Bachelor's Programme in Science and Technology

Extended Reality Layout Framework

A Design Framework for Web Content Layout in XR

Aaron Campbell

Aalto University

BACHELOR'S THESIS

Aalto University
BACHELOR'S THESIS 2017

Extended Reality Layout Framework

A Design Framework for Web Content Layout in XR

Aaron Campbell

Thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Technology.
Otaniemi, 29.4.2023

Supervisor: Professor Martin Andraud
Advisor: Professor Antti Oulasvirta

Aalto University
School of Science
Bachelor's Programme in Science and Technology

Copyright © 2023 Aaron Campbell
This work by Aaron Campbell is licensed under the CC-BY-SA 4.0 license, unless otherwise stated.

Aalto University

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Abstract

Author Aaron Campbell

Title Extended Reality Layout Framework: A Design Framework for Web Content Layout in XR

Degree programme Aalto Bachelor's Programme in Science and Technology

Major Digital Systems and Design **Code** ELEC3056

Supervisor Professor Martin Andraud

Advisor Professor Antti Oulasvirta

Level Bachelor's Thesis **Date** 29.4.2023 **Pages** 5 + 44 **Language** English

Abstract

This thesis proposes the Extended Reality Layout (XRL) framework, a tool for web developers to transition web content into immersive XR environments. The framework focuses on design freedom, cross-platform and legacy compatibility, and enhanced browsing capabilities in XR contexts. A proof-of-concept implementation utilizing Python, Flask, and Beautiful Soup modules is employed, and case studies applying the XRL framework to various website types demonstrate its adaptability to serve diverse web content types. As extended reality (XR) technologies become more widely adopted, the demand for effective web-based experiences and applications in these immersive environments will grow. The XRL framework addresses the limitations of traditional web design practices in XR contexts and enables seamless implementation with a class-attribute-based system.

Keywords Web Design, Extended Reality (XR), Virtual Reality (VR), Augmented Reality (AR), Mixed Reality (MR), Immersive Browsing Experience

Contents

Abstract **iii**

Contents **iv**

1. Introduction **1**

2. Background and Related Work **3**

2.1 Extended Reality (XR) 3

2.2 XR Hardware and Platforms 4

2.3 Traditional Web Design Approaches 4

2.4 XR User Interface Design Patterns 5

2.5 Related Work 6

3. Extended Reality Layout Framework **8**

3.1 Design Problem 8

3.2 Design Objectives 9

3.3 Design 10

3.4 Technical Implementation 12

3.5 Framework Assessment 13

4. Case Studies **14**

4.1 Info 15

4.2 Search 17

4.3 Media 18

4.4 Mail 21

5. Conclusion **25**

5.1 Topics for Further Research 26

5.1.1 Cross-Reality Dimensions 26

5.1.2 Mass Adoption 26

References **28**

A. Appendix **30**

A.1 Mock Sites 30

A.1.1 Shared Styling 30

A.1.2 Shared Script 31

A.1.3 Info 32

A.1.4 Mail 33

A.1.5 Media 34

A.1.6 Search 36

A.1.7 Store 38

A.2 Flask Server 39

A.2.1 Flask App 39

A.2.2 XRL Framework Emulator 41

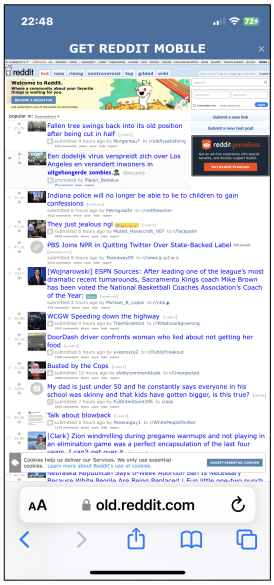
1. Introduction

The internet has become an integral part of many people's everyday lives. From work, to education, to personal entertainment, people are becoming more reliant on web services to help them access and process information. As extended reality (XR) technologies become more widely adopted, the demand for effective web-based experiences and applications in these immersive environments will grow.

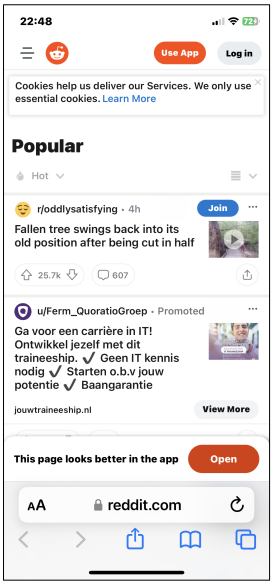
Historically, web design has been driven by technological advancements and user needs [1]. With the advent of the smartphone, small touchscreens have become one of the predominant means of browsing the internet. However, the small screen size made it more difficult for users to read and interact with content on the web. This prompted a new wave of mobile-first design philosophies, such as responsive web design and design minimalism, to address this issue [1]–[3]. Figure 1.1 illustrates this shift in design strategy, showing the transformation of the reddit.com interface into a mobile friendly design.

Like smartphones, the rise of extended reality (XR) technologies, which include virtual reality (VR), augmented reality (AR), and mixed reality (MR), have the potential to significantly change the way people engage with technology. The unique capabilities and limitations of XR technology necessitates a rethinking of traditional web design approaches to fully capitalize on the strengths of this new interaction medium.

One critical distinction of XR technologies is that the limitations to the interaction space differ from those of traditional interaction technologies. With traditional technologies, interaction spaces were primarily constrained by physical factors such as physical pixel width, screen size, and physical space needed to store hardware such as monitors, projectors, or computers. In XR contexts, these constraints no longer apply, as the screen space extends to any area within the user's field of view and the



(a) Old Reddit mobile interface.



(b) Current Reddit mobile interface.

Figure 1.1. Figures showing the UI changes made by Reddit to accommodate touchscreen devices.

hardware has shrunken to a wearable scale. However, XR is not without its own set of limitations. Presently, the most pertinent technological constraints for XR in the context of web browsing include viewing resolution and field of view. [4].

This thesis aims to develop a new web design framework to address the shortcomings of traditional web design practices in XR contexts. The proposed extended reality layout (XRL) framework offers a design approach that takes into account the unique features of XR technologies while considering both web-developer and end-user perspectives.

The thesis is organized into the following sections. Chapter 2 (Background and Related Work) covers relevant background information and related work, while identifying the limitations of these existing efforts. In Chapter 3 (Design Framework), the proposed philosophy and framework are detailed, offering a rationale for their design choices. Chapter 4 (Case Studies) presents an in-depth examination of five cases that showcase the successful realization of the design goals by implementing the proposed framework, along with a critical assessment of any shortcomings and a discussion on potential future improvements. Lastly, Chapter 5 (Conclusion) summarizes the results of the thesis and considers the scope and need for future research.

2. Background and Related Work

This section provides a comprehensive overview of the background areas which inform the design approach of the proposed method in the subsequent sections. Section 2.1 offers an introduction to XR, discussing the various technologies and devices that fall under this broad term. Section 2.2 discusses the leading XR hardware and platforms, outlining examples in both virtual and augmented reality. Section 2.3 reviews traditional web design approaches, specifically focusing on responsive web design and the component web model. Section 2.4 explores XR user interface design patterns, emphasizing the concept of 'panoramic vision' and potential interaction methods. Lastly, section 2.5 discusses related work in adapting web content for XR browsing experiences, referring to studies conducted by Michael Nebeling and Xing et al. as reference points for the thesis approach.

2.1 Extended Reality (XR)

The term 'extended reality' (XR), sometimes referred to as cross reality or mediated reality, is an encompassing term for technologies which broaden or enhance the user's perception or experience of the world [5]. While the term includes various technologies such as smart watches and welding goggles, it is colloquially used to refer to augmented, virtual, and mixed reality technologies [5]. These devices often take the form of head-mounted displays (HMDs), which alter the visual perception of the user allowing them to interact with digital content as if it were part of their physical environment. There is ongoing debate about the use of the term XR and the need for more precise definitions [5]. However, for the purposes of this thesis the term extended reality will be used to refer specifically to immersive augmented, virtual, and mixed reality technologies and devices.

In the context of XR, it is helpful to clarify a few additional terms that will be discussed later in the thesis. First, 'viewing resolution' refers to the quality and sharpness of the display device, which may influence the ability of a user to read textual information or detect intricate details on virtual content. Second, 'field of view' relates to the range of the visible environment within the XR experience, without requiring any head movement from the user. This factor can impact the ability of a user to navigate and engage with digital content.

2.2 XR Hardware and Platforms

The current leading platforms in XR technologies primarily involve head-mounted displays (HMDs). Consumer virtual reality HMDs comprise well-known products such as the Meta Rift and Quest lines (formerly Oculus products), the HTC Vive, and the HP Reverb. Augmented reality HMDs, on the other hand, include devices like the Microsoft HoloLens and Lenovo ThinkReality glasses. Additionally, smartphones often serve as accessible tools for more basic AR and VR experiences. [6].

Software for XR technologies largely comes in the form of proprietary, stand-alone applications tailored for specific devices, such as Android development for the Meta Quest. A significant portion of XR software development takes place in game engines like Unity and Unreal Engine, which offer powerful tools for creating captivating virtual experiences. A notable development in the XR software domain is WebXR, an emerging web standard for crafting fully immersive/customized solutions, such as online VR gaming experiences [7]. These experiences are available via the browser of a supported XR device. Developers can also utilize the popular WebGL and babylon.js platforms to build WebXR applications, enabling greater flexibility and innovation in XR content production [8].

2.3 Traditional Web Design Approaches

To explore a new approach to web content browsing in extended reality (XR), it is valuable to examine existing web development techniques. By understanding these well-established practices and principles, their relevance and possible adaptation to the unique context of XR environments can be considered.

