2023

Final Year Project – Interim Report

E-LEARNING MADE EASY

YUNG CHIN PANG 3035778682
YU JIAN HUI 3035790719

Abstract

As technology develops, more and more skills are required to be learnt by developers. Developers not only have to master different skill sets, but also need to utilize a manifold of technologies and tools to produce something useful. Even though the amount of online learning resources available have been growing in the past years, we believe that the current ecosystem of learning resources lacks a systematic and efficient way to teach developers how to code. Hence, we aim to build an online e-learning platform that tackles this problem. The platform mainly provides code tutorials, quizzes and practical project guides, etc. with a focus on web developments. We intend to make the learning process as smooth as possible and that developers can see immediate results from what they have learnt. In the current stage, we have decided the implementation details and have almost completed the design stage. Next steps would be to start developing and populating the platform with written content. In the future, we plan to integrate a teacher-student system where teachers can create courses utilizing the platform's learning materials and students are allowed to enroll in them.

Contents

Abstract **1**

List of Figures **3**

1 Introduction **4**

1.1 Project Background 4

1.2 Current Problem 4

1.3 Scope and Deliverable of Project 5

1.4 Outline of the Report 6

2 Project Requirements **6**

2.1 User Requirements 6

2.2 System Requirements 7

3 Methodology **7**

3.1 Front-End Implementations 7

3.2 Back-End Implementations 9

4 Current Progress **10**

5 Challenges **11**

6 Future Plans **11**

7 Conclusion **12**

References **13**

List of Figures

Figure 1. A screenshot of the tutorials page of w3school.com 4

Figure 2. Simplified view of the flow diagram of Qwik 8

Figure 3. Schema of table profiles 9

Figure 4. Diagram of user flow with Vercel, Turso 9

Figure 5. Diagram of user flow with Vercel, Turso, Bun and QStash 10

Figure 6. Diagram of the four sections on the home page 11

Figure 7. Diagram of the login pages 11

Figure 8. Diagram of the dashboard page 12

Figure 9. Diagram of an email for account verification 12

Figure 10. A sample of the code editor. Different keywords are in different colours, demonstrating syntax highlighting in programming langauges. 13

1 Introduction

1.1 Project Background

As information technology becomes more prevalent in our daily lives, there has been increasing interests towards code-related developments by many people. For example, it is estimated that there were 26.3 million [1] developers worldwide in 2022 and it is projected to reach 28.7 million in 2024. With increase in demand of relevant skills, the supply of learning resources also surged in the past few years.

While one can study and learn required skills through official documentations, it is a tedious process due to the difficult terminologies used throughout the guides. As a result, a variety of online learning resources were made available to developers to assist in learning.

1.2 Current Problem

There are several problems regarding the current ecosystem of coding tutorial platforms and websites. Firstly, developers can easily get confused and overwhelmed by all the different technologies. Referring to figure 1, the platform w3school has 75.3 million [2] visitors per month worldwide and it is widely popular (in comparison, duolingo has 98.83 million [3] visitors per month). Yet, even for such a popular site, it simply lists out the available courses in a glossary fashion.

Figure 1. A screenshot of the tutorials page of w3school.com

Most developments, especially web developments, require a combination of several to a few dozens of different coding languages and tools. Hence, developers have a difficult time knowing what skills are required to accomplish certain goals. The problem is analogous to one learning different words and grammar of a language but not knowing how to write a story or a letter.

Secondly, many learning websites lack sufficient or high-quality exercises to help developers utilize and memorize the skills. Due to the nature of some skills such as game and web development, it is difficult to design exercises in which the solution can be checked by the platform. As such, a lack of hands-on coding experiences usually leads to a lower memory retention of learnt skill.

1.3 Scope and Deliverable of Project

We aim to design and build a code learning platform that provides coding guides and tutorials, primarily focusing on web developments. We focus on web developments since we are more familiar with it and that the problems stated above are more significant. Detailed functionalities will be explained in later sections.

The platform targets beginner and intermediate developers globally. The platform does not primarily target expert developers since they should have enough experiences on learning different skills and it is difficult for us to provide in-depth tutorials on difficult topics. As such, we will avoid including tutorials on very specific topics since it is unlikely to be useful for beginner developers.

