Software Requirements Specification (SRS)
x-Browser

Team: Team 3
Authors: Andrew Rivard, Cassandra Harkins, Jacky McGrath, Pooja Patel, Tzur Almog
Customer: Front-End Developers
Instructor: Professor James Daly

Table of Contents

1 Introduction

x-Browser is a front-end development tool that combines a rich text editor with autocomplete functionality for real-time testing utilizing multiple browsers. This document is meant to outline the core functionality and requirements needed to be met by this project's

completion. High-level terminology and diagrams are used to describe the system both technically and visually for a better view of the full scope of requirements. A prototype of the final design is also included to provide a better understanding of what the final product is intended to look like and how it functions.

1.1 Purpose

The purpose of this document is to provide more detailed information about x-Browser, including functionality and requirements for the finished product. This document is intended for use by the developers and any interested parties that wish to know more about the specific details and the requirements required for the completed x-Browser.

1.2 Scope

x-Browser is a combination of a text editor and a browser preview of the code written in the text editor. The user will be able to create or open a file containing HTML, JavaScript, or CSS and edit the contents within. After saving the file(s), the preview windows of different browsers will automatically show the result of the written code along with any console output within those browsers.

This project will benefit front-end developers who want to see how their written code displays in different browsers without having to open up a new instance for each in separate applications. x-Browser shows both the written code as well as multiple browser views of that code side-by-side making a convenient testing tool for developers who want to have a more convenient way of testing and checking how their code displays and interacts with different browsers.

The ultimate goal of x-Browser is to create a platform that combines a text editor with the functionality of browser previews for ease of use and more optimized and convenient testing of front-end code. The intended application domain is for front-end development so that developers will be able to easily write and test their code.

1.3 Definitions, Acronyms, and Abbreviations

- **Application**: A program designed to fulfil a purpose, such as x-Browser
- **Browser**: A computer program with a GUI for displaying and navigating web pages
- **Build / Buildable**: The process of converting source code files into software that can be run on a computer
- **Google Chrome**: Browser application

- **Microsoft Edge**: Browser application
- **Firefox**: Browser application
- **Safari**: Browser application
- **Console**: Text output device for system messages
- **Cascading Style Sheet (CSS)**: Front-end programming language
- **Hypertext Markup Language (HTML)**: Front-end programming language
- **JavaScript**: Front-end programming language
- **Developers**: The intended users of x-Browser who will use the product for front-end development
- **Directories**: Folders that contain files or other folders. The parent directory contains all the other folders and/or files being worked on
- **Front-end development**: The process of converting data to a graphical user interface for users to view and interact with data. Front-end developers typically utilize CSS, HTML, and JavaScript to create a visually pleasing and easy-to-use user interface
- **Graphical User Interface (GUI) / User Interface (UI)**: Visuals that allow the user to interact with x-Browser
- **Kivy**: A Python framework used in the creation of x-Browser
- **macOS / Windows**: Operating Systems that support computer's basic functions
- **Preview**: A visual representation of what the user's written code will look like
- **Program**: The current project code the the user is working on
- **Project**: Used to reference the x-Browser development project as well as a collection of files that can be worked on by the user: typically a directory
- **Python**: A programming language used in the creation of x-Browser
- **Render**: Generating a new view shown to the user based on updated input
- **Screen / Window / Panel**: A visual partitioned area within an application. Can be the full application view or separate partitions within it depending on the context
- **Terminal**: Interface to the console that will type and execute text-based commands
- **Terms of Service (ToS)**: Legal agreements for use of a service that must be abided by when using that service
- **Text editor**: Program that allows the user to edit text
- **Toggle**: The action of switching between multiple states
- **Web pages**: A hypertext document on the Wold Wide Web

1.4 Organization

The rest of the document is as follows:

- Section 2: Covers how x-Browser should look and behave to the user including functionality, characteristics, constraints, and assumptions about the usage of x-Browser
- Section 3: An enumerated list of requirements needed for the x-Browser completed product
- Section 4: Descriptions of various included diagrams which include a use case diagram, a class diagram, multiple sequence diagrams, and a state diagram.

- Section 5: A description and visuals of the working prototype created for the concept of x-Browser along with instructions on how to run it
- Section 6: A list of all references used in the creation of this document and the creation of x-Browser
- Section 7: Contact information for any further questions or inquiries

2 Overall Description

In this section, the information covered will be related to what x-Browser should look and behave like. Interface descriptions, as well as constraints related to using the product will be discussed. Who can use x-Browser and why are key points in the section. Our goal in creating x-Browser was to create a user friendly application that is accessible, attractive, and safe.

2.1 Product Perspective

x-Browser is built to efficiently show the user a preview of their code on an assortment of browsers and is intended for front-end development. When the user starts up x-Browser, they

will have the option to choose to open a new file or an existing project. When the file or project loads, the user can see the project on the left with a preview of a browser on the right. The user has an option to toggle between a preview and edit mode for different views of the code and browser previews. When they click on preview mode, a temporary file will be saved of the code after a successful build. The preview will be of this temporary file in order to make sure the preview is buildable and viewable. x-Browser is compatible with the operating systems of both Windows and macOS. It requires storage in the form of directories to save temporary files for the purpose of previewing, as well as the storage required for the application itself. A large enough monitor is required to be able to view both the editable code and the tabs of preview browsers. A console view is also available for developer use in order to test the preview functions. Figure 1 shows the larger system that x-Browser will fit into.

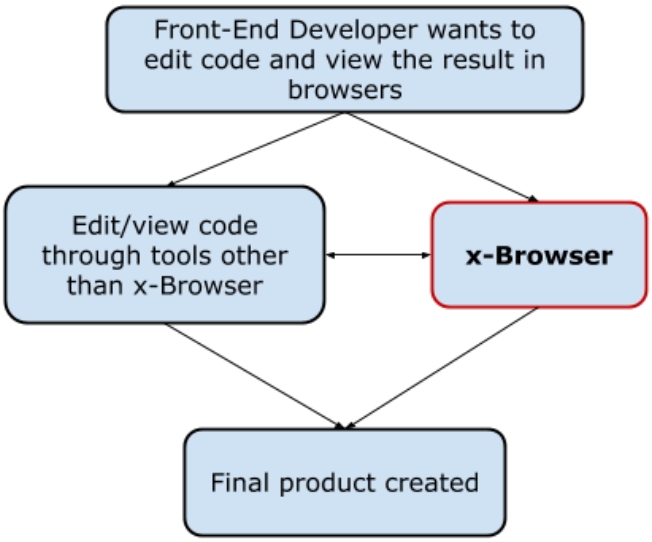


Figure 1: Bigger System

2.2 Product Functions

In order to function efficiently and effectively, x-Browser consists of several core functionalities that are performed when the application is used.