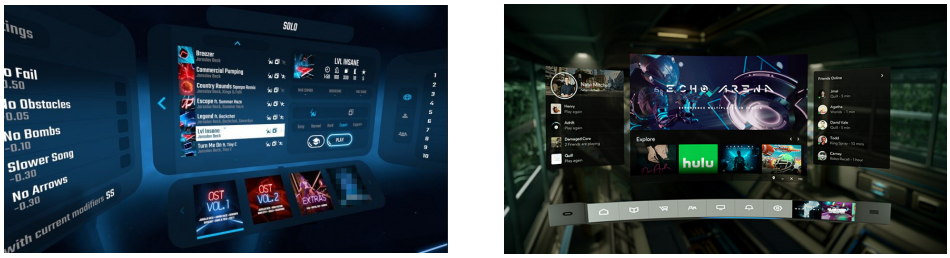
Responsive web design is a crucial principle in modern web development that ensures websites are visually appealing and user-friendly on a wide range of devices and screen sizes [3]. This approach involves utilizing flexible grid layouts, fluid images, and CSS media queries to create a seamless browsing experience tailored to the device of each user. The core concepts of responsive web design can potentially be extended to address the unique challenges presented by XR environments.

In conjunction with responsive design, the component web model has emerged as an effective way of building websites by utilizing modular and reusable components [9]. This approach encourages web developers to create self-contained units of functionality, such as navigation bars, image sliders, or search forms, to promote consistency and flexibility across a site. By developing components that can adapt their style, layout, and behavior according to different devices, environments, and user interactions, the component web model can streamline the process of creating responsive web content. Extending responsive design and the component web model to XR contexts may yield valuable insights into how web content can adapt to the unique requirements of this emerging medium.

2.4 XR User Interface Design Patterns

One of the major benefits of an XR interface over traditional 2D interfaces is the concept of 'panoramic vision'. This is a term defined by Xing et al. to describe the capability of arranging content around the user in 3D space; overcoming the traditional 2D limitations and providing a clearer and more immersive view of the content [10]. This idea can be seen used in various existing XR user interfaces such as in the menu of the VR game Beat Saber (figure 2.1a) or the Oculus Dash console (figure 2.1b). By splitting the UI into seperated element groups and arranging them in 3D space around the user at the center, the user is better able to see and interact with the content [10].

In addition to the layout of UI elements, there is ongoing research into various interaction methods for XR environments, such as hand tracking, gaze selection, and pointer tracking [13], [14]. These methods aim to provide users with intuitive ways to navigate and control XR experiences. Furthermore, text input options can differ significantly within XR, with methods such as point-and-click or pressing virtual buttons using a controller [15], [16]. However, it is essential to note that the exploration of



(a) The Beat Saber menu UI [11]. **(b)** Oculus Dash console [12].

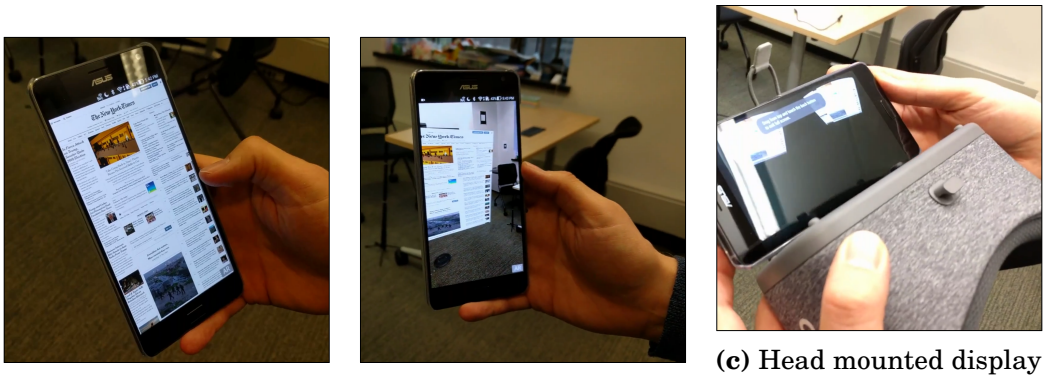
Figure 2.1. Figures showcasing common XR design trends.

these interaction methods is outside the scope of this thesis and will not be discussed in detail.

2.5 Related Work

The rapidly expanding field of extended reality (XR) has sparked numerous research efforts; however, relatively little work has been dedicated to adapting web content for XR browsing experiences.

One notable study conducted in 2018 by Michael Nebeling at the University of Michigan in collaboration with Mozilla explores reimagining web browsing in XR contexts [17], [18]. The proposed XRBrowser investigates transforming 2D web content into 3D virtual or augmented environments, as illustrated in Figure 2.2. Despite its innovative approach, the XRBrowser resembles the current state of XR browsing, which emulates desktop experiences within XR environments, and thus differs from the approach outlined in this thesis.

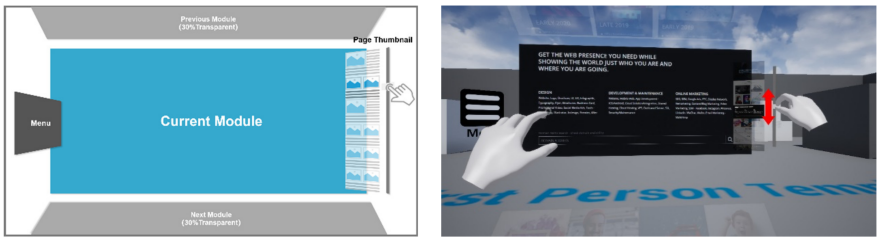


(a) Standard mobile view. **(b)** AR browsing view. **(c)** Head mounted display browsing view.

Figure 2.2. Different view options from Nebeling's XRBrowser demo video [19].

In a recent 2022 paper, Xing et al. propose a novel method for displaying and interacting with web content in XR environments [10]. The authors contend that traditional approaches, which often simulate a single device

screen, fail to fully exploit the unique interaction possibilities offered by XR technologies. To address this limitation, they introduce a set of innovative interaction methods, depicted in Figure 2.3, that leverage panoramic vision to enhance the browsing experience for the user [10]. Their user study revealed improved user performance in answering questions related to website content when using these additional external UI elements [10].



(a) (b)

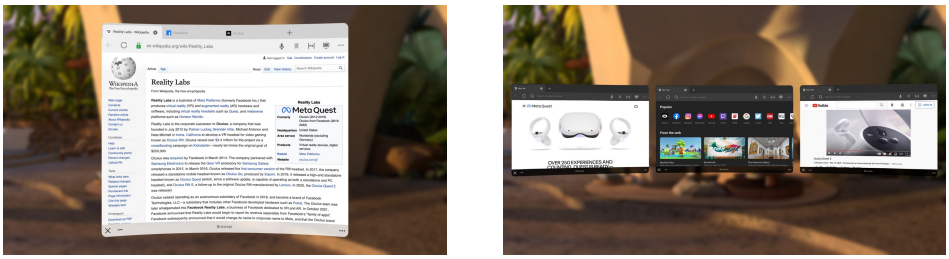
Figure 2.3. Images showing the design (a) and demo (b) of the novel UI proposed by Xing et al. [10].

The findings of the Xing et al. study demonstrate the potential for significant improvements in XR web browsing experiences through the adaptation of web content. While their proposed method focuses on the XR application development side, this thesis presents an approach that enables web designers to tailor their designs to maximize the benefits of the XR interaction space.

3. Extended Reality Layout Framework

The following sections will discuss the proposed extended reality layout (XRL) framework. Section 3.1 clearly defines the design problem addressed by the proposed framework. Section 3.2 presents the key design objectives for the framework. Finally, section 3.3 introduces the framework, discuss how these objectives were achieved by the framework, while section 3.4 briefly explore the technical implementation.

3.1 Design Problem



(a) Single window with multiple tabs. **(b)** three-window view.

Figure 3.1. Figures showing views of the Meta Quest Browser [20].

The current approach to web browsing in XR primarily involves replicating the non-XR browsing experience within an XR environment, often appearing as a single floating virtual window displaying a web page much like on a large computer monitor as seen in figure 3.1. This approach is noticeably in contrast to the previously discussed XR design trends as it overlooks the unique advantages of XR, such as infinite virtual screen space and panoramic vision. Although it is possible to develop custom web-based XR experiences, the high costs and complexity associated with designing these experiences render them impractical for most web services. Moreover, the specialized knowledge needed for XR design generally exceeds the expertise of existing web design teams. As a result, the current state of XR

browsing remains less engaging and accessible, potentially deterring users from embracing it as their preferred browsing method.

Considering the unrealized potential of XR browsing, a design problem emerges – web developers need a framework that enables them to adapt their web content for XR contexts, allowing for more intuitive and user-friendly layouts that effectively utilize the capabilities of XR environments.

3.2 Design Objectives

Design objectives provide direction to the development of the design framework and serve as evaluation criteria to assess its success in tackling the design problem. The following design objectives have been identified for the development of the XRL framework to ensure its effectiveness and alignment with user needs.

1. The XRL framework must cause minimal disruption to the web UI design process;

By minimizing disruption to the web UI design process, the XRL framework ensures that developers can adopt the framework with ease, without requiring a significant overhaul of their existing design practices. Moreover, by keeping the framework simple to learn and implement, it establishes itself as a viable and accessible alternative to designing a custom 3D experience from scratch, which is often more complex and proprietary. This objective serves to reduce any resistance to the adoption of the XRL framework by facilitating a smooth integration into the developer's workflow and extending XR support to content that might not initially receive specific attention in XR environments. The resulting ease of use enables more natural and intuitive experiences in XR with minimal overhead.

2. The XRL framework must support sufficient design freedom;

This objective emphasizes the importance of allowing designers to maintain the unique look and feel of their websites when adapting them from a traditional 2D context to an XR environment. It ensures that the XRL framework provides enough flexibility for designers to create layouts that reflect the structure of their original site, regardless of its complexity. Rather than acheive an exact replication of 2D web content layouts in XR, the XRL framework aims to facilitate an intuitive and recognizable reorganization of content, allowing users familiar with the original site

to easily locate and navigate content in an XR layout. This is achieved by allowing the creation of dynamically defined layouts that align with the designers' vision and the essence of the original site.

