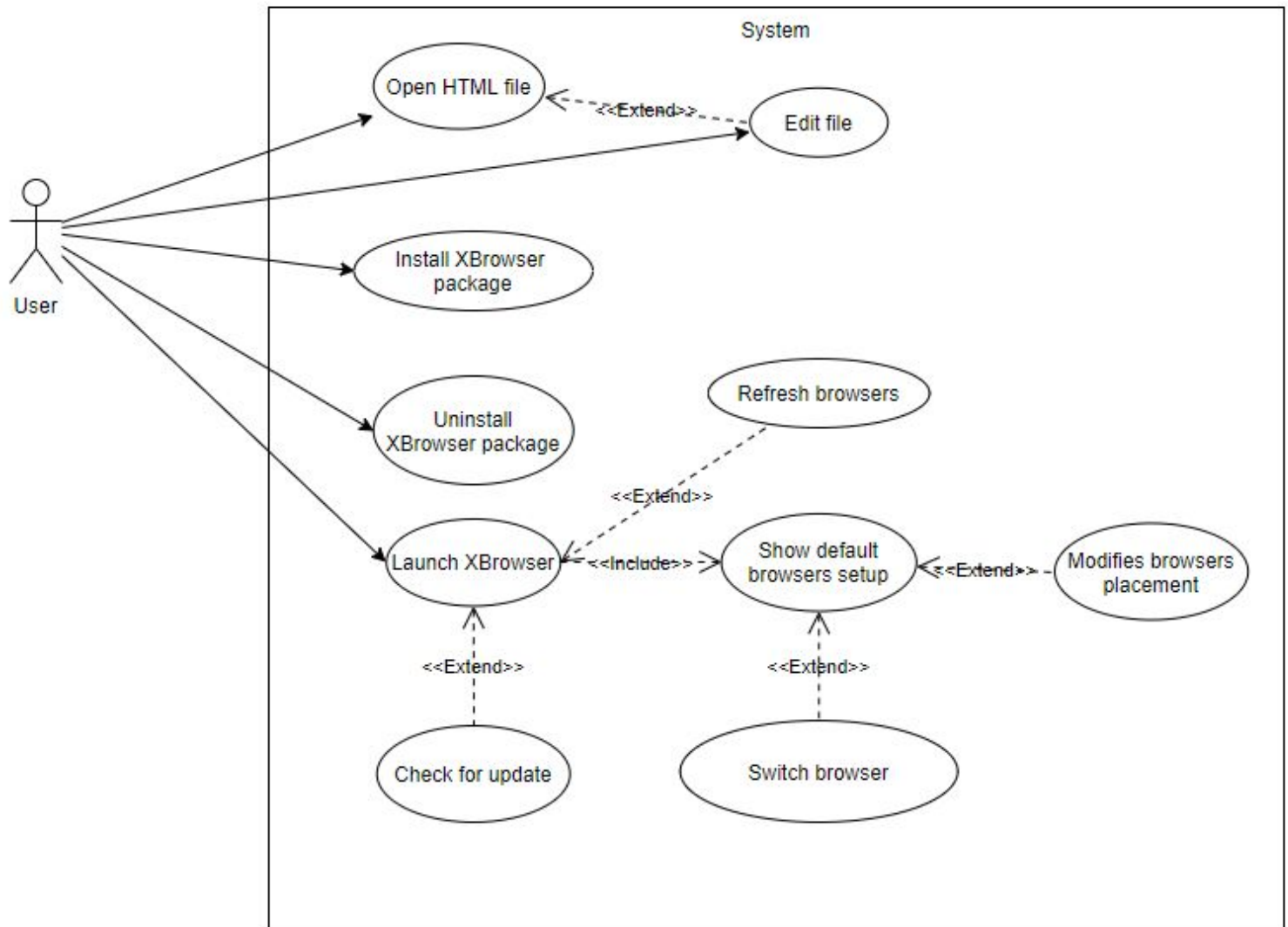
3 Specific Requirements

1. X-Browser Package should be compatible with Linux, Windows, and macOS versions of Atom
2. X-Browser will support major web browsers: Firefox, Google Chrome, Safari, MS Edge, Internet
3. After selecting the desired browsers, X-Browser will setup the layout for Atom text editor and selected browsers
 - a. On default layout, the display screen will split vertically in half
 - i. Atom text editor occupies the left half of the display
 - ii. Selected Web browsers occupy the right half of the display
 1. Divided horizontally equally between selected web browsers
 - b. On custom layout, the display screen will be set the same as the previous layout.
4. When the user opens the HTML file, X-Browser will get the directory of the file from Atom text editor to display the content of the source code.
5. When the user edits the source code, any syntax checking and parsing of HTML files will be handled by the Atom text editor.
6. User can navigate between web browser windows
 - a. Switching between web browser windows by using hot-keys
7. Web browsers will update at real-time by refreshing the web page or tab every 3 seconds
8. User will be able to remove X-Browser package from Atom
9. X-Browser should check for updates every time the plug-in is open
 - a. Users will be able to choose to update automatically or manually at the first time launching X-Browser.
10. X-Browser should be compatible with Atom updates
11. Web browser Window Management
 - a. Should be able to relocate web browser window anywhere on the right side of the display
 - b. Should be able to resize web browser window within the right side of the display
12. Atom must be launched in order to launch X-Browser
13. X-Browser must close when Atom text editor is closed

4 Modeling Requirements

Use Case Diagram:

This diagram comprises of multiple use cases and their relations towards each other



Use Case Name:	Open HTML file
Actors:	Users

Description:	User opens HTML file on Atom
Type:	Essential
Includes:	N/A
Extends:	Edit file
Cross-refs:	Requirement 4
Uses cases:	N/A

Use Case Name:	Install X-Browser package
Actors:	Users
Description:	Installs X-Browser package from atom application
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A
Uses cases:	N/A