File:

- Create New File: Creation of a new file to edit. The file is created in a specific directory created for the new project.
- Edit Existing File: Open an existing file to preview or continue editing.
- Edit Mode: Edit the open file in the text editor panel.
- Preview Mode: View what the current file looks like on several tabbable browsers. File requires a successful build in order to view previews.
- Save: Save the file to its current location.

Display:

- Launch Screen: Shows Application Logo as well as Edit Existing File or Open New File options.
- Toggle Mode: Shows the Browsers in a grid layout, or the Edit Mode in a toggle fashion.
- Toggle Console: Opens a Console to interact with the previewed browsers.
- Tabs: Tabs in the Edit Mode reveal the previews in different Browsers for quicker viewing.
- Exit: Exits the application.

The function diagram is shown below in Figure 2, which shows the listed core functionalities that x-Browser will utilize and how they interact with each other.

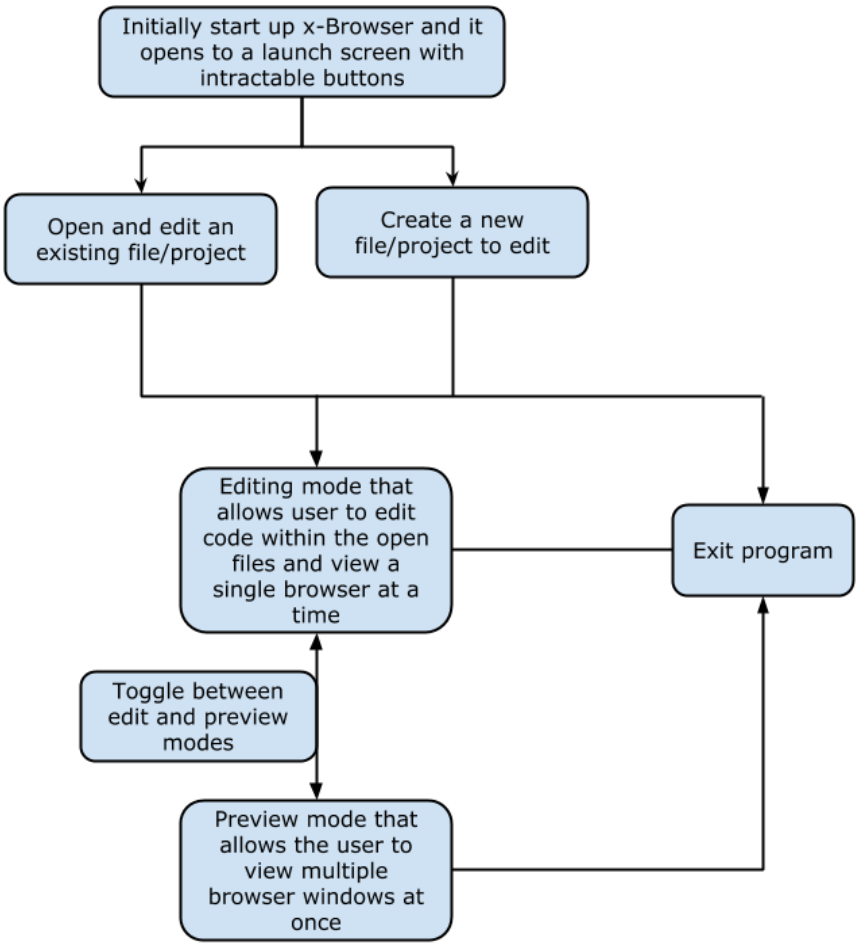


Figure 2: Function Diagram

2.3 User Characteristics

A user of x-Browser is expected to want to see their code side-by-side with a preview of that code being displayed in a browser. Typically this would be a front-end developer such as someone who works with GUI development. The user should also have knowledge on how to write buildable code that can be viewed within browsers. Users should also be able to operate a

mouse and a keyboard in order to move around the program and edit code within it given the visual application and no auditory descriptions.

2.4 Constraints

x-Browser is only designed to compile/interpret HTML, JavaScript, and CSS, to preview code on Chrome, Microsoft Edge, Firefox, and Safari browsers, and to run on Windows and MacOS operating systems. This means that any other languages, browsers, or operating systems may be incompatible with x-Browser.

In order to download the application, the user needs to have around 500mb available on the computer that is downloading x-Browser.

x-Browser does not have any support for people with disabilities at the moment, and requires a keyboard and mouse to function correctly.

2.5 Assumptions and Dependencies

Assumptions for x-Browser use would be that the computer being used to run x-Browser is able to support and have the resources needed for any language the user plans to use with x-Browser. The user should have an Internet connection in order to download Python and Kivy, which are necessary for running the application.

Dependencies for x-Browser use include the user having Python and Kivy installed, which download instructions are listed in section 5.4. The user must also have the browsers that they wish to view their code in already installed onto the computer running the x-Browser application.