3. The XRL framework must enhance XR browsing capabilities;

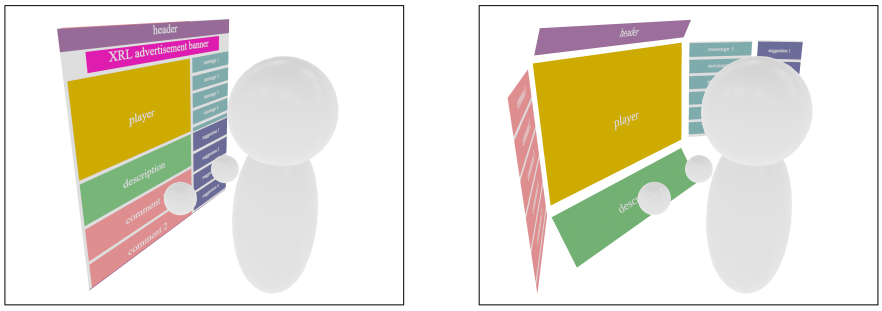
As part of facilitating a smooth transition from traditional to XR web contexts, the XRL framework should enable designers to better take advantage of the capabilities afforded by XR contexts compared to current XR browsing methods. This objective focuses on leveraging the unique advantages of XR environments, such as infinite virtual screen space and panoramic vision, to provide a more immersive, user-friendly browsing experience for end users engaging with content in XR contexts.

4. The XRL framework must ensure cross-platform and legacy compatibility.

Ensuring cross-platform and legacy compatibility guarantees that websites utilizing the XRL framework will render correctly in conventional web browsers, such as desktop and mobile web browsers, as well as in XR web browsers. To achieve this, the framework must be adaptable to various levels of implementation, allowing content with no, partial, or full XRL framework support to be viewed appropriately in supported XR browsers while maintaining an undisturbed experience on traditional web browsers.

The selection of these design objectives aims to facilitate an easy and intuitive transition from traditional to XR web contexts for both web designers and end users.

3.3 Design



(a) Web page without XRL support. **(b)** Web page with XRL support.

Figure 3.2. Mock-up of proprietary XR browsing application for sites with various levels of XRL Framework support

The proposed extended reality layout (XRL) framework is designed around an adaptable component structure, built upon the foundation of frame elements and frame paths. Frame elements serve as virtual screens containing web content, which can be positioned and organized within the framework. Frame paths control the flow and distribution of virtual frames by placing them in designated directions – either in single-file columns or rows – ensuring a well-organized and structured layout.

At the center of the framework lies the main virtual frame, which is surrounded by four 'frame paths' named head, left, right, and below. Each path plays a distinctive role in organizing content within the layout intuitively. When elements should not be displayed during XR browsing, but are necessary for desktop or mobile browsing, they can be placed in the 'ignore' frame which remains functionally loaded, allowing for related scripts and functionality to run without issue, but hidden from users. The table below outlines the components of this framework and their respective functions.

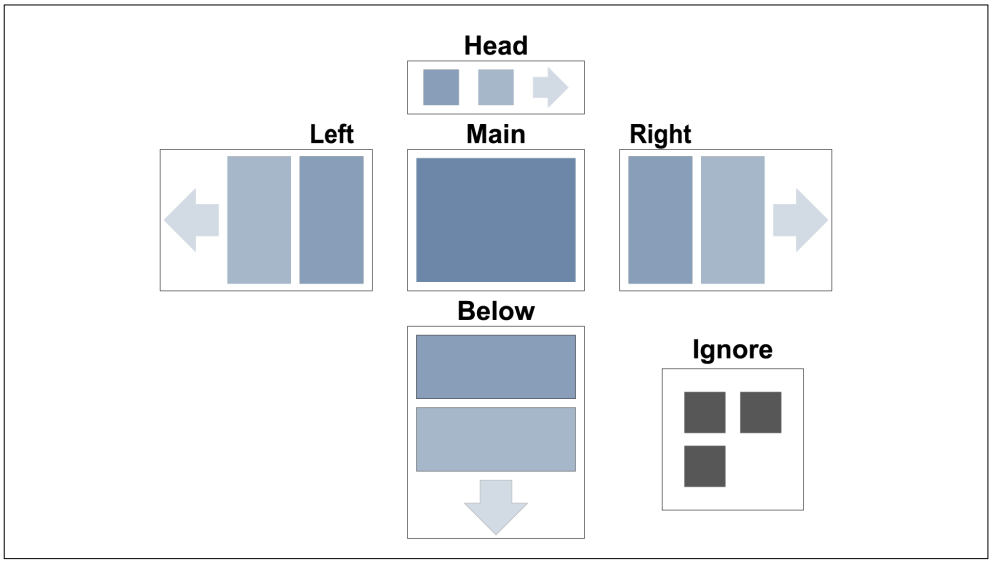


Figure 3.3. Diagram showing the different content locations in the XRL framework.

XRL Location	Description
Main Frame	Central virtual frame, providing a reference for other frame paths.
Head Path	Lateral placement of elements (left to right) above the main frame.
Left Path	Stacked elements extending from the main frame (right-to-left).
Right Path	Stacked elements extending from the main frame (left-to-right).
Below Path	Vertical placement of elements (top to bottom) below the main frame. Elements which do not have a specified XRL location will appear at the bottom of this path.
Ignore	Hidden group for content that should not be displayed in XR contexts.

Table 3.1. XR Layout Framework Elements

One notable advantage of the XRL framework is that it contributes additional functionality, without impeding the browsing experience for sites that have not yet implemented the framework. When a site has not adopted XRL, or if any elements do not specify a XRL framework location, this content is automatically loaded into the first element of the "below" path, resulting in a browsing experience very similar to a desktop experience. The XRL framework is designed with compatibility and accessibility in mind, ensuring a seamless and immersive user experience for all.

3.4 Technical Implementation

In order to achieve this functionality, the XRL framework can be implemented similarly to other popular web frameworks such as Bootstrap, a common web UI framework [21]. In Bootstrap, designers control the layout of their web elements through the use of pre-built components [21]. These components commonly take the form of HTML class attributes which de-

fine the position and behavior of the element to which they are applied. Similarly, the XRL framework implementation is based on a class-attribute system, controlling the location of a virtual frame according to the following rules:

1. Items containing one of the XRL framework classes will be placed into their respective frame.

2. In the case that multiple items are assigned the same XRL framework class, they will be arranged along the direction of the frame path in the order they appear. For example, to ensure that an element is placed directly to the right of the main frame as the first element in the XRL-right path, the element must be the first element in the HTML code that is assigned the XRL-right class.

3. Any elements that are not assigned an XRL framework class, and are not nested inside an XRL framework element, will be added to the end of the XRL-below path after all the other XRL framework content has been processed. In the case of legacy content, where the XRL framework is not implemented, the entire page will be loaded into the last frame of the XRL-below path, which is the only frame available in this case. In this way, legacy content can be viewed in a way similar to current XR browsing methods.

3.5 Framework Assessment

To assess the degree to which the XRL framework meets the design objectives, a proof-of-concept implementation was developed. Initially, a Unity-based implementation was planned; however, due to time and complexity limitations, a Python Flask server using the Beautiful Soup module was created. The Flask server parses content and displays it in a web page format which emulates the layout of the XRL framework. Furthermore, in order to demonstrate the potential utility of the framework in various scenarios, several website mockups were developed based on real-world archetypes. These mockups, along with a discussion on the extent to which the design objectives were achieved, are examined in chapter 4. Additionally, the source code for both the Flask server and site mockups is available in the appendix in sections A.2 and A.1.

4. Case Studies

To further evaluate the effectiveness of the XRL framework in meeting the design objectives, a series of case studies were conducted. These studies focused on the development of mock layouts for various common website types, incorporating the XRL framework. Each layout features color-coded elements for easy identification in both traditional and XRL framework views. The subsequent sections will delve into the rationale behind each site selection, present an overview of the respective site code, showcase rendered browser screenshots of the layouts, and discuss how the framework addresses the design objectives in each case, along with any additional benefits for users provided by the XRL framework.

The subsequent sections will focus on distinct types of websites to demonstrate the versatility of the XRL framework. Section 4.1 discusses an information site mock layout. Section 4.2 discusses a search engine mock layout. Section 4.3 discusses a media site mock layout. Finally, section 4.4 discusses a mail client mock layout. The source code of each site can be seen in section A.1.

4.1 Info

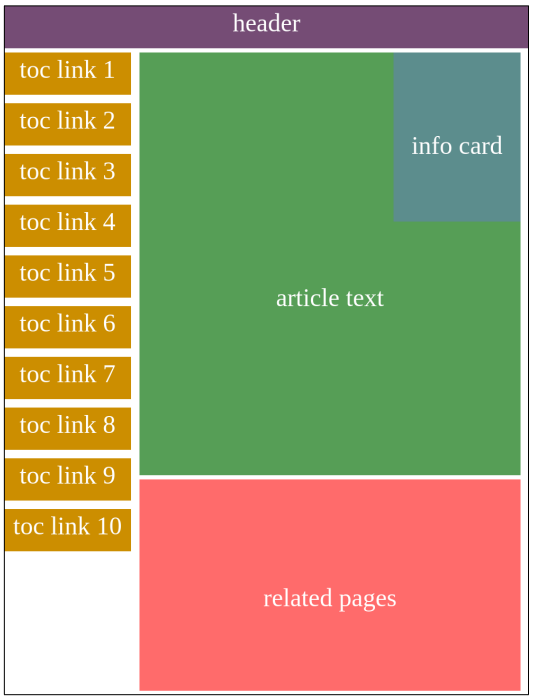


Figure 4.1. Info site mock layout.

The Info site mock layout, designed based on a generalized Wikipedia article, is representative of the static site archetype; a common type of website that can be found in various web content such as online news articles, blog posts, online recipes, and more. Figure 4.1 shows the mock layout as it would appear in a traditional desktop browser.

In the Info site mock layout, several key elements have been adapted using the XRL framework to enhance user experience and accessibility. The green article text, which serves as the main content, was chosen as the XRL-main element to ensure its prominence. The gold table of contents and the teal article info card are placed in the XRL-left and XRL-right frame paths to allow users quick access to the table of contents and important article information without disrupting the main content flow. The maroon header, which normally consists of menu and search actions, is positioned in the XRL-head frame path, maintaining its visibility for easy navigation. Finally, the salmon-colored related pages section is allocated to the XRL-below frame path to give users a focused view of relevant information. This arrangement of content elements not only ensures that they are more accessible but also provides a logical organization that allows users to interact with the various components without the need for excessive scrolling, enhancing the overall user experience within the XR

environment. The XRL framework layout of the info mock site can be seen in figure 4.2.

