COMPUTERS AND INFORMATICS
2757-8259 2024, Volume 4 Issue 2
DOI: 10.62189/ci.1516319 Research Article

Evaluation of end-user web mashup development

Alwi Bamhdi
Umm Al-Qura University, College of Computing, Department of Computer Sciences, Al Qunfudhah, Makkah, Saudi Arabia, iambamhdi@uqu.edu.sa

Submitted: 15.07.2024
Accepted: 19.12.2024
Published: 31.12.2024

Abstract:

Mashup End-User Programming (EUP) paradigms leverage tools that enable users to customize web content from various data sources, offering a potentially simple, effective, and efficient method for developing end-user applications. Although it is hypothesized that mashup technology is easy to use for individuals without programming skills, this paper examines this claim through an experiment. Using two Application Programming Interfaces (APIs) from Flickr and Google Maps as test cases, participants were tasked with creating meta-applications using one of three mashup tools: Yahoo! Pipes, Intel Mash Maker, or Dapper. The research methodology, measurement methods, and findings are presented, revealing that mashup development is not as accessible for non-programmers as widely believed, highlighting key challenges in end-user application development. The results showed that while participants found mashup tools engaging and transformative in their approach to web development, they struggled with complexity, particularly non-programmers and even some confident programmers. The findings emphasize the need for intuitive, user-friendly mashup tools that simplify development and support effortless end-user programming. This is a research and development challenge mashup facilities should offer in a seamless manner with new supportive paradigms.

Keywords: *End–user development, Mashup, Mashup tools, Meta–applications, Web 1.0 to Web 5.0*

Cite this paper as: Bamhti, A., Evaluation of end-user web mashup development, *Computers and Informatics*, 2024; 4(2); 112-129, https://doi.org/10.62189/ci.1516319

1. INTRODUCTION

Before 1999, users were primarily consumers of web content, and the internet was regarded as a sophisticated tool that only professionals, tech-savvy users, and enthusiasts could engage with. At that time, less than 5% of the global population used the internet, and web technology was still in its infancy with the advent of Web 1.0. However, the latest statistics on global internet usage and digital behaviour are now available. As of October 2024, there are 5.52 billion internet users globally, reflecting a year-on-year growth of 151 million users at a 2.8% annual rate. Social media has also seen significant expansion, with 5.22 billion active users, accounting for 67.5% of the world's population. Additionally, mobile adoption is noteworthy, with 5.75 billion unique mobile users globally, representing 70.3% of the global population [1]. This growth underscores the increasing engagement with digital platforms, highlighting the ongoing relevance of tools like mashups in empowering user-generated content.

Similarly, web technology has evolved rapidly from Web 1.0 to Web 2.0, Web 3.0, and now Web 4.0, with Web 5.0 platforms emerging to offer more than just reading news or checking emails. Emerging web applications and content services, such as smart search functions, have started to evolve into topic-specific searches and crawlers, creating new revenue streams through mechanisms like pay-per-click advertising [2], [3]. As of 2024, web development trends increasingly depend on the preferences and needs of users, with tools like Artificial Intelligence (AI) and chatbots becoming integral to mashup development. Studies such as those by Aghaee & Pautasso [5] and Lian & Tang [7] highlight how mashup tools, particularly those powered by AI or advanced API frameworks, empower non-programmers to develop meta-applications. This democratization is facilitated by simplified platforms (e.g., no-code tools like Zapier or Bubble) and evolving web technologies.

Web mashups are content aggregators that combine various types of content into a more advanced form through remixes, creating meta-applications that cater to users ranging from technical experts to casual users. For instance, by the end of 2010, over 2,500 APIs had been integrated to create more than 5,400 mashups, with new mashups being created daily [6]. In comparison, according to the latest statistics from the largest web API portal, ProgrammableWeb.com, the number of open web APIs now exceeds 24,500 [1]. With such an abundance of web APIs, developers frequently reuse or combine them to create value-added services or new meta-applications through mashup development [7].

Web development mashup tools have been made publicly available through open-source platforms, aiming to provide opportunities for end-users to create their own mashups [8]. However, the rapid expansion of web technology and intense market competition have led to several mashup tools falling by the wayside because they could not keep up with advancements made by their competitors or evolving end-user requirements for remixed meta-applications [9].

In development terms, there is no single industry-standard definition for mashup interworking or interoperability. Mashups epitomize web services technology by fusing data from two or more web applications to create an integrated functionality experience based on the original data sources. Mashup developers dynamically extract data from one source and combine it with another. For example, the Fast Food Maps mashup combines location information of major fast food restaurants with Google Maps, enabling users to see where they can find their preferred meal in a specific locality.

The rapid growth of mashups has led to the widespread belief that anyone can develop them. There is a common perception that users can remix meta-applications from web resources with minimal computing skills. This belief is being tested through the research reported in this paper.

The experiment and evaluation described in this paper were designed to assess this hypothesis and measure the performance of end-user mashup application development. Two different APIs, the Flickr and Google Maps APIs, were remixed to create a meta-application. Participants were tasked with building their own high-level meta-application using one of three mashup tools: Yahoo! Pipes, Intel Mash Maker, or Dapper. After completing the task, participants were given questionnaires to evaluate their experience. The questionnaire aimed to capture users' ideas and opinions on mashup application development, as well as assess the ease of use of the mashup tools. The evaluation focused on exploratory questions such as:

- *What were users' general impressions of the mashup development paradigm?*
- *What did users think about mashup application development in terms of ease of use, programming concepts, and the knowledge required to use these tools?*
- *Given that mashup tools are intended to assist end-user development, what factors influenced their choice of the selected mashup tool?*

The paper begins with a review of mashup development in the context of the evolution of the web as discussed in Section 2. Section 3 outlines the requirements for developing the meta-application. Section 4 details the experiment format, followed by Section 5, which specifies the meta-application's function and architecture. Section 6 presents the evaluation and results of the experimental research, and Section 7 provides discussions and conclusions.