2.6 Apportioning of Requirements

Currently x-Browser is only created to fulfil the purpose of creating an editor with real-time display of code in multiple browsers within the editor itself. Because of this there are some aspects that may not be covered in the current version, such as:

- Support for languages other than HTMl, CSS, and JavaScript
- Support for browsers other than Google Chrome, Firefox, Safari, and Microsoft Edge
- Support for Operating Systems other than Windows and macOS

- More views of the browser apart from page layout, page functionality, and browser console information

3 Specific Requirements

1. Program must be able to parse HTML, CSS, and JavaScript in a code editor
    1.1. The editor will alert the user if the code cannot compile/be interpreted or there are obvious errors within the user's code
    1.2. Must be able to switch editing files
    1.3. The user can write into the file displayed in the editor
2. Program must be able to display one or more browsers at a time
    2.1. The browser window will update when the user saves their code and display a notification when the user has unsaved changes

2.1.1. There is an option to preview current code without saving it
2.2. If the browser cannot display the code saved, there should be an error shown to the user in the browser window
2.3. Supported browsers are Firefox, Google Chrome, Safari, and Microsoft Edge
2.3.1. Program must abide by the Terms of Service for each browser supported
3. There are two display options the user can toggle between while editing
3.1. User can toggle between the two display options in either mode
3.2. In both modes, the user can see the console of the browser(s) shown
3.3. A preview mode that uses the whole window to display multiple browser screens
3.3.1. User can split the window into a four-quadrant setup that allows the browsers to be placed on the left/right and top/bottom in a single/double/quadruple sized quadrant
3.4. An editor mode that splits the window between an editor screen and a display screen
3.4.1. The code and browser screens will be displayed on a split-screen vertically so one shows up on the left and the other on the right
3.4.1.1. The two screens locations on left and right are interchangeable
3.4.2. Collapsible hierarchical display of the project files next to the code editor
3.4.3. The display screen will only show one browser
3.4.3.1. The browser window can be changed to the view of a different browser
4. Program must be compatible with both Windows and macOS systems
4.1. Program must work as expected on both systems

4 Modeling Requirements

This x-Browser use case diagram is used to show the typical user experience with the application. The user has the ability to toggle between edit and preview mode, view and/or edit code, view browser, and utilize the browser's console. The diagram is shown below in Figure 3 and the following tables describe each use case within it.

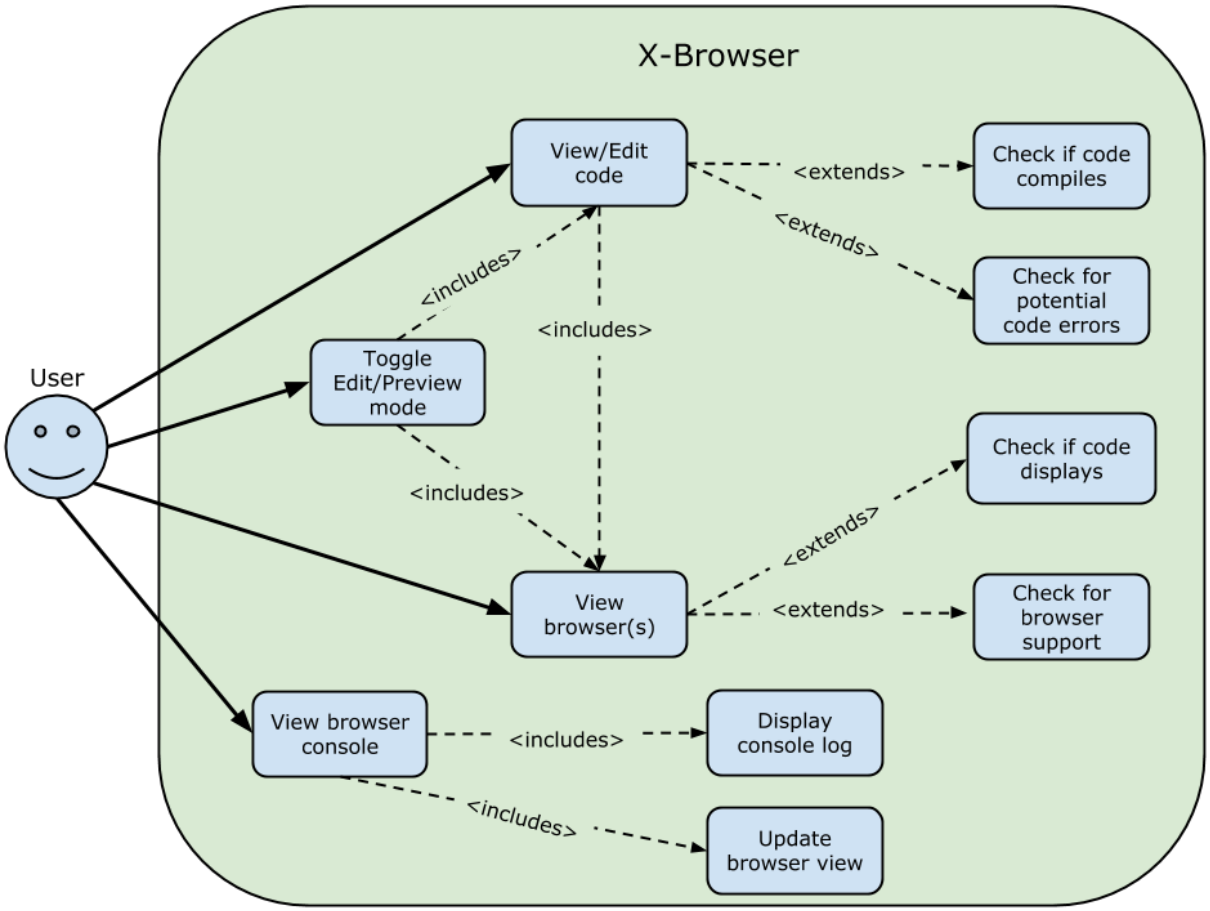


Figure 3: Use Case Diagram

Use Case Name:	**Toggle Edit/Preview mode**
Actors:	User
Description:	User can switch between edit mode and preview mode
Type:	Primary, Essential

Includes:	View/Edit code, View browser(s)
Extends:	N/A
Cross-refs:	Requirement 3
Uses cases:	None

Use Case Name:	**View/Edit code**
Actors:	User
Description:	User can view and edit code within edit mode
Type:	Primary, Essential
Includes:	View browser(s)
Extends:	Check if code compiles, Check for potential code errors
Cross-refs:	Requirement 1
Uses cases:	Toggle Edit/Preview mode

Use Case Name:	**View browser console**
Actors:	User
Description:	User can view the output of the browsers displayed
Type:	Primary, Essential
Includes:	Display console log, Update browser view