Use Case Name:	Uninstall X-Browser package
Actors:	Users
Description:	Remove X-Browser package from Atom
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 8
Uses cases:	N/A

Use Case Name:	Launch X-Browser
Actors:	Users
Description:	Browser path is added to config file
Type:	Primary
Includes:	Show default browsers setup
Extends:	N/A
Cross-refs:	Requirement 12
Uses cases:	N/A

Use Case Name:	Check for update
Actors:	N/A
Description:	Find the latest version of X-Browser to ensure X-Browser is up to date
Type:	Primary and Essential
Includes:	N/A
Extends:	Launch X-Browser
Cross-refs:	Requirement 9
Uses cases:	N/A

Use Case Name:	Edit file
Actors:	User
Description:	Allows any opened html file to be edited
Type:	Primary and Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 5
Uses cases:	N/A

Use Case Name:	Refresh browsers
Actors:	N/A
Description:	Give user a real-time testing experience by refreshing browser every 3 seconds
Type:	Essential
Includes:	N/A
Extends:	Launch X-Browser
Cross-refs:	Requirement 7
Uses cases:	N/A

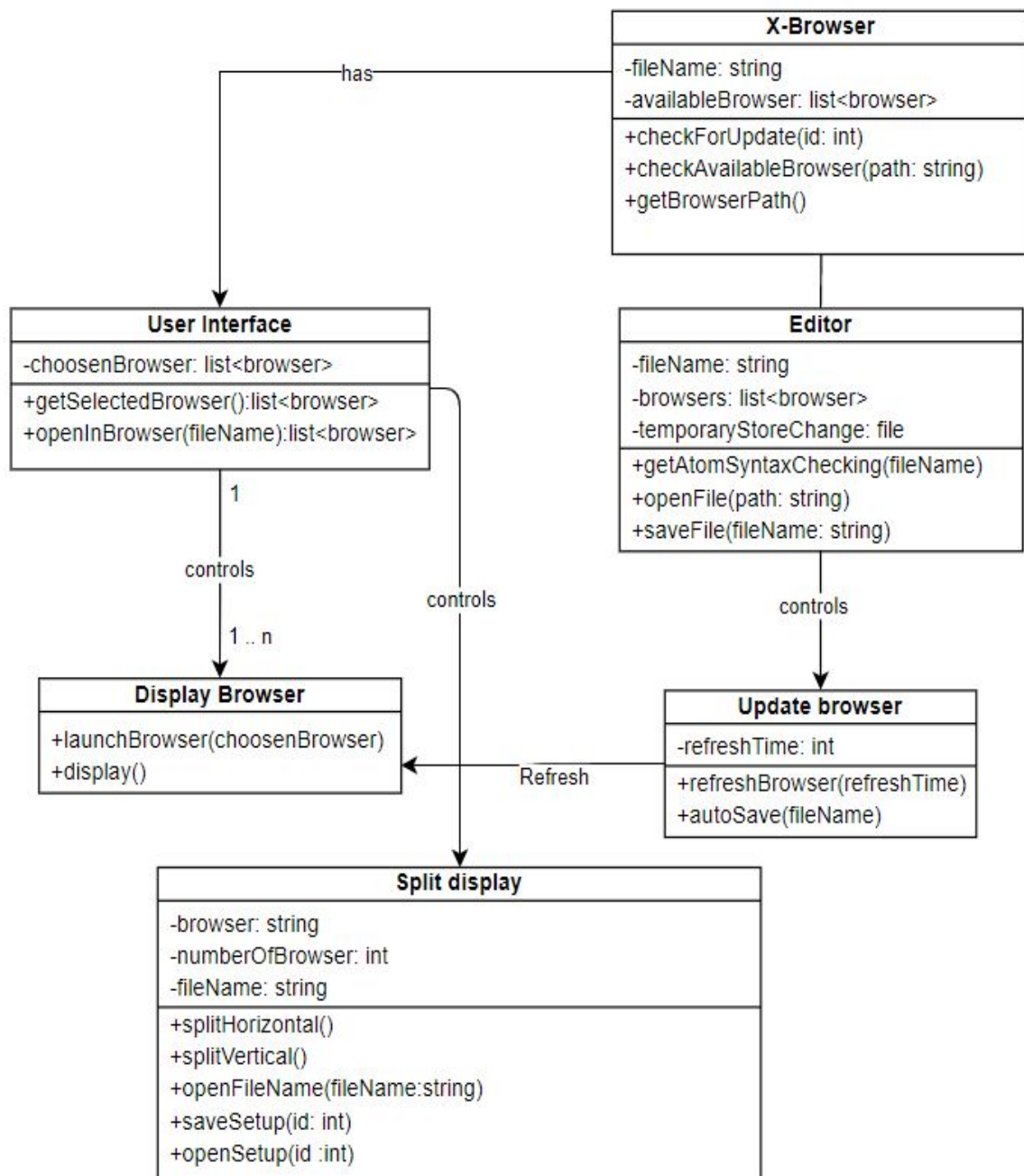
Use Case Name:	Show default browsers setup
Actors:	N/A
Description:	Browser setup is set to default when X-Browser is open
Type:	Primary and Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 3.a
Uses cases:	N/A

Use Case Name:	Modifies browsers placement
Actors:	N/A
Description:	Allows configuration of browser placement when multiple browsers are open
Type:	Essential
Includes:	N/A
Extends:	Show default browsers setup
Cross-refs:	Requirements 11.a and 11.b
Uses cases:	N/A

Use Case Name:	Switch browsers
Actors:	N/A
Description:	Opened browsers can be switched with configured hotkeys
Type:	Primary
Includes:	N/A
Extends:	Show default browsers setup
Cross-refs:	Requirement 6
Uses cases:	N/A

Class Diagrams:

The following diagram represents the main classes that are essential to the use of the X-browser software.



Below is a brief description of each class and their functions.