2. MASHUP AND WEB EVOLUTION

Following Web 1.0, referred to as static HTML for publishing and downloading Web page content, the evolution to Web 2.0 and Web 3.0, with their improved functionalities, is the reason why mashups have become possible and popular [10]. Web 2.0 refers to interoperable, interlocking types of services where websites provide components rather than finite, one-stop experiences, encouraging end-user programming (EUP) and meta-application development [11]. Users in this paradigm are therefore free to combine online services in any way they choose. Compared with Web 1.0, Web 2.0 introduced social networking sites like Flickr, which allow many people to build communities, upload, and share content they have created in an interactive and collaborative manner. On Web 2.0 platforms, users are no longer just passive readers while surfing the Internet to search for information and retrieve data; they also help create content by blogging personal and personalized information easily and seamlessly through the functionality offered by Web 2.0 platforms and multimedia applications [9]. In "What is Web 2.0?" O'Reilly [11] explicitly identified remixable data sources and the right to remix, enabling users to create hybrid meta-applications through data reuse, web services, micro-applications, and their APIs.

The evolution of the World Wide Web (WWW) never stops. With the term "Semantic Web," which started with Web 2.0 and grew progressively through ontology and knowledge-based inference, the web has matured and is now referred to as Web 3.0. In 1999, Tim Berners-Lee, the inventor of the first World Wide Web, expressed the vision of the Semantic Web as an intelligent agent [13]. It was to be a web of data that could be processed directly and indirectly by machines. Terms like Web 1.0, Web 2.0, Web 3.0, and Web 4.0 are versions of web technology growth areas coined by the W3C (World Wide Web Consortium). Web 4.0, defined as the Mobile Web, is not necessarily a new version but an alternative to Web 3.0, designed to adapt to mobile environments. Web 4.0 connects all devices in both the real and virtual world in real time, while still heavily relying on Web 2.0 and Web 3.0 technologies. Web 5.0, the open, linked, and intelligent "emotional web," has yet to make a significant impact on the application and programming scene [14]; we are not there yet!

For example, searching for Web 3.0 and Web 4.0 separately on Google results in an astounding total of about 1.77 billion search results in just 0.54 seconds for Web 3.0, and about 4.9 billion search results in 0.51 seconds for Web 4.0 [15]! This demonstrates the powerful search capabilities of Google, which can return vast amounts of related content in a very short time. However, it does not provide users with substantive, sophisticated help since one must click and open each link to determine its relevance. Users often experience information overload, feeling that they are receiving lots of useless information. Although search results can be refined with more precise queries, the intelligence required for this task lies with the end-user, rather than the search system performing more sophisticated, semantically-aware searches assisted by ontologies. This is one key problem that Web 3.0 aims to solve.

Web 3.0, as described by Wikipedia, encompasses a set of methods and technologies designed to enable machines to understand the meaning, or semantics, of information on the World Wide Web [15]. Building on this definition and through a more comprehensive analysis, Table 1(a) highlights the evolution of technology, applications, and challenges by comparing and contrasting the characteristics of Web 1.0, Web 2.0, and Web 3.0. Meanwhile, Table 1(b) elaborates on the distinctions in their features, technologies, applications, and impacts, aligning these with modern trends in Web 3.0, such as blockchain, the Metaverse, and semantic capabilities.

Table 1(a). Comparison of different versions of web technology

Versions vs. Properties	Web 1.0	Web 2.0	Web 3.0
Timeline	1991–2003	Since 2004	The term became famous in 2006 and is still evolving to date.
Type of functions	Read–only	Read–write–publish	Semantic Web: AI-driven context and decentralized systems.
Buzzword	Online	AJAX RSS	Interactive 3D content, Blockchain, Dapps, Metaverse.
Pros	Human Computer Interactive (HCI)	Facilitate collaboration & sharing between users	Allows seamless sharing and integration of information across data domains. Richer semantic functionality and decentralized systems. Offer a richer set of facilities & remix options.
Cons	Limited average Internet user's role	Information overload	Security concerns: private data, copyright issues, and high complexity.
Outcomes	Dot (.) com boom	Services like YouTube, Facebook, Wikipedia, Folksonomies, TikTok, etc.	Contextual search, Semantic reasoning, Ontology-based support, Blockchain apps, NFTs, DAOs, Metaverse.
Examples	Static websites	Blogs, Wikis, Social media platforms.	Decentralized Finance (DeFi), NFTs, Ethereum-based Dapps, Decentraland.
Key Technologies	HTML, HTTP, Static pages	AJAX, APIs, Dynamic content.	Blockchain, Smart Contracts, AI, Linked Data, Decentralized Storage (IPFS).
Notable Features	Limited interactivity	User-generated content, Collaboration.	Decentralization, Semantic Web, AI, Ownership of data, Privacy-centric systems.

Table 1(b): Overview of Web 3.0 Features, Technologies, Applications, and Impacts

Category Key Technologies	Description	Examples
Blockchain	Decentralized, secure data storage and transaction technology.	Ethereum, Bitcoin
Smart Contracts	Self-executing contracts coded directly onto blockchain networks.	Solidity-based contracts, Hyperledger Fabric
Decentralized Storage	Distributed systems for censorship-resistant and secure data storage.	IPFS, Filecoin
Interoperability	Seamless interaction between different blockchain networks and platforms.	Polkadot, Cosmos
Applications		

Decentralized Apps (Dapps)	Applications built on blockchain with no centralized authority.	Uniswap, OpenSea, Lens Protocol
Decentralized Finance (DeFi)	Blockchain-based financial systems eliminating intermediaries.	Aave, Compound, MakerDAO
Metaverse	Immersive virtual environments combining AR, VR, and blockchain.	Decentraland, The Sandbox
NFTs	Digital tokens proving ownership of unique assets.	CryptoPunks, Bored Ape Yacht Club
Decentralized Autonomous Organizations (DAOs)	Community-driven organizations governed by blockchain-based smart contracts.	MakerDAO, ConstitutionDAO
Web3 Identity	User-centric identity systems promoting privacy and self-sovereignty.	Ethereum Name Service (ENS), Spruce
Features		
User-Centric	Empowering individuals to own and control their digital identities and assets.	Web3 wallets like MetaMask
Semantic Web	Leveraging AI and ontologies to enable machines to "understand" and connect data meaningfully.	RDF, OWL, SPARQL
Trustless Systems	Reducing reliance on intermediaries via cryptographic verification and consensus mechanisms.	Proof of Work (PoW), Proof of Stake (PoS)
Impacts		
Economic Decentralization	Allowing individuals to monetize digital assets without centralized oversight.	Artists earning royalties via NFTs
Global Inclusion	Expanding access to digital tools and services, especially in underserved areas.	Blockchain-based remittance systems
Data Ownership and Privacy	Users reclaim control of their data, reducing exploitation by large platforms.	Decentralized social platforms like Mastodon

In summary:

Mashups are applications created by combining data, services, or functionalities from multiple sources into a single tool or interface. They became popular with Web 2.0, which introduced user-friendly, interactive web technologies that allowed non-programming users to assemble and customize applications without needing deep technical expertise. Tools like Google Maps APIs and drag-and-drop platforms enabled this democratization of app development.