Extends:	N/A
Cross-refs:	Requirement 3.2
Uses cases:	None

Use Case Name:	**View browser(s)**
Actors:	User
Description:	User can see all of the browsers updated by the currently open project
Type:	Primary, Essential
Includes:	N/A
Extends:	Check if code is being displayed in browser, or if there are any errors when doing so, Check for browser support
Cross-refs:	Requirement 2
Uses cases:	Toggle Edit/Preview mode

Use Case Name:	**Check if code compiles**
Actors:	None
Description:	Once the code is saved, the compiler checks to make sure the resulting program is executable or not
Type:	Secondary, Essential
Includes:	N/A

Extends:	N/A
Cross-refs:	Requirement 1.1
Uses cases:	View/Edit code

Use Case Name:	**Check for potential code errors**
Actors:	None
Description:	While typing, the program checks for any potential errors the user is making that may prevent successful compilation
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 1.1
Uses cases:	View/Edit code

Use Case Name:	**Display console log**
Actors:	None
Description:	Displays all of the browsers' console logs to the user
Type:	Secondary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 3.2

Uses cases:	View browser console

Use Case Name:	**Update browser view**
Actors:	None
Description:	Renders all of the browser displays with the current project's code
Type:	Secondary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 2.1
Uses cases:	View browser console

Use Case Name:	**Check if code displays**
Actors:	None
Description:	Checks to make sure the code is being displayed in the browser or if there are any errors preventing it from loading fully
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 1.1

Uses cases:	View browser(s)

Use Case Name:	**Check for browser support**
Actors:	None
Description:	Checks for any lines of code that are incompatible with the browser it is trying to be displayed on
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 2.3
Uses cases:	View browser(s)

This x-Browser state diagram shown in Figure 4 describes the expected states of the software. The states focus on allowing the user easy navigation between the different windows for the easiest possible web development workflow.

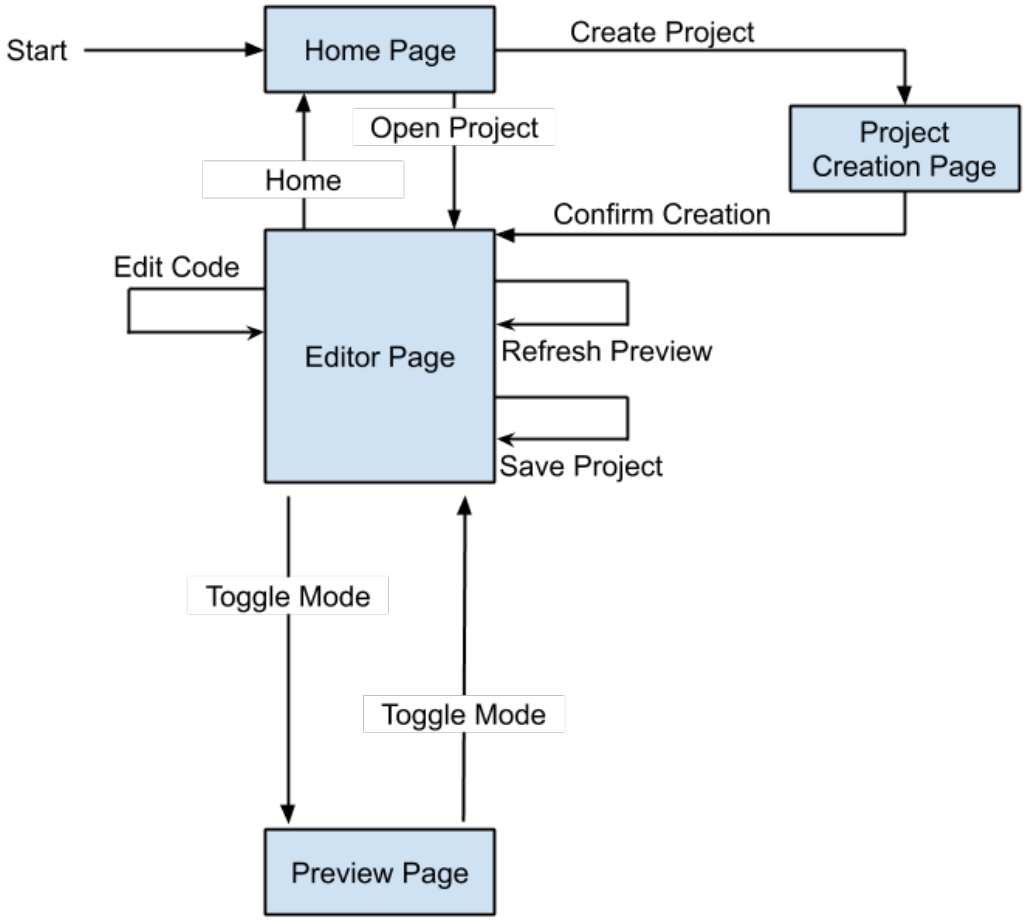


Figure 4: State Diagram

This x-Browser class diagram shown in Figure 5 describes the object oriented framework for our project. It shows the hierarchical structure of controllers from top to bottom.