In order to address the problem of overwhelming topics and skills needed to learn, the coding tutorials will be designed in a result-oriented fashion. It means instead of learning various topics blindly, developers would start from a goal, then break-down the problems into smaller problems and learn the skills required. With this approach, developers will have a clear idea of every stage in the whole project.

To address the problem of lack of coding exercises, project walkthroughs will be provided which developers can code alongside the steps displayed on the platform. Each project will be broken-down into multiple steps and will be designed to utilize different skills and tools.

These project walkthroughs will allow developers to grasp a feel of the capabilities of the skills and tools they are learning since they can see the result directly. They will also make the learning process efficient since developers do not need to spend time setting up their own computers.

1.4 Outline of the Report

The remainder of this report proceeds as follows. First, in section 2, we provide a detailed description of both the user requirements (functionality design) and system requirements of the platform. Then in section 3 we discuss the methodology and technical details of the project, including comparisons of different techniques and strategies. Next, in section 4, we report the current status of the project which includes the work we have done so far. After that, we present some of the challenges we faced during the project in section 5. Lastly, we discuss the upcoming plans and schedules for the remaining part of the project in section 6. A conclusion is included in the end in section 7.

2 Project Requirements

Both user requirements and system requirements will be discussed in this section. User requirements include the functionalities that we have designed while the system requirements include the constraints or expected performances of the platform, which are more technical.

2.1 User Requirements

In a high-level overview, we plan to implement a set of functionalities for the platform. Firstly, a login system will be created to manage user information. Then a database system will be designed to store and manage site content, including the courses and files. We will also design a progress tracker to record the user's progress in each coding guide and course.

Secondly, for coding exercises, we plan to write a variety of exercises including multiple choice questions, fill in the blank questions, interactive coding questions, and project walkthroughs. For courses and guides, we plan to focus mainly on providing content that is about web development while attempting to cover as many topics as possible. Comparisons and recommendations will be included when needed.

Also, a membership system will be implemented as a source of revenue. Users can subscribe to the platform to unlock some courses. After that, we also plan on creating a teacher-student platform where teachers can create their own courses and students can enroll in them, similar

to that of Moodle. Last but not least, we might implement a forum to allow exchange and discussing of ideas.

(In the actual progress report, the functionalities of each page of the platform will also be elaborated.)

2.2 System Requirements

Since the platform should be available globally, we need to guarantee the system's availability is not restricted geologically. We would also need to ensure the reliability and scalability of the database since it would be holding a large amount of data from all the users. The platform should also feel responsive and have a small initial loading time. If the platform is slow and unpleasant, less users would use it even if the courses are excellent. It is even estimated that for every one hundred milliseconds of increase in loading time, the company loses 1% in revenue. [4] Since we have limited funds, the platform should also minimize the operational cost and we might have to make trade-offs if necessary.

3 Methodology

There are two parts to this project. The first part is to design and build the platform. The second part is to populate the platform with content. We will mainly focus on discussing the first part.

Nowadays, there are multiple ways to create an e-learning platform, including writing codes by hand, using website creators such as Wix or Wordpress, or even generating codes from artificial intelligence tools. We have opted for writing codes by hand since it gives us full control of the platform's features and gains us hands-on experiences utilizing different skills.

Creating a website by writing codes can be largely separated into two parts: front-end and back-end. In simple terms, the front-end refers to technologies utilized to create and manipulate elements on a webpage which is visible to a user, while the back-end refers to technologies utilized to provide and manage the necessary data for the front-end technologies to work.

3.1 Front-End Implementations

(In the actual progress report, detailed comparisons of each technology will be presented. However, it is not shown here due to the restriction in page lengths.)

We have opted to use Qwik as the main framework to render all the pages in the platform. In

comparison to popular framework such as NextJS, Svelte or React, it ships a minimum amount of Javascript code to the user which reduces the load time of a website. It also uses proxy for reactive elements instead of states which means the components do not re-render unless it is unmounted and remounted again. Without constantly re-rendering and re-executing the functions, state management libraries such as Redux and Zustand are also not needed. Qwik also provides most, if not all functionalities other frameworks listed above provide.