Class Name	Description	
X-Browser	The X-Browser class will be used in order to handle all the maintenance functions of the plugin. It will handle updating, and setting up the correct files for the use of the user.	
Attributes:	fileName:String	a string variable that will be used in order to save the name of the file used in Xbrowser.
	availableBrowser:list <browser>	list that will keep track of the available browsers that the user may use in order to display their file.
Operations:	checkForUpdate()	a function that will automatically check to see if there is an update available for the X-browser plugin.
	checkAvailableBrowser()	a function that will return whether or not a specific browser is available on the local machine by checking the default directory.
	getBrowserPath()	a function that will return the path to which the browser is stored on the local system.
Relationships:	The X-Browser class will interact with the User interface class and the editor class. The User Interface class will access the checkAvailableBrowser function of the X-Browser class and use that to open only valid browsers.	
UML Extensions:	None.	

Class Name	Description	
User Interface	The User Interface class will be used in order to set up all interactions done by the user. It will take input from the user and display the desired output.	
Attributes:	chosenBrowser: list<browser>	will be an array of all the browsers that the user has selected to display their code.
Operations:	getSelectedBrowser(): list<browser>	a function that will retrieve all information needed in order to open the chosenBrowser.
	openInBrowser(fileName)	is a function that will take a file name as a parameter, and open the desired file in the browsers specified by chosenBrowser.
Relationships:	The User Interface class will use functions from the X-Browser class in order to verify which browsers may be opened, as well as from displayBrowser in order to properly display which browsers are available. It will also interact with the Split Display class in order to assist with the proper display the user desires.	
UML Extensions:	None	

Class Name	Description	
Display Browser	Display Browser will be a simple class that will be used for the sole purpose of launching the selected browsers designated by the user.	
Attributes:	launchBrowser()	is a function that will take the chosenBrowser, and launch the browser using the function getBrowserPath().
	display()	Display selected browser inside Atom application
Relationships:	The display browser class will be associated with the user interface class in order to properly display which browsers the user wishes. It will also be used by the update browser class in order to properly display the correct, updated data from the file that is desired.	

UML Extensions:	None
-----------------	------

Class Name	Description	
Split display	Split Display is a class that will be used in order to handle all of the display functions of the plugin. It will handle the modifications of the orientation of the plugin in order to save the preferences of the user.	
Attributes:	browser: string	Browser will be a string that will be used in order to modify the display of the browsers in use.
	numberOfBrowser:int	An integer variable that will be used in order to keep track of the number of browsers the user wishes to use.
	fileName: string	A string variable that will be used to keep track of the name of the file that the user is currently working on.
Operations:	splitHorizontal()	will be a function that will split the current browsers in a horizontal manner.
	splitVertical()	will be a function that will split the current array of browsers in a vertical manner.
	openFileName(fileName: string)	A function that will take a fileName as a parameter, and use this in order to open the desired file designated by the user.
	saveSetup(id: int)	A function that is used in order to store the current display setup, in order for the user to use in the future.
	openSetup(id: int)	A function which restores the previous window position setup when X-Browser is started.

Relationships:	The split browser will be used in relation with the user interface class in order to properly display the proper interface that the user wishes to use.
UML Extensions:	None

Class Name	Description	
Update Browser	Update Browser is a class that will handle all the updating between the browsers and filenames.	
Attributes:	refreshTime: int	Given time that will refresh the browser automatically
Operations:	refreshBrowser(refreshTime)	A function that will update the browser based on the refreshTime in integers.
	autosave(filename)	A function that will update any changes made to the selected filename.
Relationships:	The Update Browser class will be used in relation with the display browser class in order to display the most recently updated file in the selected browsers.	
UML Extensions:	None.	

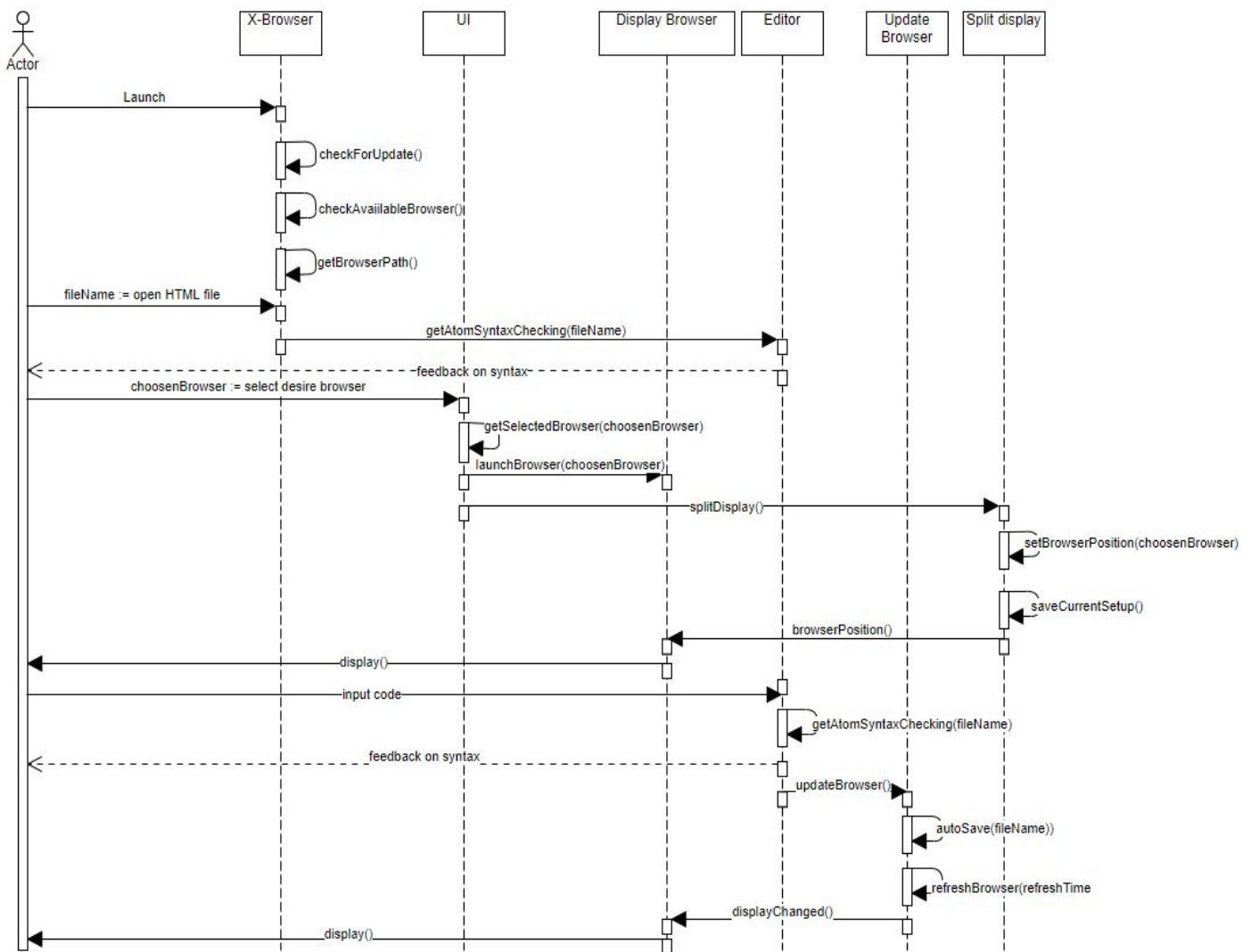
Class Name	Description	
Editor	Editor will be a simple class that will be used in order to ensure that the syntax in the file is correct.	
Attribute	fileName: string	Store the name of the working file
	browsers: list<browser>	A list which contains the names of browsers to be opened.
	temporaryStoreChange: file	Store the contents in the temporary file without saving the changes