With the advent of Web 3.0 (focused on the semantic web, decentralization, and AI), Web 4.0 (integration of IoT and smart systems), and discussions of Web 5.0 (emphasizing human-AI symbiosis and emotional computing), the scope for mashups has evolved significantly. Today, users can leverage no-code/low-code platforms powered by AI to build highly sophisticated applications. For instance, AI tools can interpret user needs, connect APIs, and structure workflows with minimal human intervention.

Thus, mashups in the Web 3.0, 4.0, and 5.0 eras are not only possible but more accessible than ever, thanks to advancements in AI, semantic technologies, and no-code platforms, enabling seamless development by non-programming users.

3. GUIDELINES AND REQUIREMENTS FOR META-APPLICATION DEVELOPMENT

3.1 Preamble

Meta-application development represents the next frontier of software engineering, integrating a variety of services and applications to form more complex and dynamic systems. These applications combine functionalities from multiple domains, often in ways that were previously unimaginable. This approach, fostered by Web 2.0 and beyond, brings about several requirements and challenges that developers need to address to build efficient, robust, and user-centric meta-applications.

3.1.1. Interoperability and integration

The primary requirement in meta-application development is seamless interoperability and integration between disparate systems. Meta-applications often involve the combination of services and components from various domains, including databases, web services, APIs, and cloud platforms. Developers must ensure that these components can communicate effectively, which often involves working with APIs, middleware, and other integration technologies. This ensures that different systems can exchange data and functionality in a manner that is transparent to the user.

Interoperability also extends to platforms and devices. For example, a meta-application might aggregate data from a social media platform, weather service, and an IoT device. Ensuring that these services function cohesively in real-time without compatibility issues is one of the key challenges of meta-application development [12].

3.1.2. Scalability and flexibility

Scalability is a crucial requirement for meta-applications, as they are expected to handle growing amounts of data and increasing numbers of users without compromising performance. This involves designing the application to be flexible enough to scale horizontally across servers or to integrate new services or functionalities seamlessly.

Web 3.0 technologies such as blockchain and decentralized storage systems require meta-applications to be scalable, as they often involve complex data structures and large-scale distributed systems. In this context, flexible architecture and dynamic resource allocation play a vital role in ensuring the application can handle growth while maintaining efficiency.

3.1.3. Security and privacy

As meta-applications aggregate and process large amounts of user data from various sources, security and privacy become critical concerns. Developers must implement robust authentication and authorization mechanisms to ensure that only authorized users can access specific features. Furthermore, data encryption is essential for protecting sensitive information, especially when dealing with personal data, financial transactions, or medical records.

Meta-applications built on decentralized technologies such as blockchain must also consider the implications of distributed data storage. While decentralized systems often offer increased security, developers must account for risks such as data breaches, fraud, and other malicious activities. Implementing mechanisms such as end-to-end encryption, zero-knowledge proofs, and secure multi-party computation can help mitigate these risks and ensure data privacy [13].

3.1.4. Usability and user experience (UX)

Meta-applications often combine services from different domains, so providing a smooth, intuitive user experience is essential. This requires careful design and attention to how users interact with the application. A well-designed meta-application should allow users to easily navigate between integrated services without confusion or disruption.

Developers must focus on creating user interfaces (UIs) that are simple and consistent, even when they incorporate complex backend functionalities. The rise of low-code/no-code platforms, which enable non-technical users to build applications, further emphasizes the importance of creating highly intuitive UIs. This trend in Web 3.0 technologies means that the integration of AI and machine learning (ML) to

enhance user experience and provide personalized services is becoming a key feature of meta-applications [14].

3.1.5. Data management and semantics

Effective data management is at the heart of meta-application development. Meta-applications need to handle data from multiple sources, often in varying formats and structures. Developers must ensure that the data is cleaned, transformed, and stored in a manner that is useful and accessible.

Web 3.0's focus on the semantic web emphasizes the need for meta-applications to interpret and link data meaningfully. Ontologies and data schemas play a critical role in structuring and linking data across different domains. The use of machine learning algorithms and AI to extract valuable insights from this data is a major development area. Meta-applications must be capable of utilizing this data efficiently and offering rich semantic search capabilities to improve user interactions [15].

3.1.6. Real-time processing and updates

Given the dynamic nature of meta-applications, real-time processing is another vital requirement. As these applications often aggregate live data from multiple sources—such as IoT sensors, user-generated content, or streaming services—developers must ensure that the application is capable of processing and displaying this data without latency. This requires integrating real-time data processing capabilities, such as WebSockets or server-sent events, into the architecture.

Furthermore, meta-applications should be able to provide real-time updates or notifications to users when critical events occur, such as changes in data or the availability of new content. In the context of decentralized applications (dApps) or blockchain-based applications, developers must ensure that updates are broadcasted securely across the network in a timely manner.

3.1.7. Cost efficiency and sustainability

As meta-applications typically rely on various third-party services, cloud storage, or decentralized networks, cost efficiency is an important consideration. Developers must optimize the use of resources, minimize redundant operations, and choose scalable cloud services to ensure the application is cost-effective over time. This includes optimizing the storage and computing needs of blockchain-based applications or distributed ledgers, which often come with high operational costs.

Moreover, as the impact of software development on the environment becomes a growing concern, developers must take sustainability into account. This may involve leveraging energy-efficient data centers, reducing the energy consumption of decentralized systems, or optimizing computational tasks to reduce their carbon footprint.

3.1.8. Cross-platform compatibility

Given the increasing diversity of devices and operating systems used by consumers, cross-platform compatibility is an essential requirement for meta-application development. Meta-applications need to operate smoothly on desktops, mobile devices, and even wearable technologies, which can have varying screen sizes, processing power, and input methods.

Using responsive design techniques, progressive web apps (PWAs), and ensuring that the application is compatible across different operating systems such as iOS, Android, and Windows is crucial for maximizing the reach of the meta-application. Developers must also consider accessibility standards to ensure the application is usable by individuals with disabilities.