The internal workings of Qwik are as follow:

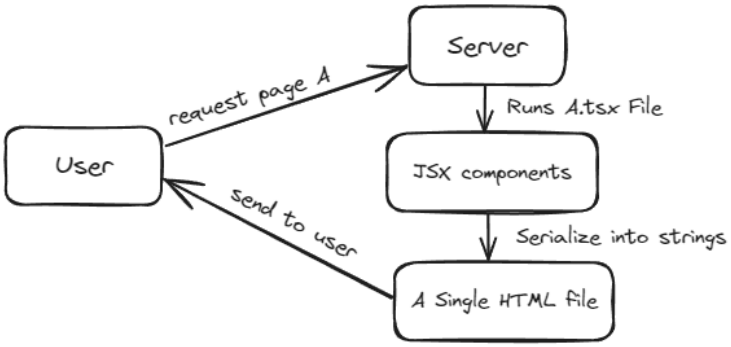

Figure 2. Simplified view of the flow diagram of Qwik

From figure 2, when a user visits a page, the server will first run the relevant typescript files belonging to the page and generate all the JSX components that should be rendered. Additional data is also fetched at this point if necessary. Then the components get serialized into strings of HTML elements which can be directly read by the browser. This HTML string is then sent back to the user and the browser will display it.

For styling, we have used Tailwind as the primary CSS library. It is similar to Bootstrap where class names are applied to style the HTML elements and no actual CSS codes need to be written. This allows for extremely fast development since writing plain CSS codes is very tedious and repetitive.

A number of other libraries are also used to achieve different purposes. One of the important one is Monaco, the primary library for our code editor. Monaco is a code editor library that includes a comprehensive list of functionalities out-of-the-box, such as code autocomplete, text formatting and searching, syntax highlighting, etc. It is also used to build one of the most popular code editor Visual Studio Code. One of the alternatives we have considered is CodeMirror which has a smaller package size of 98KB when minified (~1MB for Monaco). It offers great customizability and flexibility and we can include features that are only

required in our platform. However, it turned out to be complicated and time-consuming to customize CodeMirror and we would rather have a more well-rounded solution such as Monaco.

3.2 Back-End Implementations

To host the platform, we have chosen vercel because we can utilize its edge servers. Edge servers are servers that can run specific codes and are deployed in thousands of locations across the globe. When a user sends a request to vercel, the nearest server will respond and the latency is extremely low.

The database we have chosen at this moment is turso which is used for authentication because it hosts multiple databases across different locations. It allows users from all around the world to access the database at a low latency (tested at maximum 120ms).Turso uses postgresql, a relational database for its database and we have used a database Object Relational Mapping (ORM) library to query and update its data. Instead of writing raw SQL statements, we can use a more human-like language to access and update the records, simplifying the process.

Figure 3. Schema of table profiles

Figure 3 shows the profile schema which is accessed when a user logs in or signs up. Currently, it supports signing in through passwords, google or github Oauth.

We had originally used Redis for caching since database access before using turso is quite slow. We have since removed Redis from the tech stack.

The flow of a user request to the servers is shown below.

Figure 4. Diagram of user flow with Vercel, Turso

As shown in figure 4, when a user signs up, Vercel server will send a database update to Turso via the authentication library Lucia. Then Turso will send back the user profile and Lucia will create a session for the user.

However, even though Vercel Servers are available globally, it has very low computational power and computational intensive tasks need to be off-loaded to another server. As such, we have set up another server that currently serves two purposes: password hashing and integrating with a message queue service.

The server runs in Bun runtime which is in Javascript and much faster than Node JS. Just like how we would use express or other frameworks in Node JS, we have used Elysia JS to handle the server requests which is an extremely performant library. For password hashing algorithm, we have opted for the standard argon2id algorithm which is a balance between security and performance.

Besides hashing password, we will also use Bun for sending emails to users. Currently, a verification email will be sent to users once they sign up with a password. It may seems like a trivial process to send an email to users but messaging and notification systems are fragile and tend to malfunction quite often. There are many cases where users are not able to receive messages due to many reasons and we need a comprehensive solution to tackle the problem, which is why we employed a message queue (Qstash). The message queue will automatically manage all the notifications/emails to be sent and re-send them on failures. It also stores a record of all messages as backup which is convenient.