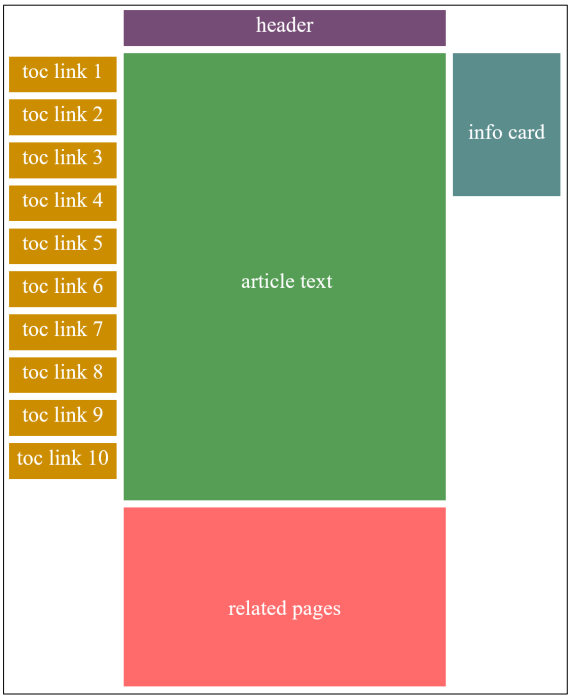
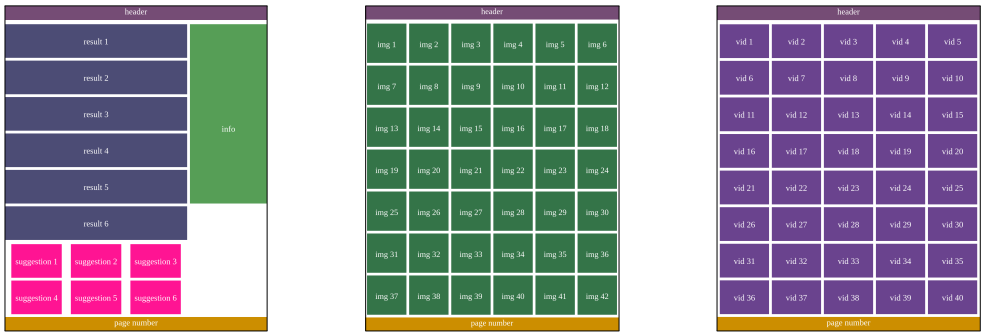


Figure 4.2. Info site mock layout when viewed in XR.

The implementation of the XRL framework significantly improves the navigability of the content within the Info site case study. By separating the content into individually scrollable frames, users can effortlessly navigate around the main content while maintaining access to additional content. As shown in Figure 4.2, this enables smooth navigation within the green article text element, without compromising the visibility of supplementary components like the teal info card or the gold table of contents elements. This streamlined organization allows users to quickly move between different elements of the article without needing to scroll to the top of the page, enhancing the overall user experience within the XR environment.

4.2 Search



(a) Results page. **(b)** Images page. **(c)** Videos page.

Figure 4.3. Search site mock layout.

The Search site mock layout, inspired by a tabbed search engine results page, represents the tabbed multi-page site archetype; another prevalent type of website found in various web content, including business sites with multiple tabs like About, Projects, Contact Us, etc. In this case, the three pages represent a link results page shown in figure 4.3a, an image results page shown in figure 4.3b, and a video results page shown in figure 4.3c.

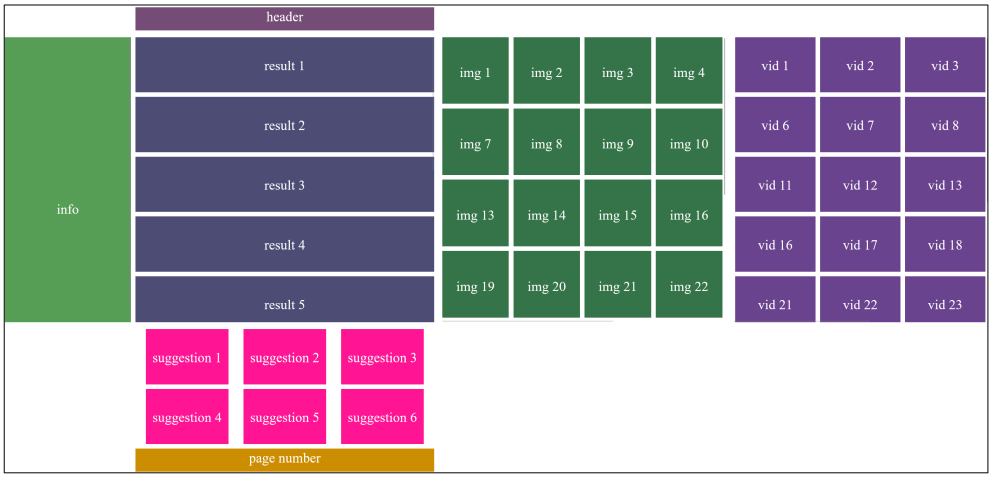


Figure 4.4. Search site mock layout when viewed in XR.

The XRL framework layout for the search site can be seen in figure 4.4. This layout improves the organization of the content and enhances user experience. The purple link results section was selected as the XRL-main element. The green info card, when present in the search results, appears in the XRL-left frame path. The XRL-right frame path consists of the dark-green image and lavender-colored video result pages, while the maroon header, containing for example a search bar and further controls, occupies the XRL-head frame path. Lastly, the related searches section and page navigation control are placed in the XRL-below frame path.

Prior to the adoption of the XRL framework, users had to scrolling back to the top of the page to modify their search query and navigate multiple sub-pages to access image and video results. The XRL framework implementation, shown in figure 4.4, streamlines the user experience by taking advantage of the virtually infinite screen space provided by XR environments to place the content in multiple virtual frames. This makes content easily accessible during browsing, as users can navigate the link, image, and video results seamlessly. This efficient design eliminates unwanted white space, improves content visibility, and contributes to a more dynamic and immersive user experience in the XR environment.

4.3 Media

Figure 4.5. Media site mock layout.

The media client mock layout, designed after common layouts for video content services, is representative of the multimedia player sites archetype found on platforms like YouTube, Vimeo, and Twitch. Figure 4.5 shows the

mock layout as it would appear in a traditional desktop browser.

In the media client mock layout, various key elements were adapted using the XRL framework to enhance user experience and accessibility. The maroon header, containing menu and search actions, is positioned in the XRL-head frame path, while the gold player element is chosen as the XRL-main component. This allows users to watch the video content without losing sight of it while interacting with other components. The green description element is placed in the XRL-below frame path for easy access, and the salmon-colored comment elements are located in the XRL-left frame path. The purple recommended video elements, providing related content suggestions, are positioned in the XRL-right frame path.

Figure 4.6. Media site mock layout with live chat.

An additional live chat component (Figure 4.6, XRL-right) is available on certain videos when applicable. When present, it is placed between the player element and the recommended video elements on the XRL-right frame path; however, no gap is left when the live chat is hidden, showcasing the flexibility and adaptability of the XRL framework.

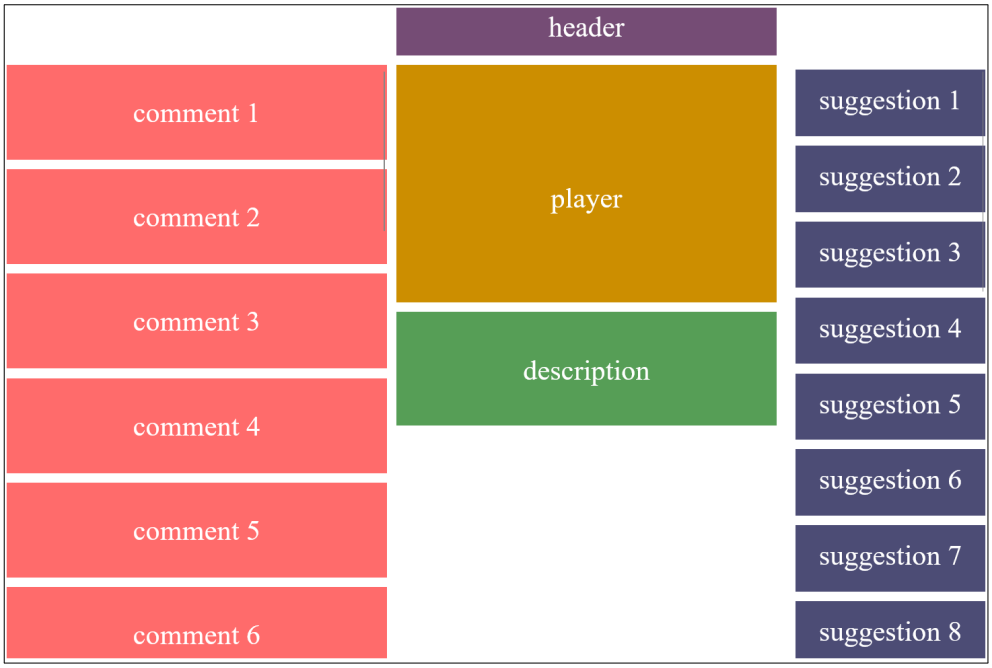


Figure 4.7. Media site mock layout when viewed in XR.

The implementation of the XRL framework, as seen in 4.7 significantly adds value to the user experience and navigability within the media client case study. By having individual scrollable frames, users can conveniently read video descriptions, browse comments, and explore recommended video suggestions while keeping the video in view.