In conclusive summary, Meta-application development presents several unique requirements, from ensuring interoperability and security to managing large volumes of data effectively. The evolution of the web, particularly Web 3.0 and beyond, introduces both challenges and opportunities, including the integration of decentralized technologies, real-time data processing, and enhanced user experiences through AI and semantic web technologies. By focusing on these core requirements, developers can create sophisticated, scalable, and user-centric meta-applications that leverage the full potential of modern web technologies.

3.2 Assessments Based on Precise Experimental Requirements

To assess the requirements for End-User Programming (EUP) application development, meta-applications were created by integrating different sets of APIs and data sources from the following categories:

Flickr's Geotagged Photos: Photos can be searched by keywords or labels that users add to a photo, making it easier to find.

Google Maps APIs: A series of predefined standards and interfaces used by developers to access Google Maps and create Google Map-related applications.

Firefox: A free and open-source web browser. New functions can be added through extensions, created by third-party developers. Firefox's wide selection of extensions has attracted many users.

The core of a mashup meta-application is the assembly of data from multiple data sources. There are two common patterns in mashup meta-application development: *without a tool* and *with a tool*.

3.2.1. Without a tool

A meta-application developed without a tool refers to the direct integration of APIs from different websites. In this case, the Flickr APIs and the Google Maps APIs were mashed up together. All procedures were completed in a PHP implementation environment, and developers were required to have a programming background in order to analyze and handle the different APIs.

3.2.2. With a tool

A meta-application developed with a tool refers to the use of widgets or other substitute methods to facilitate programming solutions. Users need to understand the working philosophies of mashup tools before using them.

Thus, to create the mashup meta-application, users have two alternative options to meet their different needs: using suitable mashup tools if they are novice or non-programmers, or using traditional APIs if they are familiar with programming and API development.

4. EXPERIMENTAL SETUP

To evaluate the process of developing mashup meta-applications, the experiment introduced two options for participants to create their own applications. The experiment aimed to compare the experiences and outcomes of developing a mashup meta-application with and without the use of

mashup tools. The primary procedures, shown in Figure 1, were planned and implemented in four distinct steps:

Meta-Application Creation Using APIs and Programming Languages: In this step, participants were tasked with integrating Flickr APIs and Google Map APIs using programming languages such as PHP and JavaScript. This approach required participants to manually code the interaction between the APIs to create a mashup. The objective was to provide insight into the challenges and complexities involved when using traditional methods (i.e., without a mashup tool) and to assess how well participants with varying levels of programming skills could manage this task.

Introduction to the Meta-Application Concept: To ensure that participants had a basic understanding of what they were tasked with, a brief introduction to the concept of mashups and the specific meta-application being developed was provided. This introduction included an explanation of how different data sources, such as Flickr and Google Maps, could be combined to create a cohesive user experience. The goal was to ensure that participants understood the functional aspects of the meta-application before beginning their work.

Demonstration of Mashup Tools: To assist participants in understanding the potential of mashup development tools, related demonstration videos were shown. These videos focused on the features and functions of the selected mashup tools (Yahoo! Pipes, Intel Mash Maker, and Dapper). Each tool was introduced with a focus on how its drag-and-drop interface, widget-based architecture, and API integration capabilities could simplify the process of mashup development. The aim was to provide participants with a clear understanding of how these tools could make the development process more accessible, especially for those with limited programming skills.

Creation of Meta-Applications Using Mashup Tools: In this final step, participants were asked to create their own mashup meta-application using one of the mashup tools they preferred or found easiest to understand. They were given the freedom to choose from the three tools presented, based on their individual preferences or prior experience. This phase allowed the researchers to compare how participants interacted with the mashup tools versus traditional API coding, measuring the ease of use, speed of development, and functionality of the final product.

The objective of the experiment was to assess the usability and effectiveness of mashup tools in simplifying the development of meta-applications, particularly for end-users without programming backgrounds. By comparing the results of these two development approaches, the study aimed to determine whether mashup tools truly offered a viable solution for non-programmers to create complex web applications efficiently.

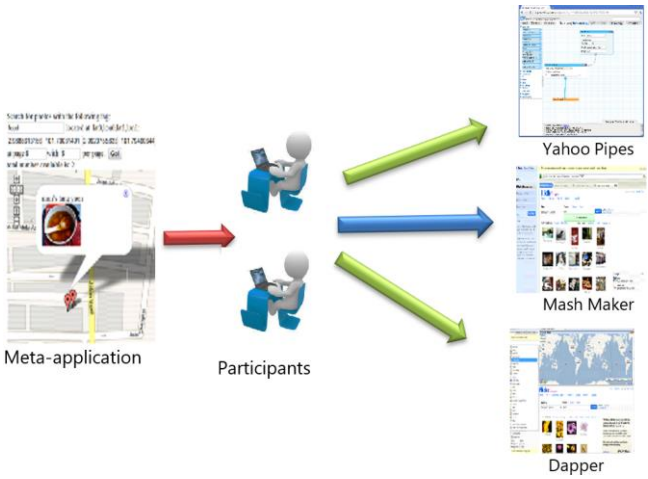

Figure1 procedure of the experiment setup

5. META-APPLICATION SPECIFICATION

The primary function of the meta–application is to enable users to search for geotagged photos on Flickr and display them on a map. These photos are tagged with geographical coordinates, such as longitude and latitude. When an end–user interacts with the map—by panning, zooming, or adjusting the zoom level—the map's southwest and northeast corners are dynamically updated. Similarly, when a new search for geotagged photos is performed, the application displays the resulting images on the map. This process ensures that the photos are shown in a synchronized and visually appealing manner, maintaining an effective and user-friendly interface.

5.1. Integrate Flickr with Google Map

As a meta–application designed to evaluate mashup tools, it was essential that the application be a simple yet common example of mashup development, meeting the core requirements of integrating data from at least two different sources. Furthermore, the functionality of the meta–application needed to be accessible and straightforward enough for average users to navigate effectively.

To achieve this, Google Maps and Flickr were chosen as the data sources for the meta–application due to their widespread use and the following considerations:

According to statistics from Programmable Web [6], [19], by the end of 2010, 30% of the tracked mashup applications were based on Google Maps. Google Maps was one of the first platforms to offer publicly accessible APIs for mashup development, making them one of the most widely adopted and utilized APIs due to their early launch, exposure, and utility.