The whole backend flow looks like below after integrating with Bun:

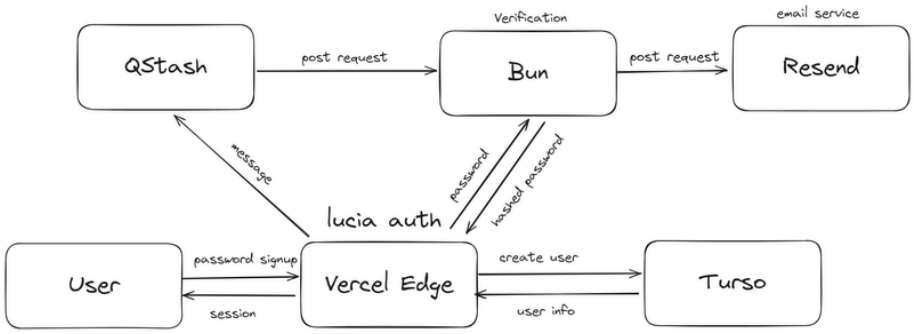

Figure 5. Diagram of user flow with Vercel, Turso, Bun and QStash

As seen from figure 5, Vercel will send a message to Qstash which will send an email request to the Bun server, which then in turn will post to the external email service. QStash will try to resend the request to Bun whenever there is a failure, making the system much more reliable.

4 Current Progress

The UI design of the platform has been partly finished, including the homepage, login pages, dashboard page, and email for account verification.

The homepage has been designed with a combination of colors to create an appealing appearance. It is divided into four sections, each illustrating what the platform can offer and how it can assist the students in learning (see Figure 6).

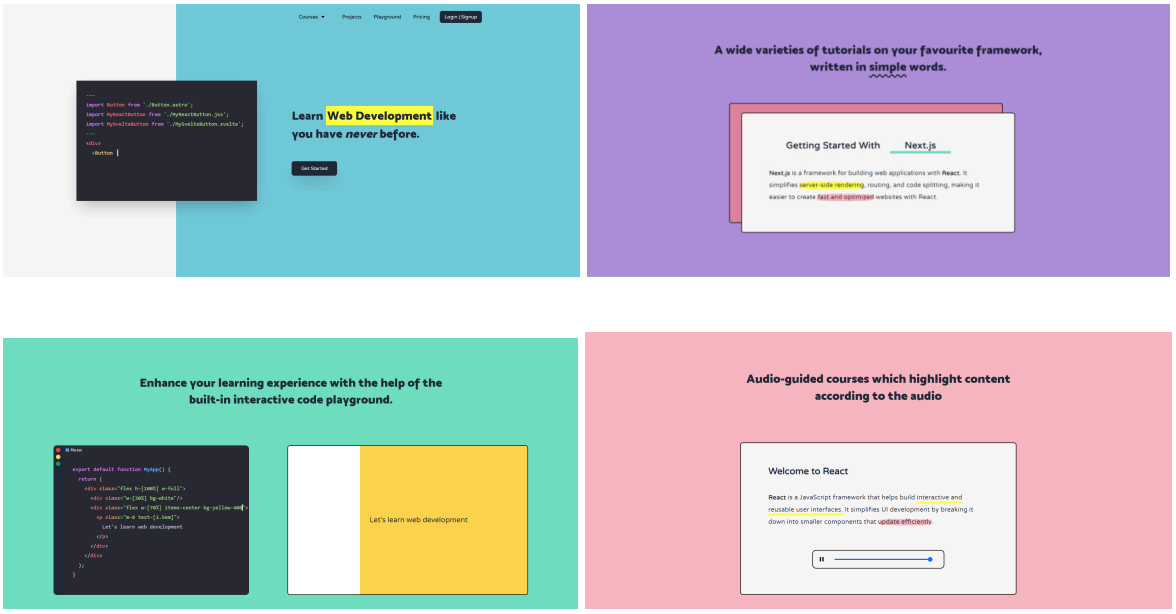

Figure 6. Diagram of the four sections on the home page

When users click on the "Login | Signup" button on the home page, they will be navigated to the login page. For new users, they can login with their Google account or Github account, or they can sign up using their personal email address (see Figure 7).