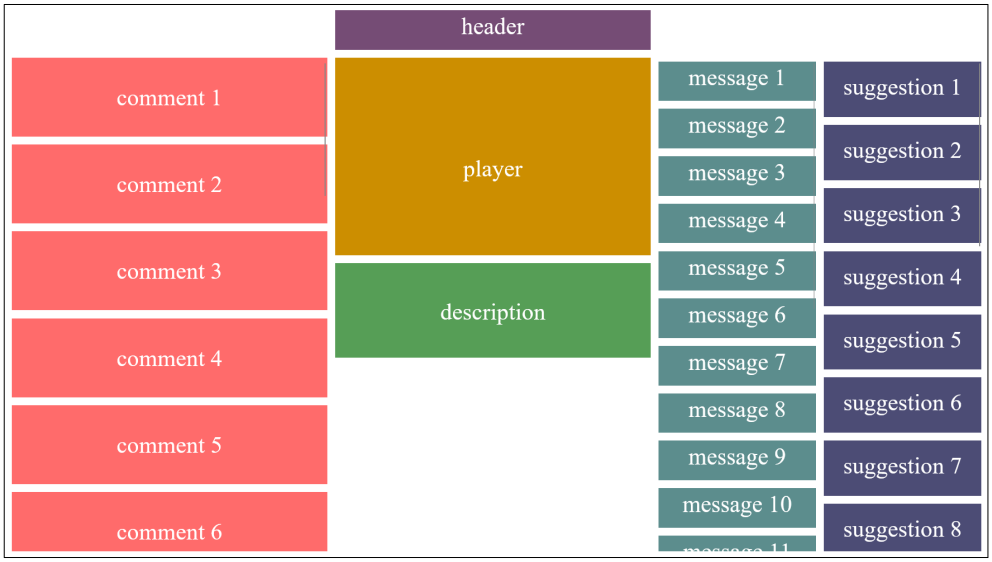


Figure 4.8. Media site mock layout with live chat when viewed in XR.

In the case that the live chat element is present for a given video, the XRL framework dynamically expands, creating a new frame for this element at the beginning of the XRL-right frame path. The separate frame facilitates smooth interaction with the live chat feature, creating a cohesive browsing experience. Overall, these adjustments of content elements within the

XRL framework make the content more accessible for the user within the XR environment.

A noticeable distinction between the XRL framework layout and the conventional layout of this simulated mail site, is the exclusion of the bright pink XRL advertisement banner. While this banner is visible to desktop and mobile site users, it is not necessary to display it to XR users who are already experiencing the XR optimized layout. Therefore, the XRL-ignore tag is assigned to this element, and it remains hidden from XR users.

4.4 Mail

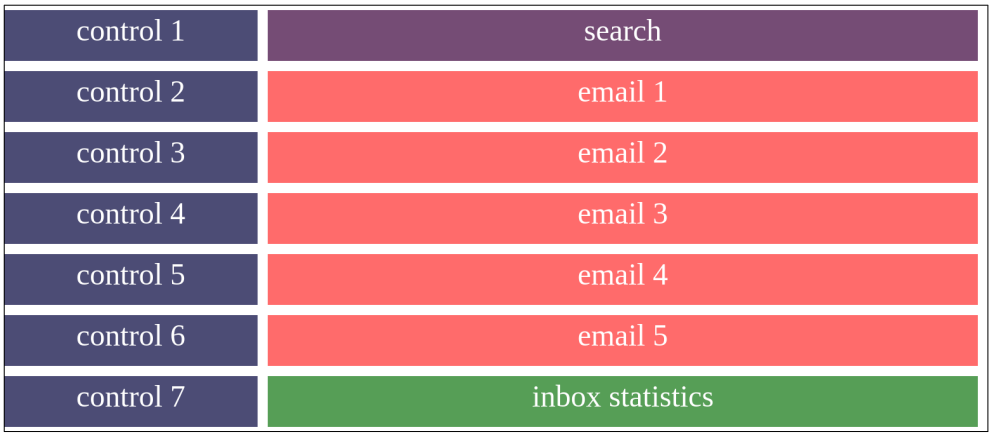


Figure 4.9. Mail site mock layout.

The Media site mock layout, designed based on a typical email client, is representative of daily workflow and communication tools. It can be found in various web applications, like mail clients, project management systems, and social media platforms. Figure 4.5 shows the mock layout as it would appear in a conventional desktop browser.

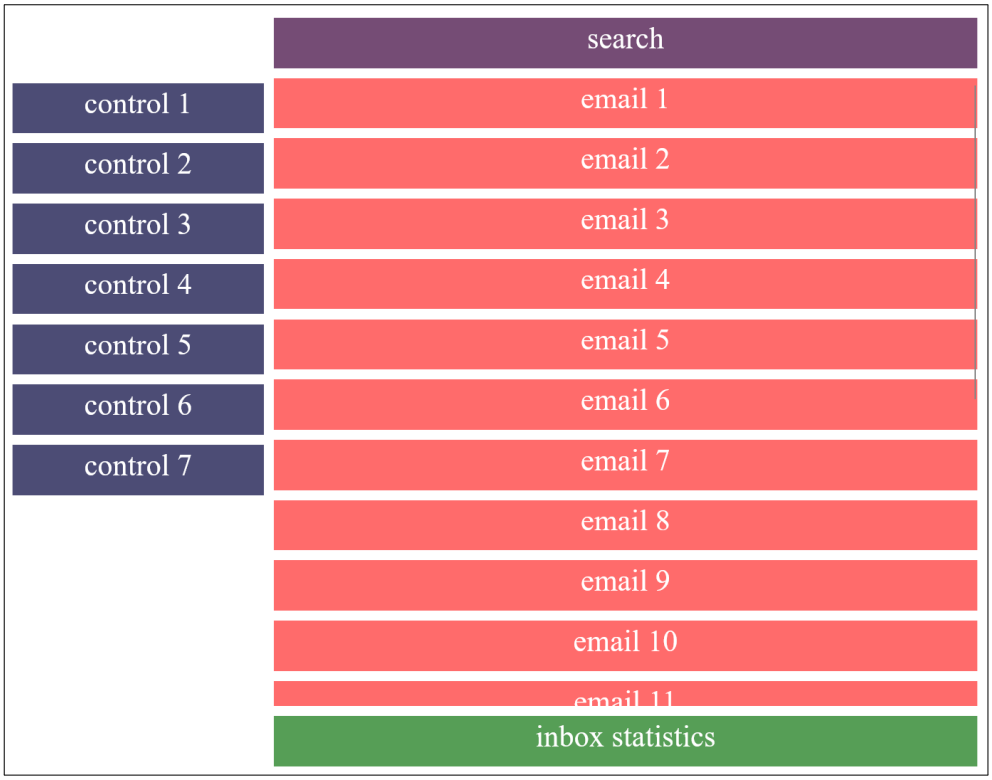


Figure 4.10. Mail site mock layout when viewed in XR.

In the Media site mock layout, the XRL framework has been employed to maximize user experience and accessibility, seen in figure 4.10. The salmon-colored inbox, serving as the main content, is designated as the XRL-main element. The purple navigation controls, including inbox and trash, are positioned within the XRL-left frame path to provide users with easy access to primary controls. The maroon header is assigned to the XRL-head frame path, ensuring its continuous visibility for enhanced navigation. The green footer with inbox statistics is allocated to the XRL-below frame path, keeping users informed about their inbox details.

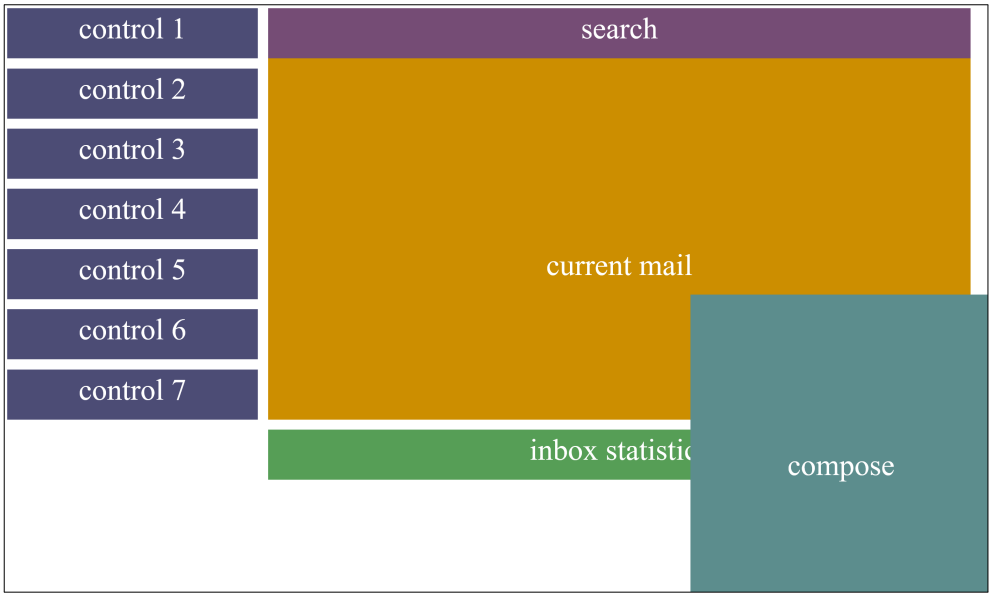


Figure 4.11. Mail site mock layout with compose email and current mail actions.

In the desktop view, additional actions are available. When engaging with individual emails, a gold-colored current mail view is introduced to display the selected message chain. Similarly, when composing an email, a teal-colored compose window appears at the bottom right of the screen, providing users with the necessary tools for composing messages. These actions can be seen in figure 4.11.

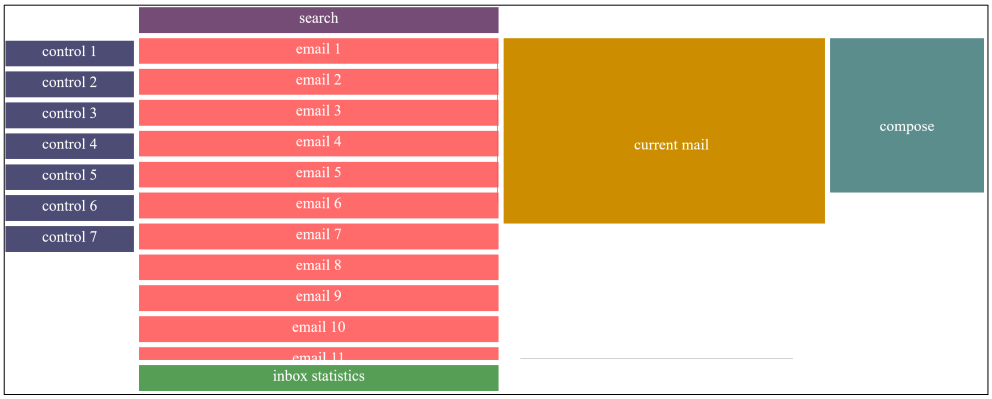


Figure 4.12. Mail site mock layout with compose email and current mail actions when viewed in XR.