Flickr, which was originally developed by Ludicorp and later acquired by Yahoo!, is considered a highly mashup-friendly platform. It facilitates the creation of meta–applications due to several features it offers, such as XML (eXtensible Markup Language), XML Web services, tagging, and Ajax (Asynchronous JavaScript and XML). Flickr's ability to handle asynchronous data retrieval via Ajax without disrupting the page's current behavior makes it an excellent candidate for mashup development. These features seamlessly integrate, enabling developers to leverage new technologies to create sophisticated applications [10].

The functional architecture of the meta–application is designed to retrieve results for Flickr geotagged photos using the Flickr APIs, which are then displayed in the browser through XHR (XMLHttpRequest) and a server-side proxy. This proxy acts as an intermediary to bridge client-side scripts with the Flickr server. Figure 2 illustrates the communication process between the browser and the Flickr server via the proxy server.

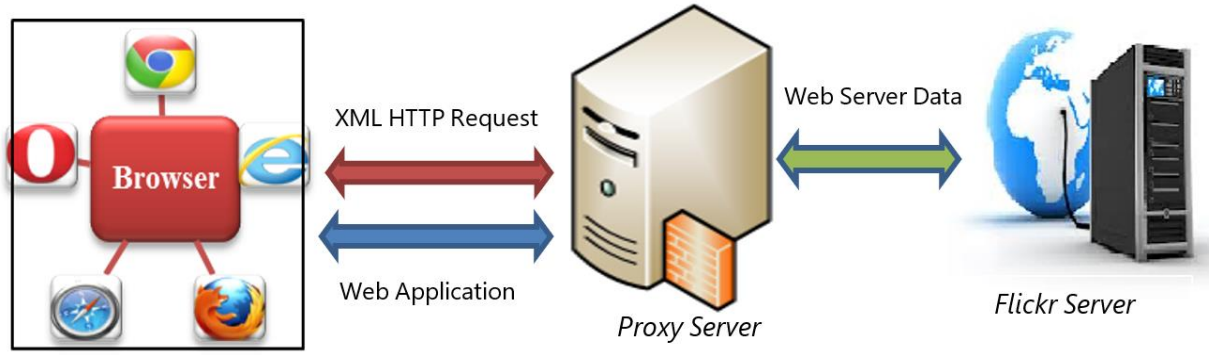

Figure 2 Communication between browser & flickr server via proxy server

In terms of structure, the meta–application falls under the category of basic mashups since it integrates data from only two sources: Google Maps and Flickr. The user interface is dynamic and visually engaging, combining geographical data with photo information. After finalizing the application's core functionality, it was introduced to the participants, who were tasked with integrating the meta–application using one of the available mashup tools. Following this, participants completed a related questionnaire to assess the ease of use and effectiveness of the mashup tools in the development process.

5.2. Mashup tools Specification

Three Mashup tools–Yahoo! Pipes, Intel Mash Maker and Dapper were selected:

Dapper: The company behind Dapper (a tool for creating APIs from websites) was based in *Berkeley, California, USA*. It was acquired by Yahoo! in 2010.

Yahoo! Pipes: This was developed by *Yahoo!*, which is headquartered in *Sunnyvale, California, USA*. Yahoo! Pipes was an online service for data mashups.

Intel Mash Maker: This experimental tool was developed by *Intel Corporation*, headquartered in *Santa Clara, California, USA*.

These mashup tools were selected and given to the respondents to implement the integrated mashup application. The selection of these was based for the differences in their user interface, functionality and data acquisition capabilities. The interfaces of these three mashup tools are shown in Figure1. Table 2 shows the basic information of the selected mashup tools in a comparative manner with their respective URLs.

Table 2. Review of Three Selected Mashup Tools

Name	Services offered	URL
Dapper	Nontechnical interface platform	http://open.dapper.net/dapperDemo/
Yahoo! Pipes	Individual programming environment	http://pipes.yahoo.com/pipes/
Intel Mash Maker	Copy–and–paste Web content	http://mashmaker.intel.com/web/

The study's methodology included ***purposive sampling*** and ***structured training***. Purposive sampling involved selecting participants with relevant experience or interest in using mashup tools, while structured training ensured that all participants had the necessary knowledge to use the tools effectively. This approach helped maintain transparency and rigor in the study's design, as recommended by research guidelines.

6. EVALUATION AND RESULTS

6. 1. Participant's portfolio

Forty (40) participants were selected from different disciplines and provided training on the concept of mashups. Thirty (30) of them successfully created or attempted to create a meta–application and completed the questionnaire afterward. The general characteristics of the participants are summarized in Table 3.

Table 3. General Characteristics of the Participants

Parameters	Characteristic	Number	Distribution (%)
Age	20–25	9	30%
	25–30	12	40%
	30–35	9	30%
Education Level	Undergraduate	10	33%
	Postgraduate	14	46%
	PhD	4	13%
	MBA	2	8%
Gender	Male	23	77%
	Female	7	23%

Most participants were postgraduate university students, aged between 20 and 35 years. This age group represents over half of the internet population and is typically more active and technically sophisticated compared to other adult internet user groups [12] [22]. Thirty percent (9) of the participants had at least one year of experience in web development and worked in ICT at a technical level.

Table 4 highlights the varying levels of prior experience in web development. Although 17% (5) of participants frequently developed web applications, 43% (13) had never developed a web application before.

Table 4. Participants Web Application Development Experience

Web Developing Experience	Numbers of Participants	Distribution Percentages
Frequently used	5	17%
Few used	12	40%
Never used	13	43%

Table 5 shows participants were classified into two general categories based on their programming skill levels: 37% (11) non–programmers and 63% (19) programmers. Only 6% (2) participants considered themselves experts in programming, which aligns with their demographic profile as highlighted earlier.

Table 5. Programming Skill Level of the Participants

Programmer / Developer Types	Level of Programming	Number of Participants	Distribution Percentages
Non–programmer	Non–programmers	11	37%
	Beginner	8	27%
Programmer	Average	9	30%
	Expert	2	6%

The results were generated through comparing and analysing the differences between programmers and non–programmers in their understanding of the performance of mashup development and mashup tools. When asked to report the level of their programming skills, only 6% (2) considered themselves expert. This was consistent with their demographic profiles introduced earlier.

Table 5 shows the participants who were divided into two general categories according to their programming skill level: 37% (11) non–programmer and 63% (19) programmers.