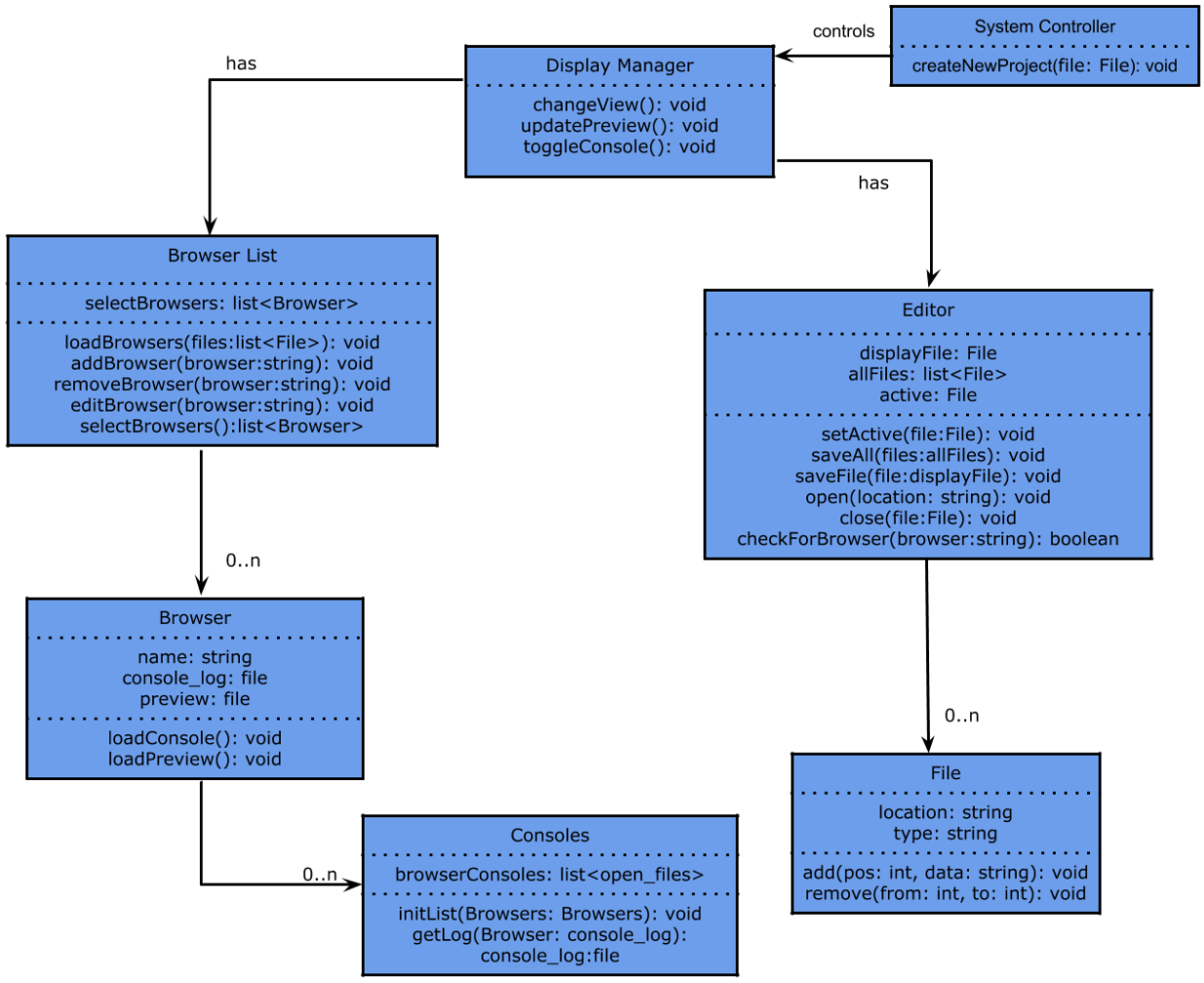


Figure 5: Class Diagram

Element Name		**Description**
System Controller / Home screen		Is the home page. Opens the user into a project
Attributes:		
Operations:	createNewProject(file: File): void	Creates a new file to begin a new project of the user

Element Name		**Description**
Display Manager		The display manager controls the window the project is on. It also has the controls to switch between the Editor and Browser List.
Attributes:		
Operations:	changeView(): void	Changes view between Editor and Browser List
	updatePreview(): void	Refreshes web pages based on the open files
	toggleConsole(): void	Displays and hides the console

Element Name		**Description**
Browser List		Shows multiple browsers. Allows the user to see differences in changes by measuring the websites side by side.
Attributes:	selectBrowsers: list<Browser>: void	A list of browsers that the program can display
Operations:	loadBrowsers(files:list<File>): void	Loads all browsers available given files
	addBrowser(browser:string): void	Adds a new browser view
	removeBrowser(browser: string): void	Removes a browser from the list
	editBrowser(browser: string): void	Edit a browser view
	selectBrowser(): list<Browser>	Selects the browser that the user wants to view when there can only be a limited number of browsers on the screen

Element Name		**Description**
Editor		The text editor that the user can read and write in files within a project
Attributes:	displayFile: File	The file that is being displayed to the user to work on at the moment
	allFiles: list<File>	List of files that are active
	active: File	The file is actively being worked on at the moment
Operations:	setActive(file:File): void	Sets the file to be active
	saveAll(files:allFiles): void	Saves all files that are active
	saveFile(file:displayFile): void	Saves the file that is being displayed
	open(location: string): void	Opens a path to a new file to be worked on
	close(id: int): void	Closes a file that was being worked on
	checkForBrowser(browser: string): boolean	Checks if the code written is compatible with a given browser and returns if yes or not

Element Name		**Description**
Browser		The browsers available to view the written code in
Attributes:	name: string	Name of the browser
	console_log: File	The console log of the browser
	preview: File	The preview file of what would be seen on the browser
Operations:	loadConsole(console_log:file): void	Load the console log given in the browser
	loadPreview(preview:file): void	Load the preview of the browser with the open file(s)

Element Name		**Description**
Consoles		All of the browser console logs when the code is saved or previewed
Attributes:	browserConsoles: list<console_log>	List of available browser console outputs
Operations:	initList(Browsers: Browsers): void	Initialize the list of browser consoles
	getLog(Browser: console_log): console_log:file	Get the log of each browser console

Element Name		**Description**
File		Files included in the project the user has open
Attributes:	location: string	Location of the file
	type: string	Type of file
Operations:	add(pos: int, data: string): void	Adds a file to the program to edit
	remove(from: int, to: int): void	Removes a file from the program

The sequence diagram shown in Figure 6 shows the scenario of when the user wants to create a new project. They need to supply a name and a location of which a directory is made as well as the project is opened. This then requires the user to select what browsers to open for testing.