These actions are similarly available in the XRL framework layout. The gold-colored current mail view is introduced within the XRL-right frame path, and teal-colored compose window is placed in the XRL-right frame path. The XRL framework layout of the media mock site with active actions can be seen in figure 4.8.

The incorporation of the XRL framework significantly enhances accessibility and interaction within the Media site case study. By separating content into individually scrollable frames, users can easily navigate

through primary components like the inbox, while gaining quick access to secondary elements such as the current mail view or compose window. As illustrated in Figure 4.8, this enables seamless navigation within the salmon-colored inbox element without compromising the visibility or functionality of the gold current mail view or teal compose window. The ability of the XRL framework to show or hide elements based on contextual needs ensures that users are not overwhelmed with information, while simultaneously presenting essential content with clear visibility. This organized and adaptable approach enriches the overall user experience within the XR environment.

5. Conclusion

In conclusion, this thesis effectively demonstrates how web content can take advantage of the unique capabilities of XR through the proposed extended reality layout (XRL) framework — a versatile and practical solution to web developers aiming to adapt their web content to extended reality environments. By utilizing a class-attribute based system, the XRL framework allows for seamless implementation into existing web UI designs, minimizing disruption and simplifying the process for designers. This encourages a more accessible and streamlined integration of XR support into diverse types of websites, as demonstrated in the various case studies such as Info, Search, Media, and Mail sites. With its focus on design freedom, the XRL framework ensures that designers retain their creative autonomy. This allows designers to preserve the essence of their original layout and design, while enhancing XR browsing capabilities offered by unique features such as infinite virtual screen spaces and panoramic vision. Moreover, the framework's focus on cross-platform and legacy compatibility guarantees a seamless transition from conventional browsing to XR, ensuring that users can experience an immersive and intuitive web browsing experience.

The four case studies showcased the adaptability and functionality of the XRL framework in addressing distinct design problems, catering to the needs of various types of web content. From simplifying navigation in Info and Search sites to improving accessibility of multimedia content in the Media site and optimizing complex multi-step tools in the Mail site, the XRL framework has proven its potential to transform the XR browsing experience into a more engaging and user-friendly platform. By providing a unified framework that caters to the unique challenges of extended reality layouts, the XRL effectively bridges the gap between traditional and XR web browsing, opening doors for a realm of possibilities in the future of

web design and extended reality experiences.

5.1 Topics for Further Research

Throughout the research and development process of the extended reality layout (XRL) framework, a number of compelling topics for future research emerged. This section will introduce two key areas that warrant further exploration: developing a new approach for content sizing in XR environments through cross-reality dimensions, and fostering mass adoption of the XRL framework by integrating it with popular website builder frameworks.

5.1.1 Cross-Reality Dimensions

In traditional web design, dimensions are typically based on the proportion of the device's physical size or the screen resolution. However, these methods are not suitable for XR environments. As such, there is a need to develop a new approach for content sizing in XR platforms. One potential solution is to emulate the 'screen size' to a standard 4k resolution when loading web content, then computing the size of the frames in-environment by assigning a relationship between pixel and world dimensions (e.g., 1 pixel = 1 millimeter). This would enable more robust and consistent sizing of content across various XR environments.

The implementation of such a method would open the possibility of using tags like XRL-pos[x/y]-[number] and XRL-dim[x/y]-[number] to determine the positioning and dimensions of XRL frames based on this new unitary standard. These tags could be easily incorporated into existing content and allow for more precise control of the XRL framework layout.

5.1.2 Mass Adoption

Regardless of the benefits provided by the XRL framework, the XR browsing experience will remain unchanged until the framework reaches mass adoption. One way to facilitate such an adoption is through the development of a WebXR application which renders a given page using the XRL framework. By open sourcing such an application, developers could include it within their websites allowing XR users to enter the custom WebXR experience from within their browser.

Further adoption could be encouraged by integrating this feature into popular site builder frameworks, such as Wix or WordPress. Standardiza-

tion of components within these frameworks, such as header and menu elements, would allow for seamless integration with the framework through this WebXR-based browsing experience. This, in turn, could serve as a catalyst for popularizing the XRL format and kick-starting mass adoption. The ubiquity of these site builder frameworks, coupled with the growing interest in XR technologies, presents an opportune moment for integration of the XRL framework into the broader digital landscape.

In conclusion, further research into cross-reality dimensions and mass adoption strategies for the XRL framework holds significant potential for shaping the future of web content and extended reality experiences. By overcoming the challenges inherent in these areas, the stage will be set for the widespread proliferation of XR-based web content and the growth of the extended reality ecosystem.

References

[1] A. Hussain and E. O. C. Mkpojiogu, "The effect of responsive web design on the user experience with laptop and smartphone devices," 2015.

[2] C. Mullins, "Responsive, mobile app, mobile first: Untangling the ux design web in practical experience," *Proceedings of the 33rd Annual International Conference on the Design of Communication*, 2015.

[3] W. I. Bader and A. I. Hammouri, "Responsive web design techniques," *International Journal of Computer Applications*, vol. 150, pp. 18–27, 2016.

[4] J. C. Joibi, Y. M. Kim, G. W. Shin, K.-J. Lee, and M. H. Yun, "1h3-3 a systematic review of extended reality (xr)'s head-mounted display (hmd) hardware design factors from the perspective of usability," *The Japanese Journal of Ergonomics*, 2019.

[5] S. Mann, T. Furness, Y. Yuan, J. Iorio, and Z. Wang, "All reality: Virtual, augmented, mixed (x), mediated (x, y), and multimediated reality," *ArXiv*, vol. abs/1804.08386, 2018.

[6] X. Qiao, P. Ren, S. Dustdar, L. Liu, H. Ma, and J. Chen, "Web ar: A promising future for mobile augmented reality—state of the art, challenges, and insights," *Proceedings of the IEEE*, vol. 107, pp. 651–666, 2019.

[7] B. Maclntyre and T. F. Smith, "Thoughts on the future of webxr and the immersive web," *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 338–342, 2018.

[8] Vtieto, *Using webxr with windows mixed reality - mixed reality*, `https://learn.microsoft.com/en-us/windows/mixed-reality/develop/javascript/webxr-overview`, Accessed: Apr. 28, 2023, Jul. 2022.

[9] T. Savage, "Componentizing the web," *Communications of the ACM*, vol. 58, pp. 55–61, 2015.

[10] Y. Xing, J. Shell, C. Fahy, T. Xie, H.-Y. Kwan, and W. Xie, "Web xr user interface research: Design 3d layout framework in static websites," *Applied Sciences*, 2022. [Online]. Available: https://doi.org/10.3390/app12115600.

[11] Steam Community, *Campaign, ost vol. 2 & the 1st music pack release info*, https://store.steampowered.com/news/app/620980/view/4036870438631349382, [Online]. Accessed: Apr. 28, 2023.

[12] Oculus Developers, *Oculus dash*, https://developer.oculus.com/documentation/native/pc/dg-dash/, [Online]. Accessed: Apr. 28, 2023.

[13] F. Zhang, V. Bazarevsky, A. Vakunov, *et al.*, "Mediapipe hands: On-device real-time hand tracking," *ArXiv*, vol. abs/2006.10214, 2020.

[14] F. L. Luro and V. Sundstedt, "A comparative study of eye tracking and hand controller for aiming tasks in virtual reality," *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, 2019.

[15] T. J. Dube and A. S. Arif, "Text entry in virtual reality: A comprehensive review of the literature," in *Interacción*, 2019.

[16] M. Speicher, A. M. Feit, P. Ziegler, and A. Krüger, "Selection-based text entry in virtual reality," in *CHI '18 Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ACM, 2018, p. 647. DOI: 10.1145/3173574.3174221.

[17] M. Nebeling, "Xrbrowser: Rethinking the web browsing experience for ar/vr," 2018.

[18] J. Webster, *Mozilla grant helps fund reimagining augmented reality browsers*, https://www.si.umich.edu/about-umsi/news/mozilla-grant-helps-fund-reimagining-augmented-reality-browsers, Accessed: Apr. 28, 2023, Feb. 2019.

[19] Mn, *XRBrowser with New York Times*, https://vimeo.com/242573100, [Online]. Accessed: Apr. 28, 2023, Mar. 2023.

[20] Oculus, *Meta quest browser on oculus quest 2*, https://www.oculus.com/experiences/quest/1916519981771802/, Accessed: Apr. 28, 2023, 2019.

[21] M. Otto, J. Thornton, and B. contributors, *Bootstrap · the most popular html, css, and js library in the world.* Dec. 2022. [Online]. Available: https://getbootstrap.com/.

A. Appendix

A.1 Mock Sites

The following subsections contain the source code of the various site mock-ups used in the case studies chapter of this thesis, chapter 4. When trying the sites locally, the number keys (1, 2, 3, ...) can be used to trigger any additional site functionality.