6.2. Motivation and Confidence

After attending the step–by–step training on mashup development and tools, participants were asked about their motivation and confidence in creating a mashup. The results indicate the following: 60% (18) of the participants found Web mashups easy to understand. 27% (8) remained neutral, while 13% (4) expressed a negative attitude toward mashups.

As exposed by [13], [20] that mashup tools were easy to create by showing the demonstration videos on mashup creation, 90% (27) of participants expressed curiosity and interest in the process, while 73% (22) believed that mashup tools were easy to understand.

The data collected on participants' confidence in creating the meta–application is shown in Table 6.

Table 6. Non–programmer & Programmer Confidence on Developing the Meta–application

Confidence in Developing Meta-applications	Non-programmer	Programmer
Positive in Confidence	0	32% (6)
Neutral in Confidence	36% (4)	53% (10)
Negative in Confidence	64% (7)	15% (3)

From Table 6, it is clear that programmers felt significantly more confident about creating the meta–application compared to non–programmers. None of the non–programmers reported positive confidence in completing the meta–application. Meanwhile, among the programmers, 32% (6) felt confident, and 53% (10) were unsure. A higher proportion of non–programmers (64%, 7) felt negative about their ability to develop the application.

Overall, the results also indicate that programmers' confidence was obviously higher than non–programmers and non–programmers generally lacked confidence in developing the meta–application. The fact is that the programming level is still considered as a key factor which influences the confidence of participants in their subconscious.

6.3. Mashup Tools with Selection Distribution

Following an introduction to the basic characteristics of Yahoo! Pipes, Intel Mash Maker, and Dapper, participants were asked to choose their preferred mashup tool. The results are shown in Table 7.

Table 7. Distributions of Mashup Tools Selection by Participants

Mashup Tools	Numbers of Participants	Distribution Percentages
Dapper	13	44%
Yahoo! Pipes	15	50%
Intel Mash Maker	0	0%
Others	2	6%

An interesting observation is that none of the participants selected Intel Mash Maker, as they preferred not to use plug-ins in their web browsers. Many participants believed that plug-ins would slow down their browsing experience. Yahoo! Pipes was the most popular choice, with 50% (15) of participants selecting it, followed by Dapper at 44% (13).

6.4. Average Time Spent

The time spent on mashup tools encompassed both learning the basic infrastructure and prototyping the meta-application mashups. This process reflected the fundamental cognitive learning curves and challenges associated with using mashup tools from an end-user's perspective, as illustrated in Figure 3. To validate the experimental process, a pilot study was conducted with two students: one programmer and one non-programmer. This approach provided a balanced control, capturing both technical and non-technical perspectives.

Each student underwent a 1-hour training session that involved viewing a demonstration video explaining the features and functionality of mashup tools in meta-application development.

Subsequently, they spent approximately 1 hour familiarizing themselves with the tools' functions and another hour developing a mashup meta-application. The combined 3-hour duration was established as the benchmark for this experiment, providing a structured framework for evaluating the learning and development phases. This benchmark also helped standardize the evaluation process, ensuring consistency across participants regardless of their programming background.

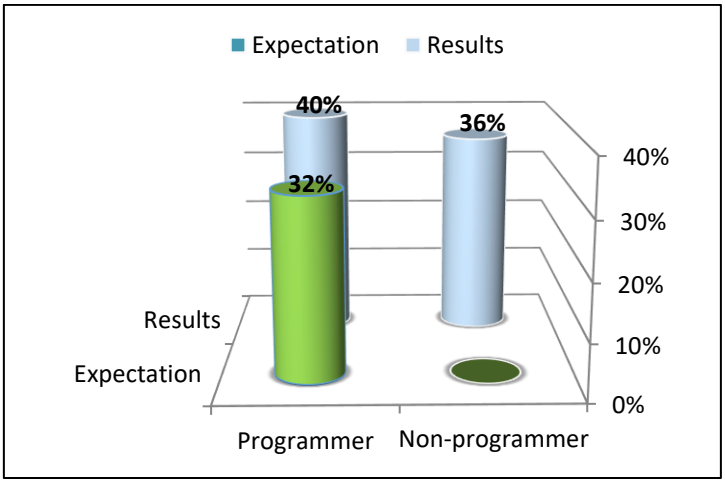

Figure 3 Contrary results on confidence and developing the meta–applications

Yahoo! Pipes, which was chosen by the majority of the users in this research study, took, on average, about 4 hours to complete—much longer than expected. On the other hand, users selecting Dapper spent an average of 4 hours just analyzing the interface and working philosophy alone, compared to the allotted time of 3 hours.

6.5. Variables Examined Mashup Tools

After the participants chose their preferred mashup tool and attempted to create the meta-application, 63% (19) experienced difficulties using the mashup tool. Among these 19 participants, 37% (7) were unable to solve the problems. The remaining 17% (11) found the mashup tools easy to handle and operate, without encountering any serious barriers that might delay or block the creation of the meta-application. Table 8 shows the evaluation criteria and variables used to assess the ease of use of the mashup tools.

Table 8. Evaluated Variables of Mashup Tools

Evaluation Criteria as Measurement Variables	Overall Opinion Offered	Number of Samples	Distribution Percentages
V1: Easy to get started	Agree	19	63%
V2: Friendly interface	Agree	18	60%
V3: Facilitate EUP	Agree	19	63%
V4: Getting real-time hints and help	Agree	9	30%
V5: Seeing related videos and publications for help	Agree	18	60%
V6: Searching out similar applications for help	Agree	14	47%
V7: Prefer pure programming than Mashup tools	Agree	9	30%

The data captured shows that 63% (19) of participants were able to apply the mashup tools directly and easily without any difficulties. 60% (18) had a positive opinion about the user-friendly interface provided by the three mashup tools, and 63% (19) believed that the mashup tool development paradigm was capable of facilitating end-user programming.

However, only 30% (9) of participants felt they received timely help when using the mashup tools. One user noted that the current mashup tools lacked real-time hints and assistance, which was seen as a key factor influencing efficiency and confidence in end-user application development. The search function, used to find similar applications, was ignored by nearly 53% (16) of participants when creating the meta-application. Fortunately, 60% (18) participants sought help through related videos and documentation. Although 63% (19) of participants found it difficult to handle the mashup tools, 30% (9) still agreed that this programming paradigm is easier than writing pure programming code.