Operations:	getAtomSyntaxChecking(fileName)	A function that will take a fileName as a parameter, and scan the file in order to use atom to check the syntax in the file.
	openFile(path: string)	The function will open the file in the given list of browsers
	saveFile(fileName: string)	The function will save the file given the name of the file.
Relationships:	The Editor class will borrow attributes from the X-Browser class to acquire the filename into the atom editor. This class will be used for the Update Browser class in order to keep all the data up to date onto the editor.	
UML Extensions:	None	

Representative Scenarios of the System:

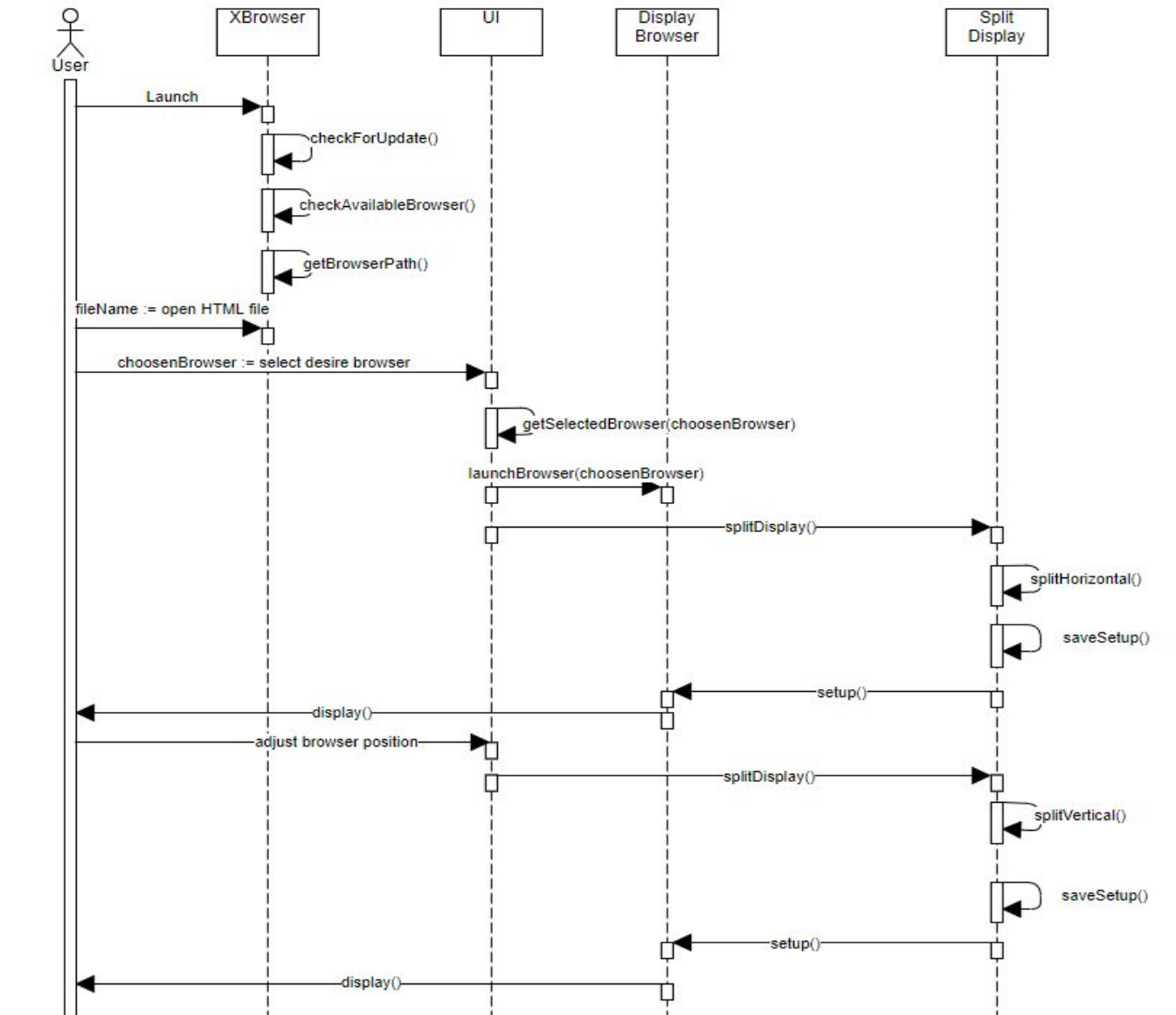
Upon loading up the atom application the user may choose to run the X-Browser extension, which will then prompt them with the default UI, requesting they input which browsers they would like to open and edit on. They will then have the ability to open a new file, or an existing file from which the program will display. The user can then edit this file and experience real-time updating of the website on the right hand side of the program. Upon completion of their website, or their session, the user may then save all progress and exit the program accordingly.