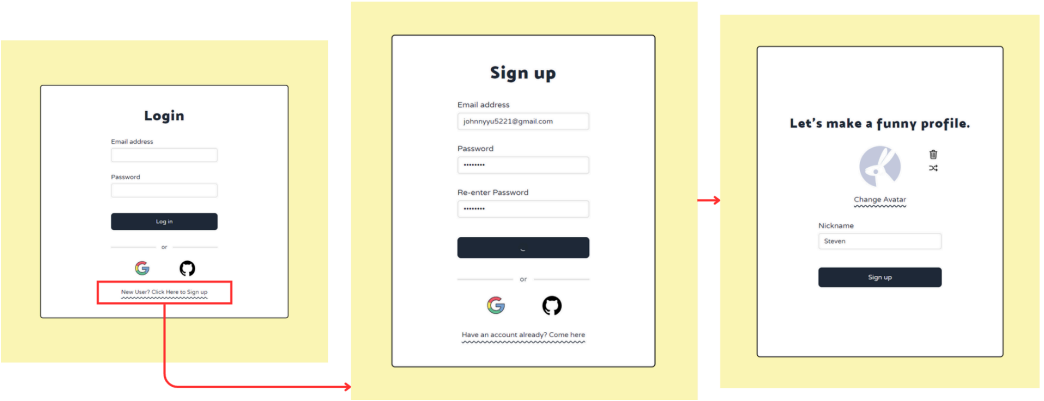

Figure 7. Diagram of the login pages

Once users have successfully logged in, the users will be navigated to the dashboard page. On this page, users can view all the registered courses and projects, and monitor their learning progress (see Figure 8).

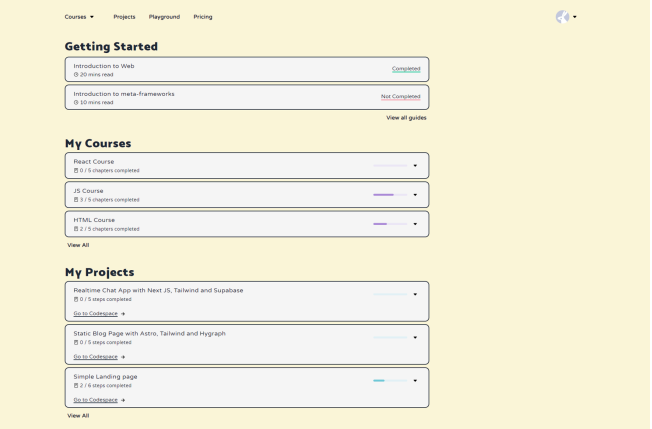

Figure 8. Diagram of the dashboard page

Before new users can register for courses or projects, they will be asked to verify their email address. The platform will send an email to the user. By clicking on the yellow button in the email, the user can gain access to the registration process (see Figure 9).

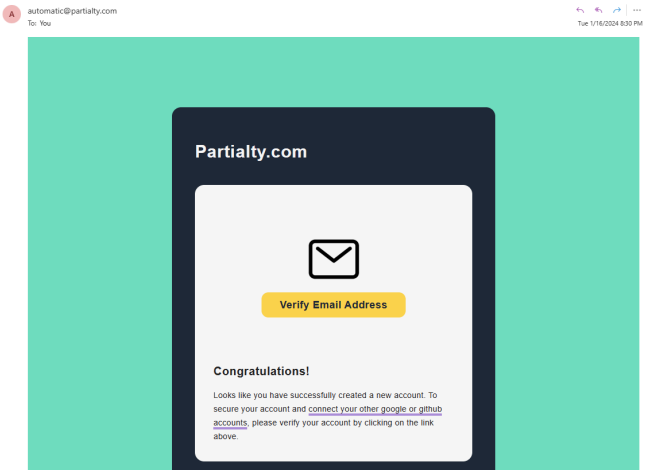

Figure 9. Diagram of the email for account verification

Regarding the functionality implementation, Upstash Redis was integrated to cache the user's session information returned from supabase. When a user visits his/her own dashboard, the platform will first look for the user's session data in Upstash Redis before authenticating through supabase. This reduces latency and makes the platform more responsive.

A code editor utilizing Monaco library was implemented. Currently, it can open a file and edit the file content with code intellisense and syntax highlighting.

```
1  const express = require('express');
2  const app = express();
3  const port = 3111;
4
5  app.use(express.static("public"))
6
7  app.get('/', (req, res) => {
8    console.log("dsajdklsajkdlas")
9    res.send('Welcome to a WebContainers app! 🎉');
10 });
11
12 app.listen(port, () => {
13   console.log(`App is live at http://localhost:${port}`);
14 });
```

Figure 10. A sample of the code editor. Different keywords are in different colours, demonstrating syntax highlighting in programming langauges.