6.6. Lack of Confidence

As the previous figures showed in the motivation and confidence section, 32% (6) participants had confidence in completing the meta–application. In fact, there were 40% (7) programmers who completed the meta–application. The completion rate of non–programmers was 36% (4). It was quite contrary with their preceding expressions of confidence that none of them had confidence in the completion of the meta–application. Figure 3 shows that the overall results were better than expected in the numbers completing the meta–application.

6.7. Ease of Use Assessed

The findings underscore significant challenges faced by participants in utilizing mashup tools effectively within the constraints of the experiment. Only 37% (11) of participants managed to complete the meta-application within the allocated 60 minutes, highlighting the difficulty of the task. The pre- and post-experiment questionnaire results, detailed in Table 9, further reveal a striking gap between participants' expectations and their actual experiences with the tools.

Notably, 60% (18) of participants initially believed developing a meta-application would be straightforward. However, post-experiment results showed a stark contrast: 63% (19) reported that even creating a simple meta-application was challenging. This reversal in perception strongly indicates that the mashup tools were not as intuitive or user-friendly as participants had anticipated.

The data also emphasize a broader issue: 27% (8) of participants who initially found the tools easy to understand ultimately failed to complete the task. Furthermore, a significant 67% (20) stated that the tools were too complicated to use effectively for developing a meta-application. These findings reveal a fundamental disconnect between the design of mashup tools and the actual needs and capabilities of end-users, suggesting the need for more accessible interfaces, better training resources, or both. The outcomes highlight the importance of addressing these usability challenges to enhance the practical application and adoption of mashup tools.

Table 9. Before & after questionnaire results of mashup

Questions Before Experiment	Questions After Experiment
60% (18) participants considered Web mashup to be easy to understand.	27% (8) participants who thought mashup was easy but finally failed in the building task.
73% (22) of participants considered mashup tools to be easy to understand.	43% (13) EUPs were unable to complete the mashup building task.
-- oOo --	63% (19) of the participants had difficulties & among them 37% (7) were unable to solve the problem at hand.
	60% (18) of the participants thought the mashup development process required programming support.

Most participants attributed their failure to the difficulties encountered in searching for and obtaining useful information or solutions within the allotted time, either through their own initiative, from fellow colleagues, or from other online developers, as also pointed out by [18].

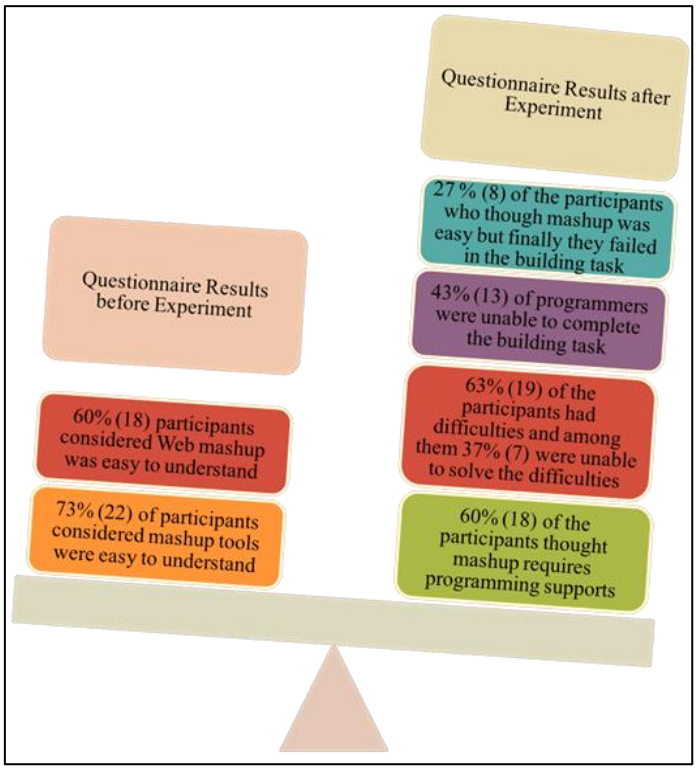

Figure 4 Before and after results on assessment of mashup and mashup tools

Figure 4 graphically illustrates the significant discrepancy between the anticipated usability of mashup tools and their actual outcomes. The sizable gap between the expected usability of mashup tools and the final results highlights the need for further investigation into more advanced end-user mashup and meta-mashup development techniques and technologies, along with easy-to-use supporting tools.

7. DISCUSSION AND CONCLUSION

Web mashups, facilitated by mashup tools, were perceived as a rapid and easy method for end-user development of meta-applications, particularly by non-programmers or those with basic ICT skills. With the rapid evolution of web technologies—from Web 1.0 to Web 2.0 and now Web 3.0—there are ongoing movements within W3C and related organizations that highlight the potential for providing users with unprecedented opportunities for customizing web content through context-aware and semantically rich scenarios. This progression is likely to extend beyond Web 4.0 and Web 5.0, which are often referred to as the intelligent or emotional web, and will eventually be fully utilized by new applications supported by easy-to-use API programming [23].

The results based on the existing Web 2.0 technologies show that users' motivations and confidence in engaging with web content remixing are fairly well understood. Remixing made participants rethink the role of web technology in the context of mashup application development, which many initially viewed as a domain for experienced programmers. However, they felt excited, curious, and engaged when they realized that anyone—regardless of technical expertise—could become the owner of web resources, rather than leaving this role to just skilled programmers and service providers. With the advent of Web 3.0 and Web 4.0 technologies, such as the semantic web and ontologies, the development of meta-applications has become considerably easier.

The results of the experiments further revealed that participants selected mashup tools based on simple criteria and their previous experiences. They preferred using online mashup tools over browser plug-ins and extensions. However, the overall findings indicate that mashup development, even with mashup tools, is still a challenge for non-programmers who lack sufficient ICT skills. Even 50% of the programmers among the respondents, who were confident in completing the task, failed due to the complexity of concepts and the availability of underlying APIs and new paradigms [23]. Users are seeking simple, easy, and stable mashup tools that are intuitive and easy to use in real time. The tools must not only have a user-friendly interface but also feature characteristics that fully support rapid and effortless end-user development.

Recent research by Lian & Tang [22] has explored the challenges and non-trivial issues related to APIs for mashups. They suggested that API recommendations for mashup creation should be based on neural graph collaborative filtering, rather than just using content-based APIs. However, this approach requires further experimental validation and research.