Sequence Diagram of this scenario:

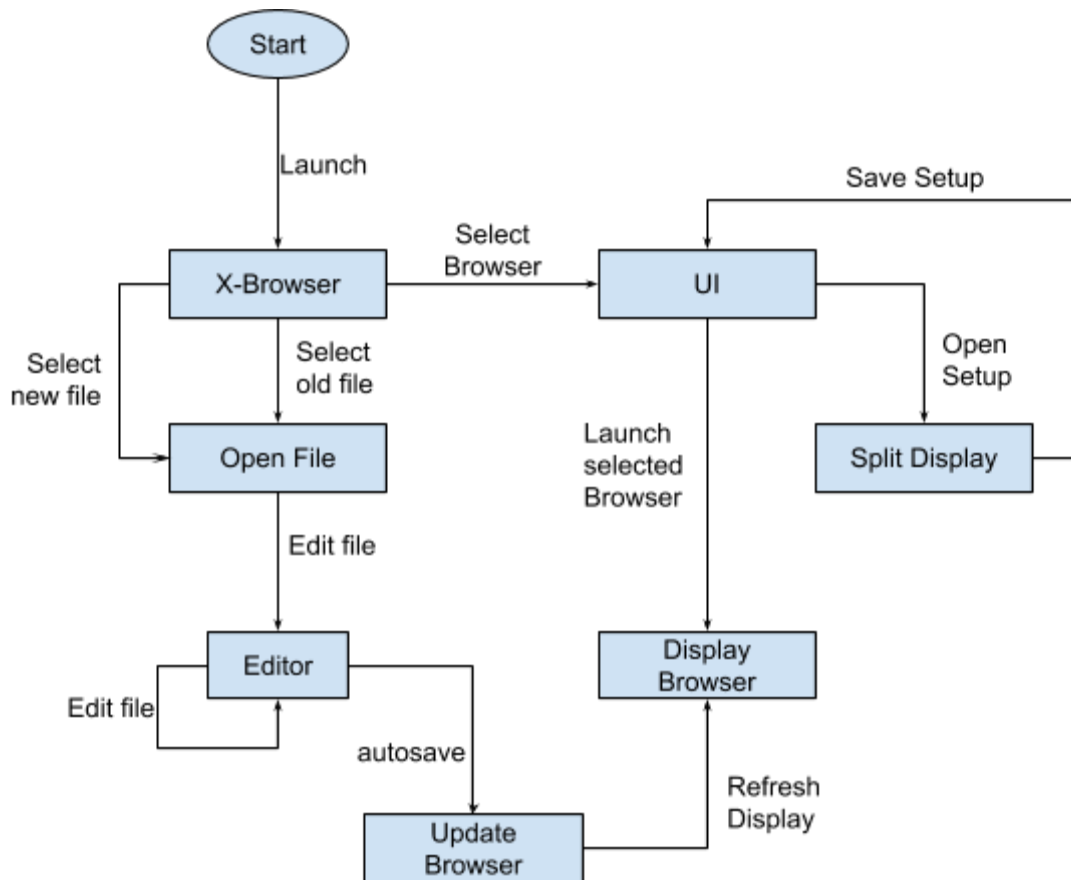


An existing user may launch the X-Browser plugin and then the user can modify the placement of the browsers. X-Browser will begin by displaying the default setup which will split the browsers in horizontal setup. Afterward, the user may modify this to a vertical setup which is considered as a “custom” setup that is saved in X-Browser for future use that allows users to avoid modifying the browser placement everytime X-Browser is launched.

This is a sequence diagram to show how this would work:



Below is a State Diagram that represents each state that our software can be in, and the transitions that make it go from state to state. The states are represented by rectangles, and the transitions represented by arrows.



5 Prototype

The prototype will demonstrate the implementation of X-Browser in the environment of Atom desktop text editor. The potential users will be able to experience the interface and features of X-Browser.

5.1 How to Run Prototype

System Requirements:

- Operating system: macOS 10.9 or later, Windows 7 and later, and Linux.

Network Requirements:

- A working internet connection is required to download Atom text editor.

Installing Atom desktop text editor:

- Follow instructions [here](#) to download and install Atom desktop text editor.