Also, the ability to run code directly in the user's browser is implemented by utilizing Webcontainer api from Stackblitz. We can run program code inputted by the users directly in users' machines without needing to run the code in another dedicated server. Not only does it reduce costs, the delay is also minimized and the experience is smoother.

In addition to using vercel servers, we have also routed the platform through cloudflare. It acts as a gateway to our back-end services as it handles all the incoming traffic. Cloudflare can detect traffic anomalies and balance the loads across multiple servers in case of high traffic volume. Most importantly, it can cache most of the files generated from vercel and reduces the loading time of the platform, especially if images or videos are present.

Currently, the design of the platform is almost complete and we can begin coding shortly. The original plan was to design every part in detail so we can focus on coding afterwards. However, we have yet to design the student-teacher system and the theming of code editor is also not thought of. Moreover, we are not satisfactory with some of the tools we are using. For example, Turso database could be a better alternative to Supabase because the free tier includes more storage but it is difficult to integrate. Overall, more tweakings are needed in the future to achieve better results.

5 Challenges

One of the biggest challenges we faced so far is actually the first problem mentioned in section 1.2: there is an overwhelming number of technologies and possible solutions to build a website that are presented to us. Choosing the suitable methodologies and technologies took us more time than implementations. Yet, there might be an even better solution that is more suited for our use case available and we can improve the platform yet again. For now, we are

putting this problem aside since it makes more sense to progress more rather than making everything perfect at the start. We will keep paying attention to new technologies and alternatives while coding the platform and make changes near the end if needed.

6 Future Plans

After creating the basis of the platform in March, we plan to implement a teacher-student system where the teachers can create courses according to their preferences. They might choose to incorporate part of our platform's courses into their own teaching materials. Students can then enroll the courses and complete tasks assigned by the teachers.

Moreover, to increase the interactivity of the courses, we plan to create a voice-over for the course materials in April. The voice-over can replace some of the text and words in the course materials and we can display the information in bullet-points instead. The voices should be mostly artificially generated and should be high clarity after some tuning.

Another plan is to create a recommendation system through data training algorithms and artificial intelligence. We can display the recommended courses to a user in their dashboards. However, we have not set a specific date since we are still accessing the feasibility of it.

7 Conclusion

This project aims to create an e-learning platform that tries to solve some of the problems a beginner and intermediate developer might face during learning. For example, they might get overwhelmed by all the technologies and options available to them and struggle to combine all the skills learnt to produce something useful. Most e-learning platforms also lack sufficient and high-quality coding practices which are crucial in learning and retaining programming skills. The platform should provide learning processes that are efficient and result-oriented while being pleasant and fun. It should also have an abundant number of exercises that come in different varieties to let developers practice their skills. Currently, the front-end of the platform is mostly powered by Qwik and styled by Tailwind while the back-end utilizes Vercel servers, Upstash redis and Supabase. A user sign-up and login system, code editor and code execution environment were developed and the website is being routed through cloudflare to cache most of its content. Landing page, login page, dashboard and other important pages were designed as well. It is still too early to judge whether the platform has achieved its goal since no actual content has been curated yet. However, after polishing the content of the platform, we plan to enrich it with even more features such as teacher-student platform and voice-over for coding tutorials.

References

[1] Grid Dynamics, "How Many Software Developers Are in the World?," 8 8 2023. [Online]. Available: https://www.griddynamics.com/global-team-blog/number-software-developers-world. [Accessed 23 10 2023].

[2] SEMRUSH, "w3schools.com Web Traffic Statistics," 21 10 2023. [Online]. Available: https://www.semrush.com/website/w3schools.com/overview/. [Accessed 23 10 2023].

[3] SEMRUSH, "duolingo.com Web Traffic Statistics," 21 10 2023. [Online]. Available: https://www.semrush.com/website/duolingo.com/overview/. [Accessed 23 10 2023].

[4] K. M. Androutsos, "How much does a slow website cost you?," 26 11 2021. [Online]. Available: https://www.papaki.com/blog/en/2021/11/26/a-slow-website-costs-you/#:~:text=So%2C%20every%20year%20for%20a,the%20traffic%20up%20to%2020%25.. [Accessed 25 10 2023].