A.1.1 Shared Styling

```
html {
  background-color: white;
  display: flex;
  justify-content: center;
}

body {
    margin: 0;
}

.inner {
  margin: 0.5vw 1vw;
}

.innerCol {
  display: flex;
  flex-direction: column;
  padding-top: 0.5vw;
  gap: 1vw;
}

*::-webkit-scrollbar {
  display: none;
```

```
}

/* Hide scrollbar for IE, Edge and Firefox */
* {
  -ms-overflow-style: none;  /* IE and Edge */
  scrollbar-width: none;  /* Firefox */
}
```

A.1.2 Shared Script

```
colors = [
  "#754C75", // Purple
  "#CC8E00", // Orange
  "#569E56", // Lawn Green
  "#5C8D8D", // Cyan
  "#FF6B6B", // Deep Pink
  "#4C4C75", // Navy
  "#FF1493", // Deep Pink
  "#347448", // Olive
  "#6A438E", // Dark Violet
  "#4C7575", // Teal
  "#4F96CD", // Light Blue
  "#CCA300", // Gold
];
function change_colors(array) {
  var r = document.querySelector(":root");
  for (i = 0; i < array.length; i++) {
    r.style.setProperty(`--color${i + 1}`, array[i % array.length]);
  }
}
function fill_list(name, count, content1, content2) {
  try {
    for (i = 0; i < count; i++) {
      document.getElementById(name).innerHTML += `${content1}${i + 1}${content2}`;
    }
  } catch {
    console.log(`${name} not found`);
  }
}
document.addEventListener("keypress", (e) => {
  try {
    keypress(e);
  } catch {
    console.log("no custom keypress function defined");
  }
});
window.onload = function () {
```

```
  try {
    add_elements();
  } catch {
    console.log("no elements to add");
  }
  document.querySelectorAll("*[alt]").forEach((element) => {
    element.style.backgroundImage = `url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg'%3E%3Ctext x='50%' y='60%' font-size='calc(min(10vw,4em))' text-anchor='middle' fill='%23fff'%3E${element.getAttribute(
      "alt"
    )}%3C/text%3E%3C/svg%3E")`;
  });
  change_colors(colors);
};

```

A.1.3 Info

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Layout Mockup: Info</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="content" style="display: flex; flex-direction: column;">
        <div alt="header" class="XRL-head " style="width: 62vw; height: 5vw; background-color: var(--color1);">
        </div>
        <div style="display: flex; column-gap: 1vw;">
            <div id="toc" class="XRL-left innerCol" style="width: 15vw;"></div>
            <div class="innerCol">
                <div alt="article text" class="XRL-main "
                    style="width: 45vw; height: 100vw; background-color: var(--color3); display: flex; flex-direction: row-reverse;">
                    <div alt="info card" class="XRL-right "
                        style="width: 25vw; height: 35vw; background-color: var(--color4);">
                    </div>
                </div>
                <div alt="related pages" class="XRL-below "
                    style="width: 45vw; height: 25vw; background-color: var(--color5);">
                </div>
            </div>
        </div>
```

```
26      </div>
27  </body>
28  <script>
29      function add_elements() {
30          fill_list('toc', 10, '<div alt="toc link ', '" class="" style="height: 5vw;
            ↪  background-color: var(--color2);"></div>');
31      }
32  </script>
33  <script src="thesis.js"></script>
34  </html>
```

A.1.4 Mail

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <meta charset="UTF-8">
5       <title>Layout Mockup: Mail</title>
6       <link rel="stylesheet" href="style.css">
7   </head>
8   <body>
9       <div id="content" style="display: flex; justify-content: space-between;">
10          <div id="leftMenu" class="XRL-left innerCol" style="width: 25vw;"></div>
11          <div style="display: flex; flex-direction: column;">
12              <div class="inner">
13                  <div alt="search" onclick="toggle_compose()" class="XRL-head "
14                      style="width: 70vw; height: 5vw; background-color: var(--color1);">
15                  </div>
16                  <div id="currentMail" alt="current mail" onclick="toggle_mail()"
                    ↪  class="XRL-right "
17                      style="display: none; width: 70vw; height: 35vw; margin-top: 1vw;
                        ↪  background-color: var(--color2);">
18                  </div>
19              </div>
20              <div id="inboxWrapper" class="inner" style="max-height: 35vw; overflow-y:
                ↪  auto;">
21                  <div id="inbox" class="XRL-main innerCol" style="width: 70vw; padding-top:
                    ↪  0;"></div>
22              </div>
23              <div style="position: absolute; bottom: 0; right: 0; pointer-events: none">
24                  <div id="compose" alt="compose" class="XRL-right "
25                      style="display: none; width: 40vw; height: 40vw; background-color:
                        ↪  var(--color4);">
26                  </div>
27              </div>
28              <div class="inner">
29                  <div alt="inbox statistics" class="XRL-below "
```

```
                    style="width: 70vw; height: 5vw; background-color: var(--color3);">
                </div>
            </div>
        </div>
    </div>
</body>
<script>
    function toggle_mail() {
        const toggle = document.getElementById('currentMail').style.display == "block";
        document.getElementById('currentMail').style.display = toggle ? "none" : "block";
        document.getElementById('inboxWrapper').style.display = toggle ? "block" : "none";
    }
    function toggle_compose() {
        let compose = document.getElementById('compose');
        if (compose.style.display != 'none') { compose.style.display = 'none'; display_chat = false; }
        else { compose.style.display = 'block'; display_chat = true; }
    }
    function add_elements() {
        fill_list('inbox', 20, '<div alt="email ', '" style="height: 5vw; background-color: var(--color5);" class="" onclick="toggle_mail()"></div>');
        fill_list('leftMenu', 7, '<div alt="control ', '" style="height: 5vw; background-color: var(--color6);" class=""></div>');
    }
    function keypress(e) {
        switch (e.key) {
            case '1':
                toggle_compose();
                break;
            case '2':
                toggle_mail();
                break;
            case '3':
                toggle_compose();
                toggle_mail();
            default:
                break;
        }
    }
</script>
<script src="thesis.js"></script>
</html>
```

A.1.5 Media

```
<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset="UTF-8">
    <title>Layout Mockup: Media</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="content" style="display: flex; flex-direction: column;">
        <div alt="header" onclick="toggle_chat()" class="XRL-head" style="width: 62vw; height: 5vw; background-color: var(--color1);">
        </div>
        <div alt="XRL advertisement banner" class="XRL-ignore" style="width: 50vw; height: 5vw; margin: 0.5vw auto; background-color: var(--color7);">
        </div>
        <div style="display: flex; justify-content: center;">
            <div style="display: flex; flex-direction: column;">
                <div class="inner">
                    <div alt="player" class="XRL-main "
                        style="width: 40vw; height: 25vw; background-color: var(--color2);"></div>
                </div>
                <div class="inner">
                    <div alt="description" class="XRL-below "
                        style="width: 40vw; height: 12vw; background-color: var(--color3);"></div>
                </div>
                <div class="inner">
                    <div id="comments" class="XRL-left innerCol" style="width: 40vw; padding-top: 0;"></div>
                </div>
            </div>
            <div style="display: flex; flex-direction: column;">
                <div style="max-height: 26vw; overflow-y: auto;">
                    <div id="chat" class="XRL-right innerCol" style="width: 20vw;"></div>
                </div>
                <div id="suggested" class="XRL-right innerCol" style="width: 20vw;"></div>
            </div>
        </div>
    </div>
</body>
<script>
    function toggle_chat() {
        const chat = document.getElementById('chat');
        if (chat.style.display != 'none') { chat.style.display = 'none'; }
        else { chat.style.display = 'flex'; }
    }
    function add_elements() {
```

```
        fill_list('chat', 20, '<div alt="message ', '" class="" style="height: 5vw; flex-shrink: 0; background-color: var(--color4)"></div>');
        fill_list('comments', 15, '<div alt="comment ', '" class="" style="height: 10vw; background-color: var(--color5)"></div>');
        fill_list('suggested', 15, '<div alt="suggestion ', '" class="" style="height: 7vw; background-color: var(--color6)"></div>');
    }
    function keypress(e) {
        if (e.key === '1') {
            toggle_chat()
        }
    }
</script>
<script src="thesis.js"></script>
</html>
```

A.1.6 Search

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Layout Mockup: Search</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="content" style="display: flex; flex-direction: column; row-gap: 1vw;">
        <div alt="header" onclick="{state = (state + 1) % 3; switch_page(state);}" class="XRL-head"
            style="width: 93.5vw; height: 5vw; background-color: var(--color1);">
        </div>
        <!-- Links page -->
        <div id="webResults" style="display: flex; justify-content: space-between; column-gap: 1vw;">
            <div style="width: 65vw; display: flex; flex-direction: column-reverse; row-gap: 1vw;">
                <div id="suggested" class="XRL-below innerCol"
                    style="flex-direction: row; flex-wrap: wrap; justify-content: space-evenly;">
                </div>
                <div id="results" class="XRL-main innerCol"></div>
            </div>
            <div alt="info" class="XRL-left "
                style="width: 27.5vw; height: 64vw; margin-top: 0.5vw; background-color: var(--color3);"></div>
        </div>
        <div id="imageResults" style="display: none; width: 93.5vw;">
```

```
25              <div id="imageWrapper" class="XRL-right innerCol"
26                  style="min-width: 50vw; justify-content: center; flex-direction: row;
                    ↪  flex-wrap: wrap;">
27              </div>
28          </div>
29          <div id="videoResults" style="display: none; width: 93.5vw;">
30              <div id="videoWrapper" class="XRL-right innerCol"
31                  style="min-width: 50vw; justify-content: center; flex-direction: row;
                    ↪  flex-wrap: wrap;">
32              </div>
33          </div>
34          <div alt="page number" class="XRL-below " style="height: 5vw; background-color:
            ↪  var(--color2);">
35          </div>
36      </div>
37  </body>
38  <script>
39      function add_elements() {
40          fill_list('results', 10, '<div alt="result ', '" class="" style="width: 65vw;
            ↪  height: 12vw; background-color: var(--color6);"></div>');
41          fill_list('suggested', 6, '<div alt="suggestion ', '" class="" style="width: 18vw;
            ↪  height: 12vw; background-color: var(--color7);"></div>');
42          fill_list('imageWrapper', 42, '<div alt="img ', '" class="" style="width: 14.75vw;
            ↪  height: 14.75vw; background-color: var(--color8);"></div>');
43          fill_list('videoWrapper', 35, '<div alt="vid ', '" class="" style="width: 17.9vw;
            ↪  height: 14.75vw; background-color: var(--color9);"></div>');
44      }
45      function switch_page(state) {
46          ['webResults', 'imageResults', 'videoResults'].forEach((page, index) => {
47              document.getElementById(page).style.display = index === state ? 'flex' :
                ↪  'none';
48          });
49      }
50      let state = 0;
51      function keypress(e) {
52          switch (e.key) {
53              case '1':
54                  state = (state + 1) % 3;
55                  switch_page(state);
56                  break;
57              case '2':
58                  state = 2;
59                  switch_page(2);
60                  break;
61              default:
62                  break;
63          }
```