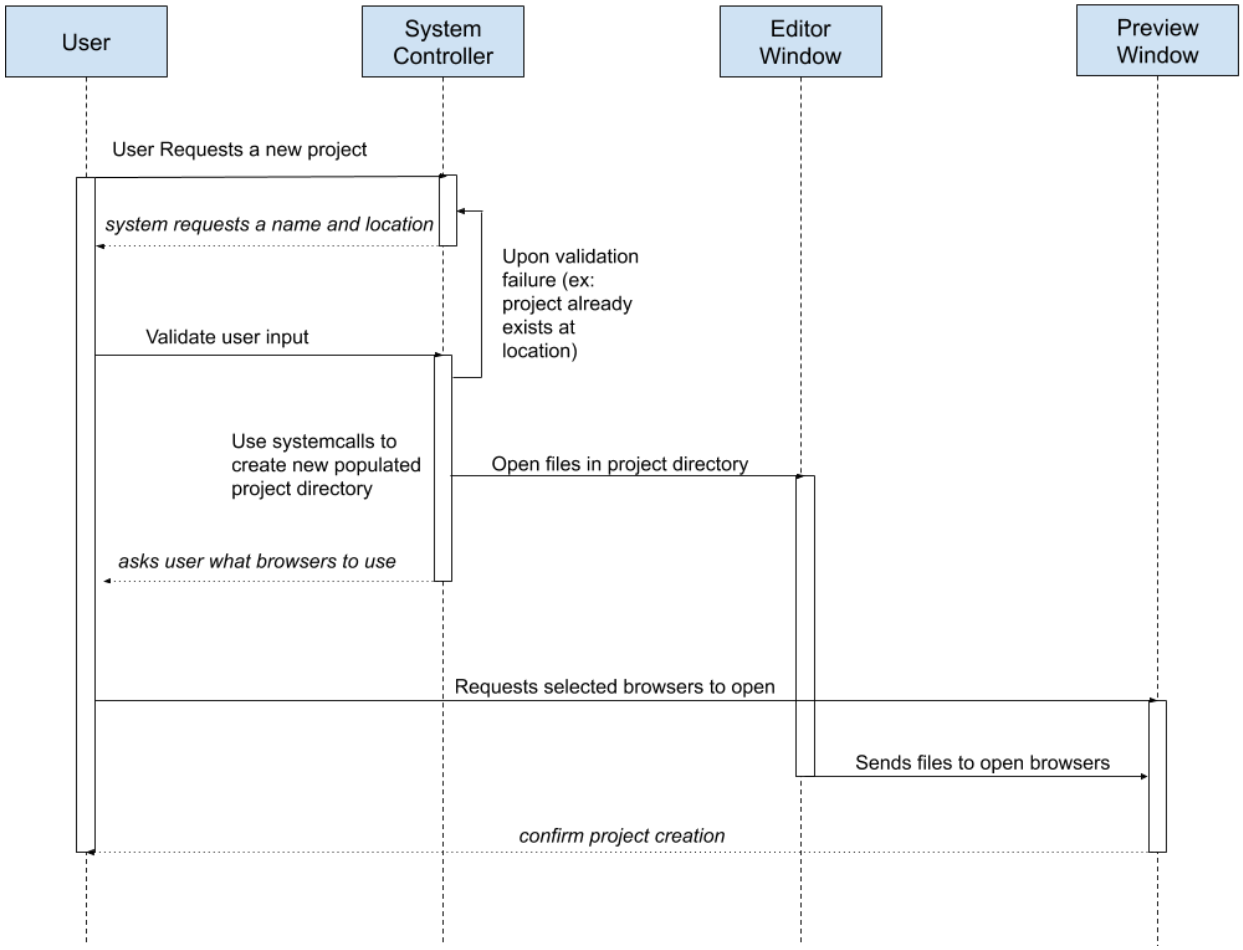


Figure 6: Sequence Diagram 1

The sequence diagram shown in Figure 7 shows the scenario of when the user has a project open and now wants to make a change that will update the browser view to reflect that change. They will then switch to preview mode to view all browsers side-by-side, and then save the changes to the file.