Installing X-Browser package in Atom:

1. After installing Atom, clone the git repository by type the following command:

```
git clone https://github.com/wxuhao/x-browser-prototype
```
2. Change the working repository to the freshly made x-browser-prototype directory
3. Run `apm link` to create a symbolic link to the `~/.atom/packages` directory
4. Refresh Atom

Using X-Browser:

- Once a package is installed in Atom, it will show up in the Settings View under the "Packages" tab as 'x-browser-prototype' (*Figure 5.2.2*. See section 5.2 below).
- To launch a web browser select 'Select browsers to launch' and mark web browsers desired before clicking the 'Launch' button (*Figure 5.2.3*. See section 5.2 below)

5.2 Sample Scenarios

The initial screen after launching Atom text editor:

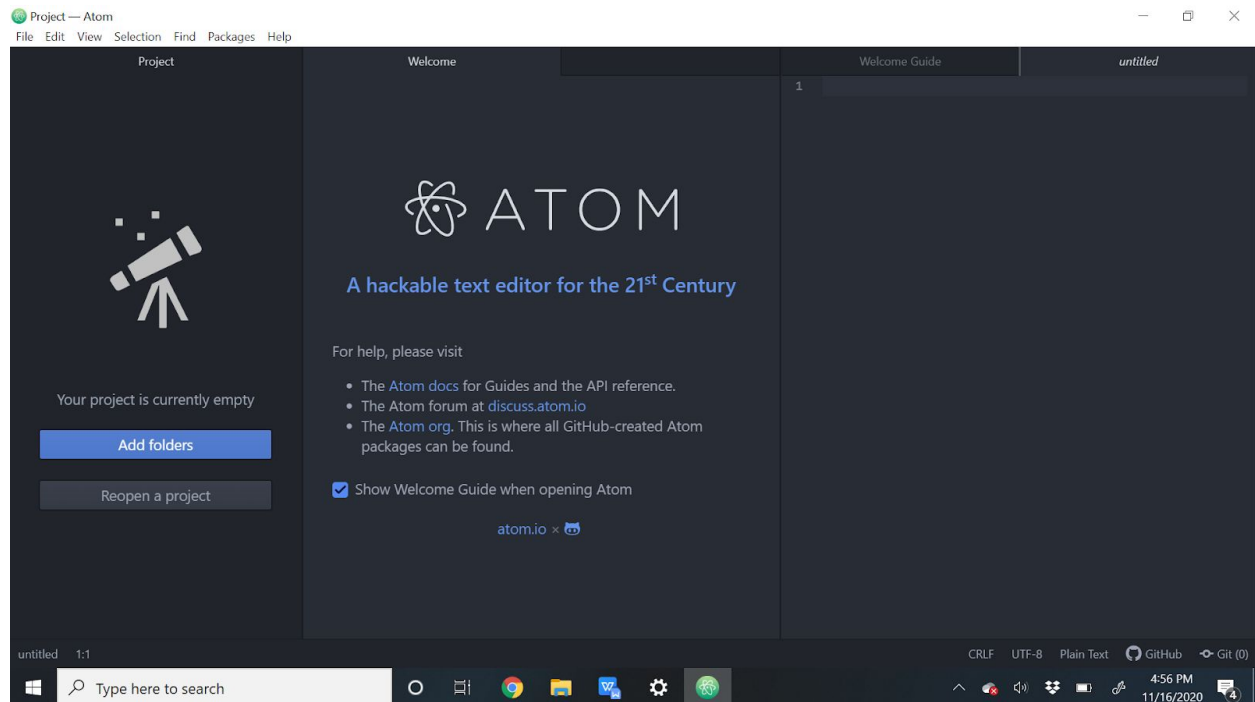


Figure 5.2.1

X-Browser in the 'Packages' menu:

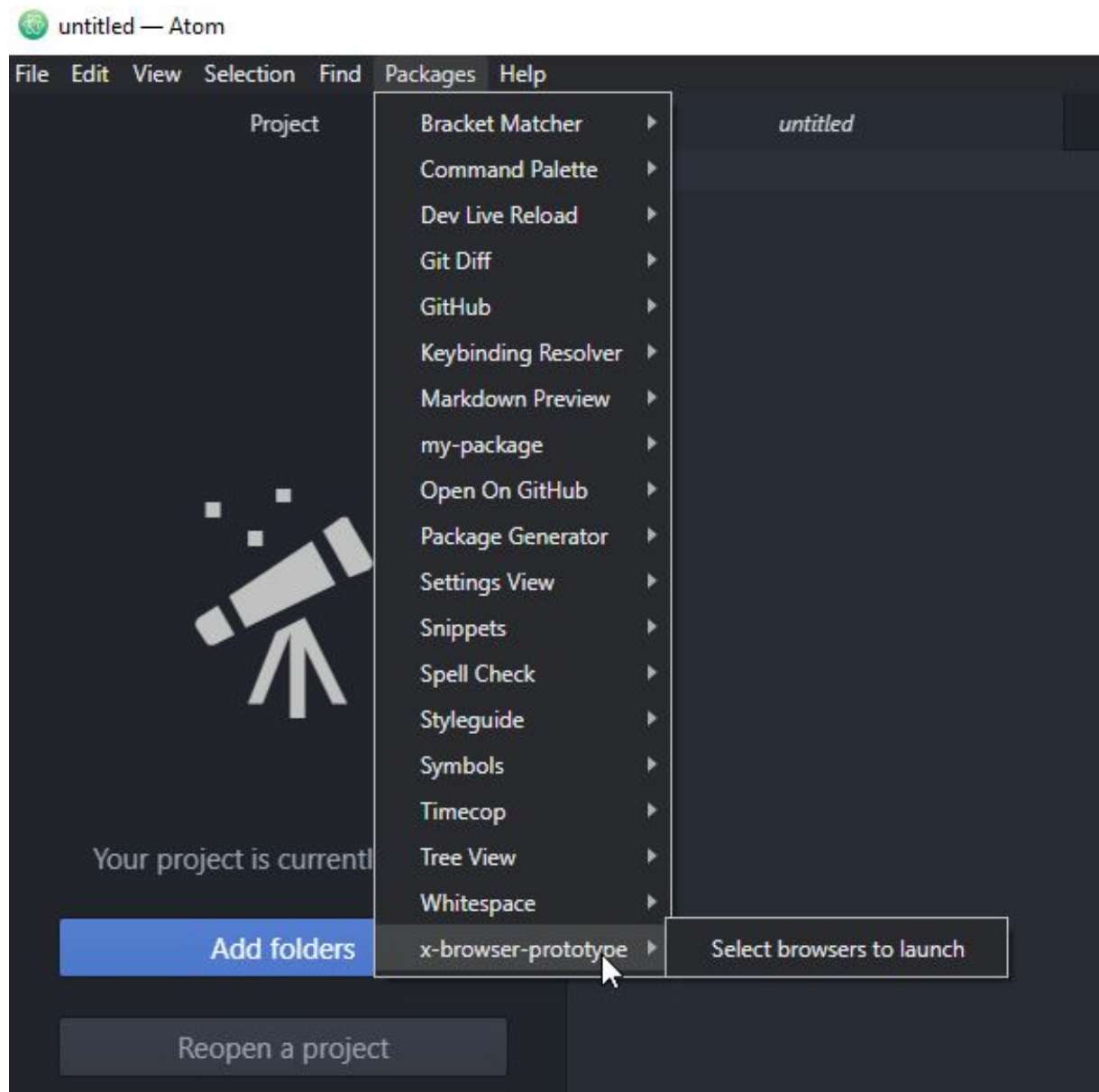


Figure 5.2.2

The 'Select browsers' menu for launching desired web browsers:

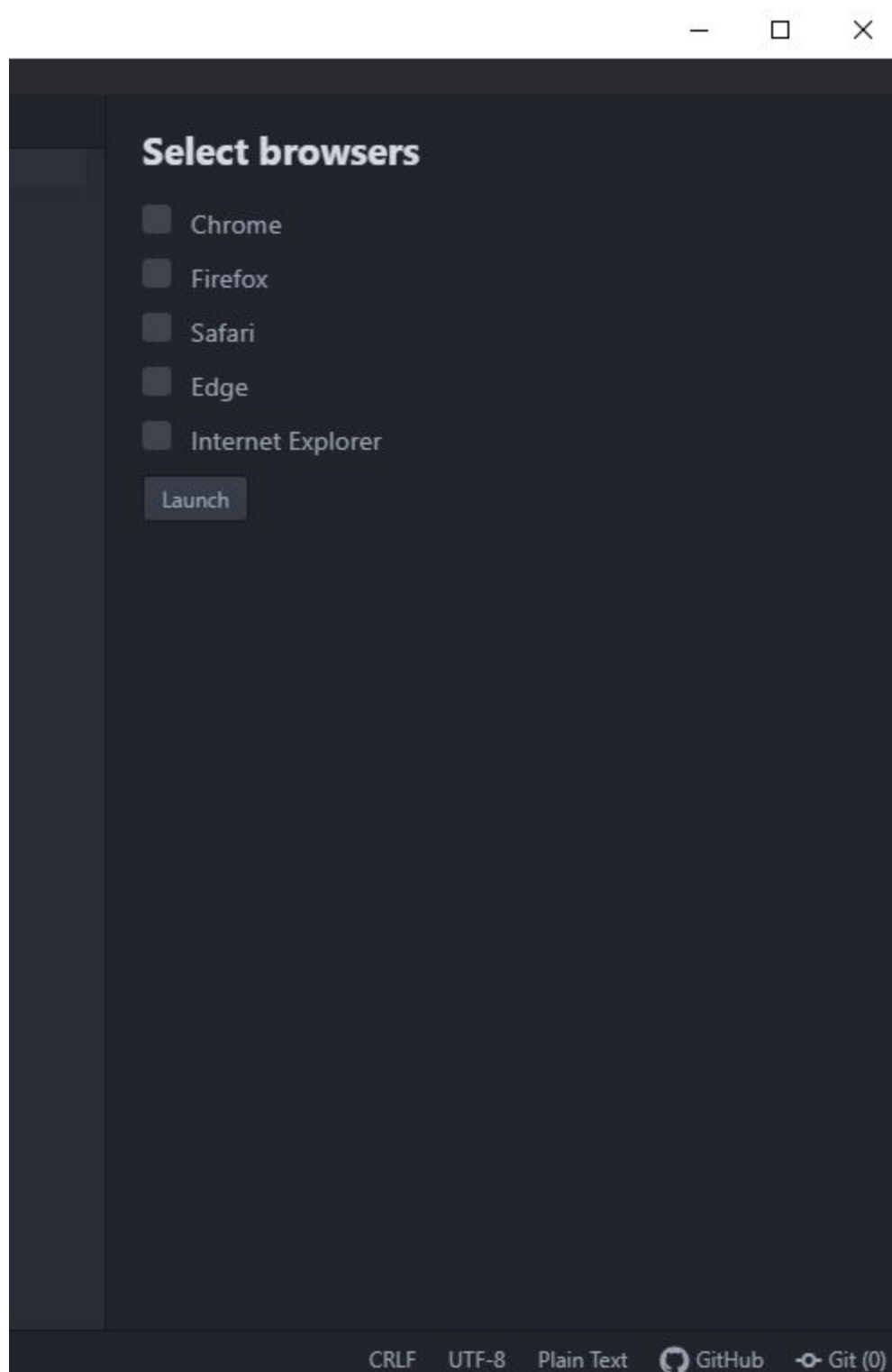


Figure 5.2.3

User selecting desire browser they want to see changes

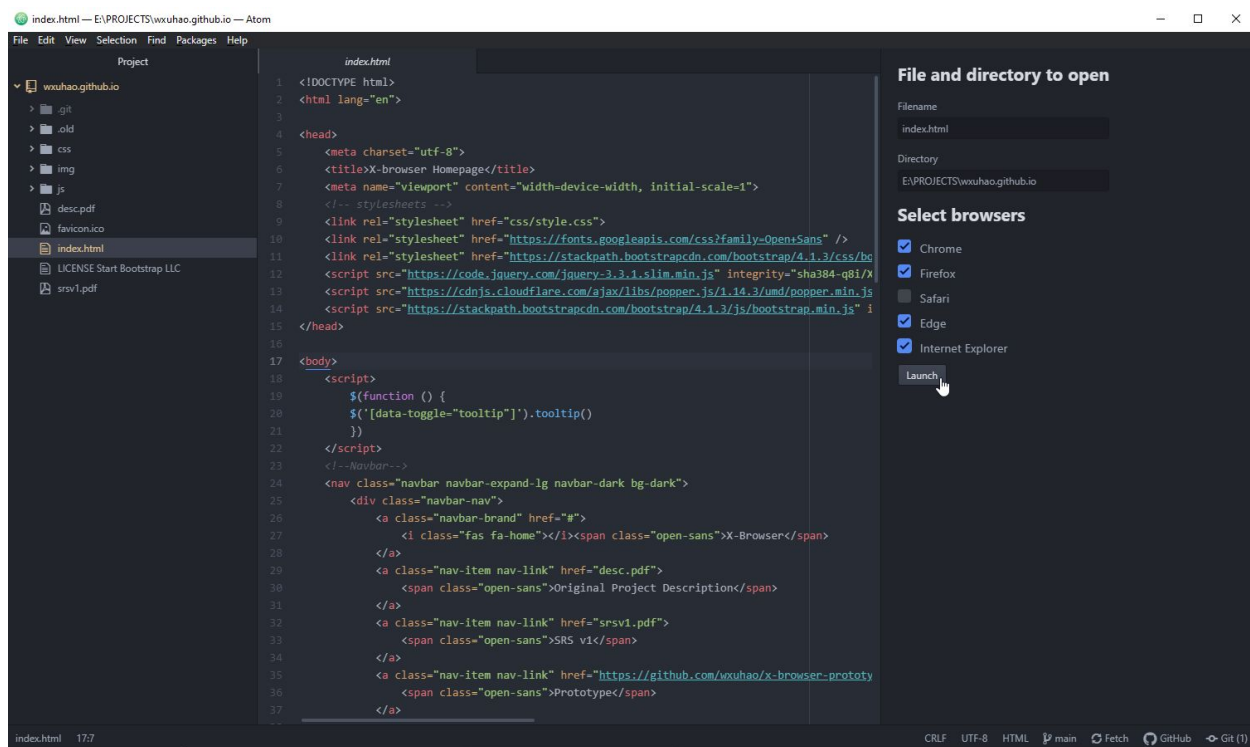


Figure 5.2.4

The selected browsers will pop-up and refresh as changes are made

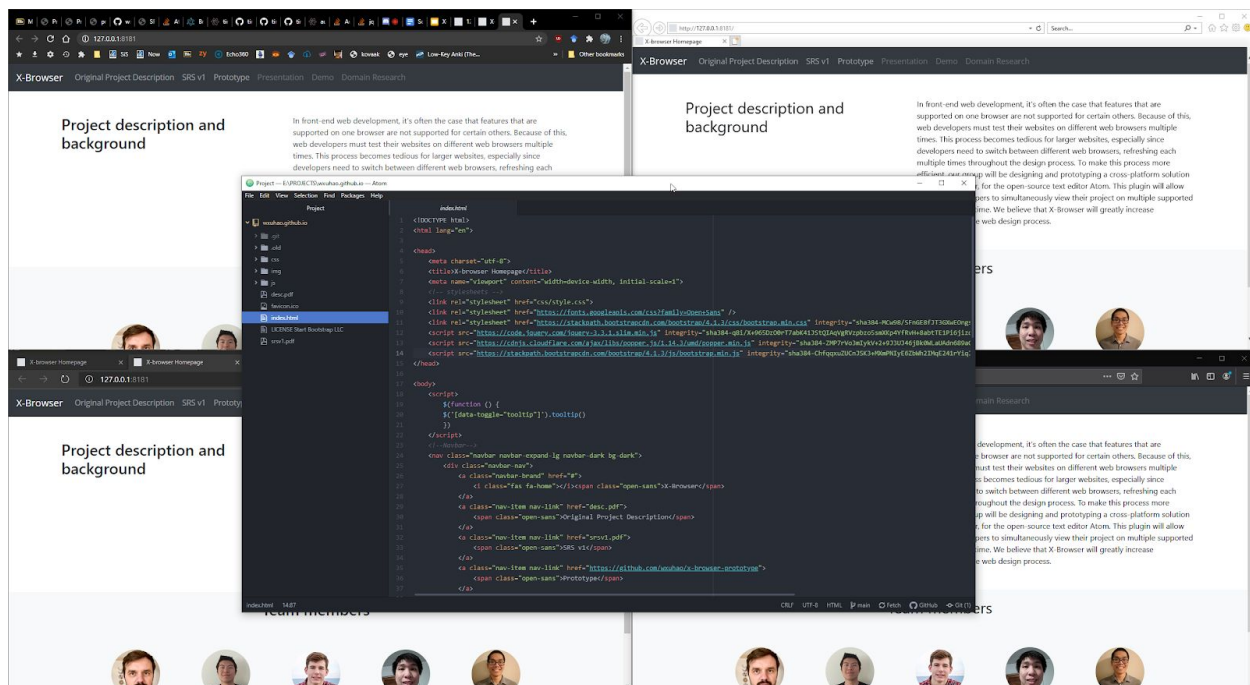


Figure 5.2.5

6 References

- [1] <https://gs.statcounter.com/browser-market-share/desktop/worldwide#monthly-202009-202009-map>
- [2] <https://flight-manual.atom.io/faq/sections/what-platforms-does-atom-run-on/>
- [3] M. Batbouta, L. Byan, J. Dom, M. Santana, “Software Requirements Specification (SRS) Project: ArithMagic”
- [4] D. Allen, B. Bean, N. Mezher, J. Summers, R. Terravecchia, “Software Requirements (SRS) Tap Tap Computation,” (2019)
- [5] A. Almanhady, D. Faubel, A. Witt, V. Gordiyevsky, “Software Requirements (SRS) Number Pop” (2019)

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly@uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.