Mashup development should be presented as a simple, process-oriented technology to encourage novice users to adopt it and inspire them to create more complex meta-applications with ease and confidence. The lessons learned from this study suggest that more advanced and user-friendly mashup tools should be developed and made available if this technology is to gain widespread adoption and popularity. The findings of this study emphasize the need for experts, researchers, and developers within the web technology and programming communities to make the process truly user-friendly, as users are often not experts.

Acknowledgment

The author expresses heartfelt gratitude to Professor Ahmed Patel and Na Liu for meticulously validating the technical accuracy and diligently proofreading this manuscript.

REFERENCES

[1] Datareportal. Global Digital Insights [Internet]. Datareportal; [cited 2024 Jun 6]. Available from: https://datareportal.com/global-digital-overview#:~:text=Internet%20use%20around%20the%20world,500%2C000%20new%20users%20each%20day

[2] Patel A, Khan MJ. Evaluation of service management algorithms in a distributed web search system. *Comput Stand Interfaces. 2007*;29(2):152–60. Available from: https://doi.org/10.1016/j.csi.2006.03.002

[3] Schmidt N, Patel A. Design and implementation of a distributed search and advertising system. Proceedings of the 7th International Conference on Information Integration and Web Based Applications & Services (iiWAS 2005); 2005 Sep 19–21; Kuala Lumpur, Malaysia. Available from: https://api.semanticscholar.org/CorpusID:1102286

[4] Westagilelabs. Web development trends [Internet]. Westagilelabs; [cited 2024 Jun 6]. Available from: https://www.westagilelabs.com/blog/what-is-going-to-be-new-web-development-trends-in-2022/

[5] Aghaee S, Pautasso C. End-user programming for web mashups. In: Harth A, Koch N, editors. Current Trends in Web Engineering. ICWE 2011. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; *2011*. p. 347–51. Available from: https://doi.org/10.1007/978-3-642-27997-3_38

[6] Programmableweb.com. Mashup timeline [Internet]. Programmableweb; [cited 2024 Jun 6]. Available from: http://www.programmableweb.com/

[7] Tang M, Xia Y, Tang B, Zhou Y, Cao B, Hu R. Mining collaboration patterns between APIs for mashup creation in web of things. *IEEE Access. 2019*;7:14206–15. Available from: https://doi.org/10.1109/ACCESS.2019.2894297

[8] Gao Y. Design and implementation of end-user programming tools for web mashups [master's thesis]. Edmonton: University of Alberta; *2019*. Available from: https://era.library.ualberta.ca/items/d27775ea-4c5e-4299-9132-14dc9c911445

[9] Liu N, Patel A, Latih R, Mulla R, Shukur Z, Wills C. A study of mashup as a software application development technique with examples from an end–user programming perspective. *J Comput Sci. 2010*;6(11):1406–15. Available from: https://doi.org/10.3844/jcssp.2010.1406.1415
[10] Rognsoy TM. Using ERP mashups to improve business processes. *Glob J Enterp Inf Syst. 2017*;9(3):1–15. Available from: https://gjeis.com/index.php/GJEIS/article/view/302
[11] O'Reilly T. What is Web 2.0? [Internet]. O'Reilly Media; [cited 2024 Jun 6]. Available from: http://oreilly.com/web2/archive/what-is-web-20.html
[12] Cake M. Web 1.0, Web 2.0, Web 3.0 and Web 4.0 explained [Internet]. Marcus Cake; [cited 2024 Jun 6]. Available from: http://www.marcuscake.com/key-concepts/internet-evolution
[13] Berners-Lee T, Fischetti M, Dertouzos ML. Weaving the web: the original design and ultimate destiny of the World Wide Web by its inventor. HarperOne; *1999*. ISBN: 9780062515872. Available from: https://www.amazon.com/Weaving-Web-Original-Ultimate-Inventor/dp/0062515861
[14] Flatworldbusiness. Web 1.0 vs Web 2.0 vs Web 3.0 vs Web 4.0 vs Web 5.0 – a bird's eye on the evolution and definition [Internet]. Flatworldbusiness; [cited 2024 Jun 6]. Available from: https://flatworldbusiness.wordpress.com/flat-education/previously/web-1-0-vs-web-2-0-vs-web-3-0-a-bird-eye-on-the-definition/
[15] Wikipedia. Semantic Web [Internet]. Wikipedia; [cited 2024 Jun 6]. Available from: https://en.wikipedia.org/wiki/Semantic_Web
[16] Daniel F, Gaedke M, editors. Rapid mashup development tools: Second international rapid mashup challenge, RMC 2016 proceedings, Lugano, Switzerland, June 6, 2016. Springer; *2017*. ISBN 9783319531748. Available from: https://www.researchgate.net/publication/321531132_Rapid_Mashup_Development_Tools_Second_International_Rapid_Mashup_Challenge_RMC_2016_Lugano_Switzerland_June_6_2016_Revised_Selected_Papers
[17] Pomonis T, Christodoulou SP, Gizas AB. Towards Web 3.0: a unified development process for web applications combining semantic web and Web 2.0 technologies. *Eng Manag Rev. 2013*;2(2):1–12. Available from: https://ia801703.us.archive.org/30/items/EMR026/EMR026.pdf
[18] Na L. Evaluation of mashups from end-user application development perspective [master's thesis]. Bangi: Universiti Kebangsaan Malaysia; *2011*. Available from: https://ptsldigital.ukm.my/jspui/handle/123456789/476510
[19] Programmableweb.com. Mashup dashboard [Internet]. Programmableweb; [cited 2024 Jun 6]. Available from: http://www.programmableweb.com/
[20] Jones S, Fox S. Generations online [Internet]. Pew Internet & American Life Project; [cited 2024 Jun 6]. Available from: http://pewresearch.org/pubs/1093/generations-online
[21] Cappiello C, Daniel F, Matera M, Picozzi M, Weiss M. Enabling end-user development through mashups: requirements, abstractions and innovation toolkits. In: End-User Development. Berlin, Heidelberg: Springer; *2011*. p. 9–24. Available from: https://doi.org/10.1007/978-3-642-21530-8_3
[22] Lian S, Mingdong T. API recommendation for mashup creation based on neural graph collaborative filtering. *Connect Sci. 2021*;34(1):124–38. Available from: https://doi.org/10.1080/09540091.2021.1974819
[23] Maaradji A, Hacid H, Soukane A. From service composition to mashup editor: a multiperspective taxonomy. *Future Internet. 2023*;15(2):59. Available from: https://doi.org/10.3390/fi15020059