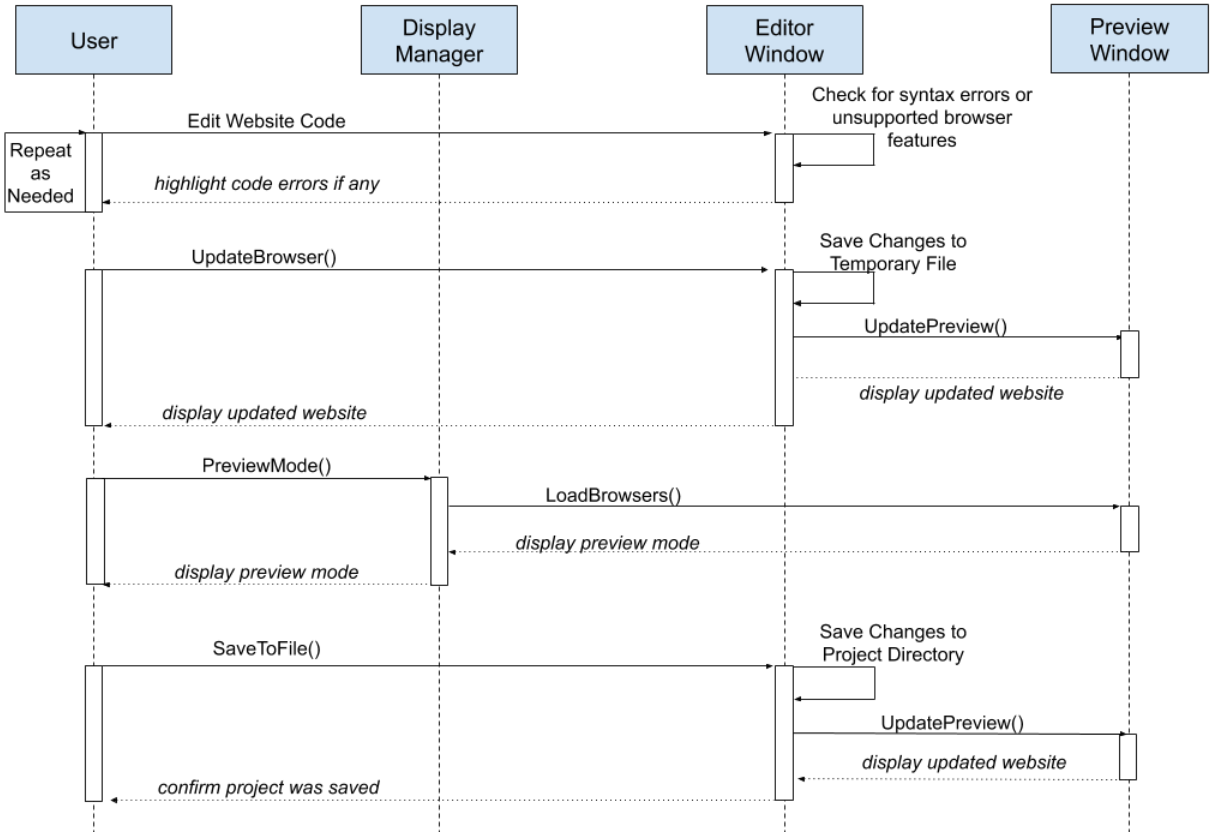


Figure 7: Sequence Diagram 2

5 Prototype

The prototype for x-Browser shows basic UI components and how the user should interact with the system. In future prototype implementations, we will demo more of the core functionality of the system (editor, browser rendering, etc.) This prototype keeps basic user interaction in mind, focusing on simplicity to navigate so that the users can focus more on getting work done. It consists of 2 workflows: editor mode and preview mode. Preview mode has viewable windows for each browser currently opened (in the form of a screenshot for UI prototype purposes). Editor mode has a panel on the left to show files in the project directory, text editor on the left half of the window, and a browser preview on the right side. This browser preview shows only one browser at a time, but the user can choose what browser is currently showing. In both modes, there is a collapsible panel at the bottom that will show the consoles for all open browsers.

5.1 How to Run Prototype

To Run from Source Code:

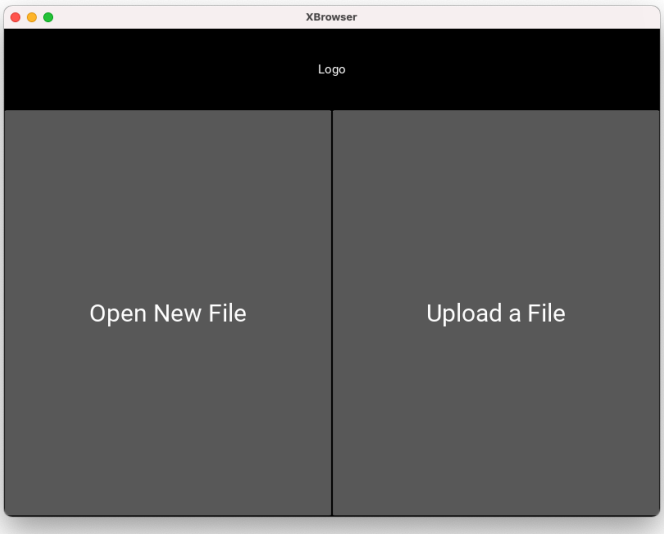
1) Ensure you have python3 downloaded:
    a) type 'python3 --version' or 'python --version' in terminal. It should reply with a version number 3..
    b) if you get an error or don't have version 3+ follow these instructions
        i) Install Python: https://realpython.com/installing-python
        ii) NOTE: We encountered issues when using a python version newer than 3.7
2) Download Kivy (dependency for this prototype) :
    a) Windows Installation: https://kivy.org/doc/stable/installation/installation-windows.html
    b) OS X Installation: https://kivy.org/doc/stable/installation/installation-osx.html
    c) Linux Installation: https://kivy.org/doc/stable/installation/installation-linux.html
3) Run the program:
    a) Download the source code here
    b) https://github.com/jackyyym/x-browser
    c) Open the downloaded directory in terminal
    d) run 'python3 xbrowser.py'
4) Exit the program:
    a) Simply close the window

Downloading Windows Executable:

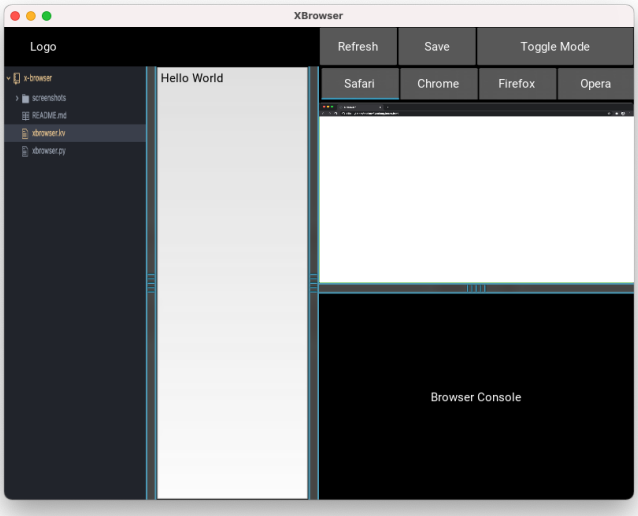
1) Download the latest version from https://github.com/jackyyym/x-browser/releases
2) Unzip the archive file and run *XBrowser.exe - Shortcut*

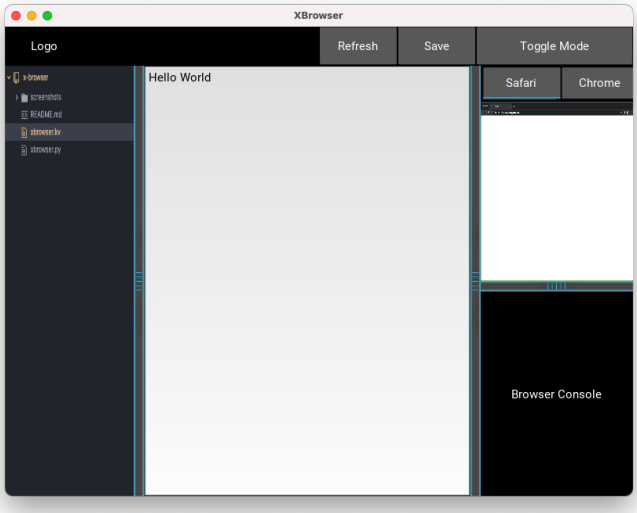
5.2 Sample Scenarios

A web developer is tasked with a job to make a simple webpage. From the menu page, the developer selects ‘Open New File’; this launches the editor mode. From there, the developer can resize each individual panel to their liking. Once the developer sets up their environment how they want it, they begin typing code into the editor. If they want to see changes rendered without saving the ‘Refresh’ button will do that. In our case, the developer wants to save the file and preview changes, so they press the ‘Save’ button. After viewing the changes on Safari, they decide to check how the website looks on Firefox. The developer wants a side by side view of each browser, so they press the ‘Toggle Mode’ button to enter preview mode. From here the developer notices that each browser rendered the webpage differently and can act accordingly.