```
    }
</script>
<script src="thesis.js"></script>
</html>
```

A.1.7 Store

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Layout Mockup: Store</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="content" style="display: flex; flex-direction: column; padding-bottom: 1vw;">
        <div alt="header" class="XRL-head " style="width: 93.5vw; height: 5vw;
        ↪  background-color: var(--color1);">
        </div>
        <div style=" display: flex; flex-direction: column;">
            <div style="display: flex; justify-content: space-evenly;">
                <div id="controls" class="XRL-left innerCol"></div>
                <div class="innerCol">
                    <div id="results" class="XRL-main innerCol" style="padding-top:
                    ↪  0;"></div>
                    <div style="width: 60vw; height: 8vw; overflow-x:scroll;">
                        <div id="related" class="XRL-right innerCol" style="flex-direction:
                        ↪  row;padding-top: 0;">
                        </div>
                    </div>
                    <div alt="suggestions" class="XRL-below " style="height: 5vw;
                    ↪  background-color: var(--color3);">
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
<script>
    function add_elements() {
        fill_list('controls', 10, '<div alt="control ', '" class="" style="width: 20vw;
        ↪  height: 5vw; background-color: var(--color4);"></div>');
        fill_list('related', 10, '<div alt="similar item ', '" class="" style="width: 20vw;
        ↪  height: 8vw; flex-shrink: 0; background-color: var(--color5);"></div>');
        fill_list('results', 6, '<div alt="result ', '" class="" style="width: 60vw;
        ↪  height: 10vw; background-color: var(--color6);"></div>');
    }
```

```
</script>
<script src="thesis.js"></script>
</html>
```

A.2 Flask Server

The flask server has the following file structure.

```
+-- app.py
+-- examples
|   +-- info.html
|   +-- mail.html
|   +-- media.html
|   +-- search.html
|   +-- store.html
|   +-- style.css
|   `-- thesis.js
`-- templates
    `-- xrl_framework_emulator.html
```

A.2.1 Flask App

```
# Standard library imports
import os

# Third-party library imports
from bs4 import BeautifulSoup as bs
from flask import (
    Flask,
    render_template_string,
    Response,
)

### APP SETUP ###

app = Flask(__name__)
app.secret_key = os.urandom(24)

### Routes ###

@app.route('/thesis/<path:name>')
def example(name):
    if name == "style.css":
        return Response(open('examples/style.css', 'r').read(), mimetype='text/css')
    if name == "thesis.js":
```

```
24          return Response(open('examples/thesis.js', 'r').read(), mimetype='text/javascript')
25
26      if name not in [os.path.splitext(file)[0] for file in os.listdir('examples') if
        ↪  file.endswith('.html')]:
27          return 'Site not found', 404
28
29      return Response(open(f'examples/{name}.html', 'r').read())
30
31  @app.route('/thesis/xrl/<path:name>')
32  def xrl_example(name):
33      if name == "style.css":
34          return Response(open('examples/style.css', 'r').read(), mimetype='text/css')
35      if name == "thesis.js":
36          return Response(open('examples/thesis.js', 'r').read(), mimetype='text/javascript')
37
38      if name not in [os.path.splitext(file)[0] for file in os.listdir('examples') if
        ↪  file.endswith('.html')]:
39          return 'Site not found', 404
40
41      xrl_main, xrl_head, xrl_below, xrl_right, xrl_left, head_list, footer_list, style_list,
        ↪  script_list, remainder = render_xrl(name)
42      # Render the template with the extracted elements
43      return render_template_string(open(f'templates/xrl_framework_emulator.html',
        ↪  'r').read(), xrl_head=xrl_head, xrl_main=xrl_main, xrl_left=xrl_left,
        ↪  xrl_right=xrl_right, xrl_below=xrl_below, head_list=head_list,
        ↪  footer_list=footer_list, style_list=style_list, script_list=script_list,
        ↪  remainder=remainder)
44
45  ### HELPER FUNCTIONS ###
46
47  def render_xrl(url):
48      with open(f'examples/{url}.html', 'r') as file:
49          html_content = bs(file.read(), "html.parser").prettify()
50
51      # parse the HTML content using Beautiful Soup
52      soup = bs(html_content, "html.parser")
53
54      # Create empty lists for each category of elements
55      head_list = []
56      footer_list = []
57      script_list = []
58      style_list = []
59
60      # Find all head elements and add them to the head_list
61      for element in soup.find_all('head'):
62          head_list.append(element.extract())
63
```

```
    # Find all footer elements and add them to the footer_list
    for element in soup.find_all('footer'):
        footer_list.append(element.extract())

    # Find all script elements that are not in the head or footer, and add them to the
    ↪ script_list
    for element in soup.find_all('script'):
        if element.parent.name in ["head", "footer"]:
            script_list.append(element.extract())

    # Find all style elements that are not in the head or footer, and add them to the
    ↪ style_list
    for element in soup.find_all('link'):
        if element.has_attr('rel') and 'stylesheet' in element['rel']:
            if element.parent.name in ["head", "footer"]:
                style_list.append(element.extract())
        else:
            head_list.append(element.extract())

    xrl_head = [elem.extract() for elem in soup.find_all(class_='XRL-head')]
    xrl_left = [elem.extract() for elem in soup.find_all(class_='XRL-left')]
    xrl_right = [elem.extract() for elem in soup.find_all(class_='XRL-right')]
    xrl_below = [elem.extract() for elem in soup.find_all(class_='XRL-below')]
    xrl_ignore = [elem.extract()
                  for elem in soup.find_all(class_='XRL-ignore')]

    main = [elem.extract() for elem in soup.find_all(class_='XRL-main')]
    xrl_main = main[:1]
    xrl_below = main[1:] + xrl_below

    return xrl_main, xrl_head, xrl_below, xrl_right, xrl_left, head_list, footer_list,
    ↪ style_list, script_list, soup

### MAIN ###

if __name__ == "__main__":
    app.run()
```

A.2.2 XRL Framework Emulator

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="icon" href="data:,">
    <title>XR Demo</title>
    {% for elem in head_list %}
```

```
8           <!-- Head -->
9           {{ elem.prettify()|safe }}
10      {% endfor %}
11      {% for elem in style_list %}
12          <!-- Style -->
13          {{ elem.prettify()|safe }}
14      {% endfor %}
15  </head>
16  <body style="overflow-x: scroll; margin: 1rem; position: absolute; left: 0;">
17      <div style="display: flex; flex-direction: column; row-gap: 1rem;">
18          <div style="display: flex; column-gap: 1rem;">
19              {% for elem in xrl_head %}
20              <div id="XRL-demo-head" style="display: flex; column-gap: 1rem; max-width:
                ↪  1500px; max-height: 1000px; overflow: auto;">
21                  {{ elem.prettify()|safe }}
22              </div>
23              {% endfor %}
24          </div>
25          <div style="display: flex; column-gap: 1rem;">
26              <div id="XRL-demo-left" style="display: flex; flex-direction: row-reverse;
                ↪  column-gap: 1rem;">
27                  {% for elem in xrl_left %}
28                  <!-- XRL-left -->
29                  <div style="max-width: 1000px; max-height: 1000px; overflow: auto;">
30                      {{ elem.prettify()|safe }}
31                  </div>
32                  {% endfor %}
33              </div>
34              <div style="display: flex; flex-direction: column; row-gap: 1rem;">
35                  <div id="XRL-demo-main">
36                      {% for elem in xrl_main %}
37                      <!-- XRL-main -->
38                      <div style="max-width: 1500px; max-height: 1000px; overflow: auto;">
39                          {{ elem.prettify()|safe }}
40                      </div>
41                      {% endfor %}
42                  </div>
43                  <div id="XRL-demo-below" style="display: flex; flex-direction: column;
                    ↪  row-gap: 1rem;">
44                      {% for elem in xrl_below %}
45                      <!-- XRL-below -->
46                      <div style="max-width: 1500px; max-height: 1000px; overflow: auto;">
47                          {{ elem.prettify()|safe }}
48                      </div>
49                      {% endfor %}
50                      {% if remainder %}
51                      <div style="max-width: 1500px; max-height: 1000px; overflow: auto;">
```

```
                        {{ remainder.prettify()|safe }}
                    </div>
                    {% endif %}
                </div>
            </div>
            <div id="XRL-demo-right" style="display: flex; column-gap: 1rem;">
                {% for elem in xrl_right %}
                <div style="max-width: 1000px; max-height: 1000px; overflow: auto;">
                    {{ elem.prettify()|safe }}
                </div>
                {% endfor %}
            </div>
        </div>
    </div>
    <div id="XRL-demo-remainder" style="display: none;">
    </div>
</body>
<footer>
    {% for elem in footer_list %}
    <!-- Footer -->
    {{ elem.prettify()|safe }}
    {% endfor %}
    {% for elem in script_list %}
    <!-- Script -->
    {{ elem.prettify()|safe }}
    {% endfor %}
</footer>
<script>
    document.body.style.transform = "scale(0.5)";
    document.body.style.transformOrigin = "0 0";

    const left = document.getElementById("XRL-demo-left");
    const head = document.getElementById("XRL-demo-head");
    const main = document.getElementById("XRL-demo-main");
    const below = document.getElementById("XRL-demo-below");

    // Create the XRL-demo-head-spacer element with the same width as XRL-demo-left
    const spacer = document.createElement("div");
    spacer.id = "XRL-demo-head-spacer";
    spacer.style.width = window.getComputedStyle(left).width;

    // Prepend the spacer element to the head element
    head.parentNode.insertBefore(spacer, head.parentNode.firstChild);

    // set head width
```

```
    head.style.maxWidth = window.getComputedStyle(main).width >= window.getComputedStyle(below).width ? window.getComputedStyle(main).width : window.getComputedStyle(below).width;

    // Create a ResizeObserver instance to observe changes in the left element's size
    const resizeObserver = new ResizeObserver(entries => {
        for (const entry of entries) {
            if (entry.target === left) {
                spacer.style.width = window.getComputedStyle(left).width;
            }
            if (entry.target === main || entry.target === below) {
                head.style.maxWidth = window.getComputedStyle(main).width >= window.getComputedStyle(below).width ? window.getComputedStyle(main).width : window.getComputedStyle(below).width;
            }
        }
    });

    // Start observing changes in size
    resizeObserver.observe(left);
    resizeObserver.observe(head);
    resizeObserver.observe(below);
</script>
</html>
```

To run this server locally, recreate the file structure as described above, navigate to the root directory, and run the command:

```
python app.py
```

For more information on running flask servers locally, see https://flask.palletsprojects.com