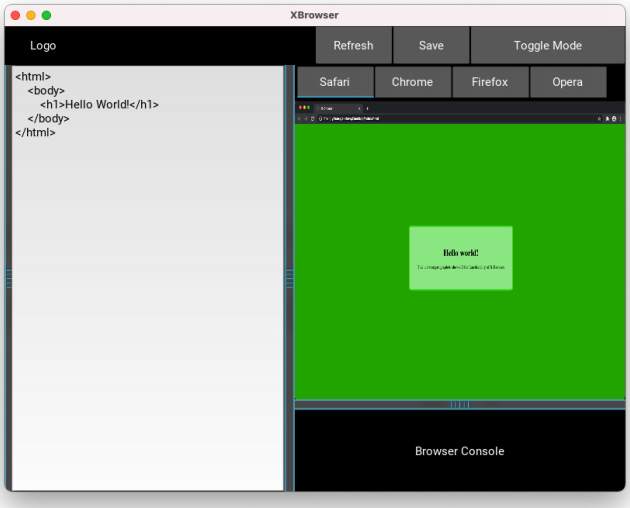


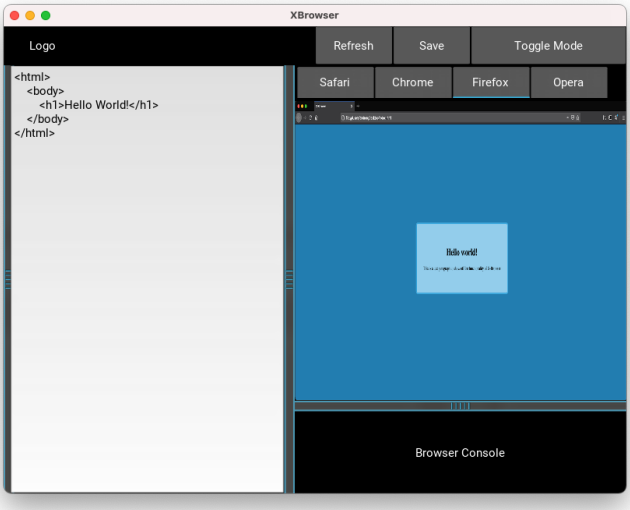
The Developer opens xBrowser and selects ‘Open New File’



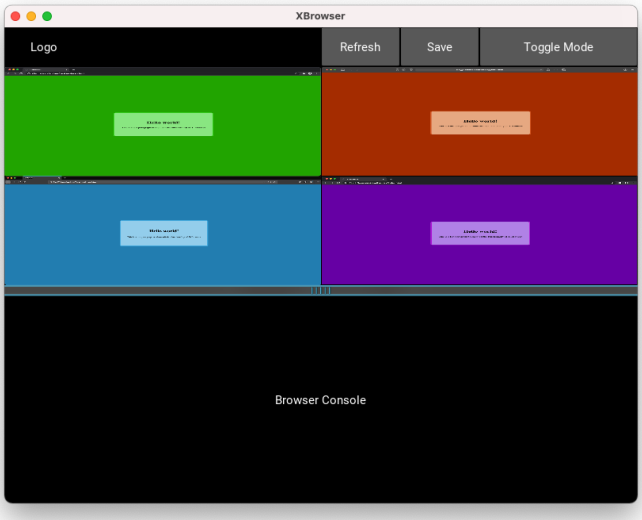


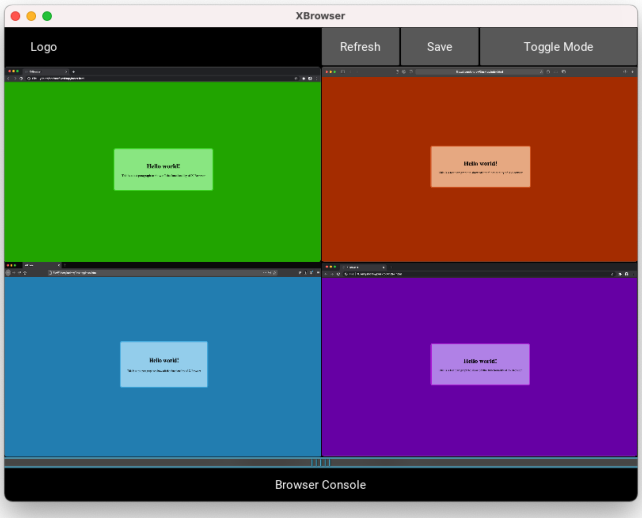
The editor is launched and the developer makes the text editor larger





Text is entered and the developer clicks ‘Save’. The preview to the right then loads. After that, the developer switches to the Firefox tab





To view all browsers, the developer presses the ‘Toggle Mode’ button and resizes the console to be smaller.

6 References

[1] A. Almanhady, D. Faubel, A. Witt, V. Gordiyevsky, “Software Requirements (SRS) Number Pop” (2019)

[2] A. Rivard, C. Harkins, J. McGrath, P. Patel, T. Almog, “x-Browser-site,” Available at: https://jackyyym.github.io/x-browser-site, GitHub, (2020)

[3] A. Rivard, C. Harkins, J. McGrath, P. Patel, T. Almog, “x-Browser,” Available at: https://github.com/jackyyym/x-browser, GitHub, (2020)

[4] A. Santos, D. Amos, D. Bader, J. Anderson, J. Jablonski, J. Mertz, J. Schmitt, J. Sturtz, M. Driscoll“Python3 Installation & Setup Guide,” Available at: https://realpython.com/installing-python, Real Python, (2020)

[5] D. Allen, B. Bean, N. Mezher, J. Summers, R. Terravecchia, “Software Requirements (SRS) Tap Tap Computation,” (2019)

[6] “Kivy Download” Available at: https://kivy.org/#download, Kivy (2019)

[7] M. Batbouta, L. Byan, J. Dom, M. Santana, “Software Requirements Specification (SRS) Project: ArithMagic”

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at 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.