JPEE 실천공학교육논문지 JOURNAL OF PRACTICAL ENGINEERING EDUCATION

실천공학교육

J. Pract. Eng. Educ. 16(3), 267-274, 2024

# Creating a Standardized Environment for Efficient Learning Management using GitHub Codespaces and GitHub Classroom

**Aaron Daniel Snowberger[1], Kangsoo You[2]***

[1]Department of Computer Education, Jeonju National University of Education, Jeonju 55101, Korea
[2]School of Liberal Arts, Jeonju University, Jeonju 55069, Korea

**[ Abstract ]**

One challenge with teaching practical programming classes is the standardization of development tools on student computers. This is particularly true when a complicated setup process is required before beginning to code, or in remote classes, such as those necessitated by the COVID-19 pandemic, where the instructor cannot provide individual troubleshooting assistance. In such cases, students who encounter problems during the setup process may give up on the class altogether before even beginning to code. Therefore, this paper recommends using GitHub Codespaces as a tool for implementing standardized student development environments from day one. Codespaces provides Docker containers that an instructor can configure in such a way as to enable students to practice installing various coding tools within a controlled space, while also providing a language-specific, fully optimized development environment. In addition, Codespaces may be used more effectively in collaboration with GitHub Classroom, which helps instructors manage both the starter code and coding environment in which students work. In this paper, we compare two semesters of university Node.JS programming classes that utilized different development environments: one localized on student computers, the other containerized in Codespaces online. Then, we discuss how GitHub Codespaces and GitHub Classroom can be used to increase the effectiveness of practical programming classes while also increasing student engagement and programming confidence in class.

**Key Words:** GitHub classroom, GitHub codespaces, Docker containers, Integrated development environment, Programming education

## I. Introduction

One of the challenges in teaching practical programming courses is the installation and standardization of development tools across various student devices. While many schools provide computer labs for classroom use with software pre-installed, students also use personal devices in the computer lab and for homework and need to install the software individually. Helping students get their coding environments configured often requires dedicating at least one or two lectures at the beginning of the semester on setup and troubleshooting. However, different installation steps may be required for multiple student devices running different operating systems.

On Windows, for example, many program installers don't automatically update the $PATH environment variable, which must then be updated manually in the Windows settings. Mac and Linux also may recommend using command line tools for program installation. However, none of these processes are usually easy for beginner

**http://dx.doi.org/10.14702/JPEE.2024.267**



**Received** 9 May 2024; **Revised** 29 May 2024
**Accepted** 21 June 2024

***Corresponding Author**
E-mail: gsyou@jj.ac.kr

programmers to understand, and the complexity of these processes often increases the amount of time an instructor spends helping students simply get ready for programming. Therefore, standardizing the entire installation process within a virtual or containerized programming environment provides the best method to quickly get up and running. To that end, this paper recommends GitHub Codespaces [1] (i.e. Codespaces), a configurable Docker-based containerized programming environment that can be customized to a specific class. In addition, Codespaces can be utilized very effectively with GitHub Classroom [2] (i.e. Classroom), an assignment repository management platform, which can be used to distribute and collect student assignments in GitHub.

In this paper, we describe how Codespaces can be used in collaboration with Classroom to increase the effectiveness of practical programming education while increasing student engagement and programming confidence in class. We also compare the use of the fully online environment provided by Codespaces that includes an online implementation of both git and Visual Studio Code (VS Code) with a localized environment in which both a git client and a local Integrated Development Environment (IDE) must be installed.

## II. Related Studies

The concern about standardized student coding environments has appeared in research literature from at least 2010 [3-5], when virtual machines (VMs) started being used in programming classes. In particular, Harvard university's introductory programming class CS50 was one of the first to take this approach [5]. CS50 has undergone significant changes to optimize the standardized student coding experience [6-8], from an on-campus cluster in 2007, to an off-campus cloud in 2008, to client-side VMs like VM Player in 2011, to containerized environments based on Docker in 2015, and finally to GitHub Codespaces when it was launched in 2021. This most recent implementation of CS50, including VS Code extensions and the CS50 Docker image, `cs50/codespace`, is currently available at https://cs50.dev/. Accessing the CS50 link requires a GitHub login, after which a customized CS50 Codespace is created within the logged in account, as shown in Fig. 1.

Although the CS50 Codespace provides a standardized container-based coding environment and plugins required for the Harvard class, it is lacking assignment starter code. This is where GitHub Classroom can help. Classroom was launched in 2015 but began to take prominence during the COVID-19 pandemic. This online classroom management platform helps instructors distribute, collect, manage, and organize student repositories that include starter code and a customized Codespace for each assignment. Student repositories are forked into a student's account from a template repository in the instructor's account and linked

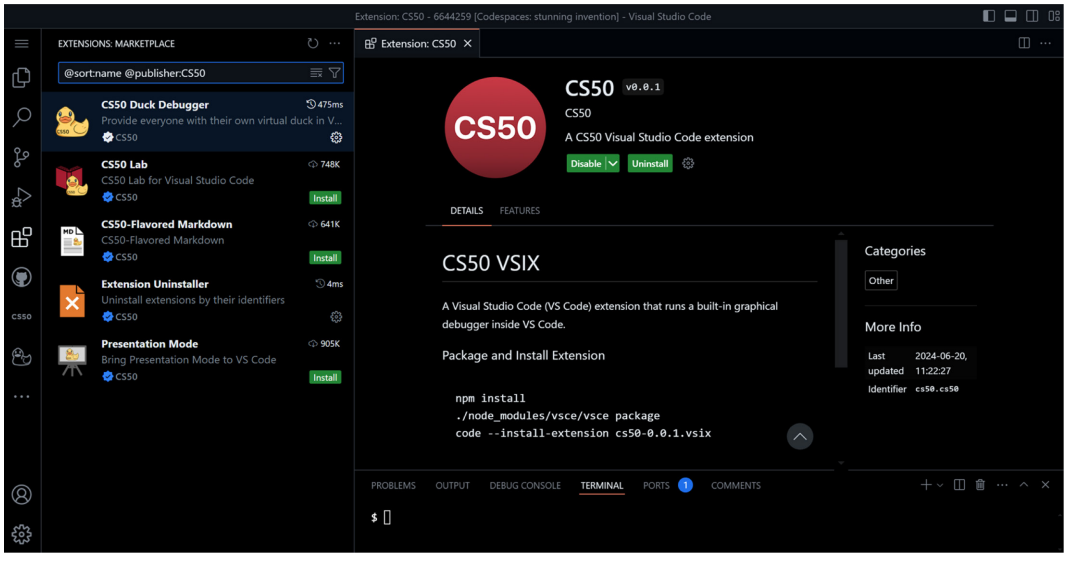


**Fig. 1.** The CS50 Codespace with a handful of custom VS Code extensions to support the class.

in the admin view of Classroom. In this case, the only thing an instructor needs to do, apart from creating the template repository, is share the Classroom link that will fork the repository. An effective method for sharing the assignment link is described in another paper [9] that describes how Classroom can be used with GitHub Pages [10] to create a simple classroom website for student access to lecture notes and assignments.

## III. Comparison of Codespaces and Local Development Environment

In this research, GitHub Classroom was used in four university classes, over two semesters, as a means to distribute and collect students programming assignments. In two of the classes, GitHub Codespaces was used to support a Node.JS web backend programming environment. Therefore, all programming and git commits were handled online in the Codespace itself. In the other two classes, a local Node.JS programming environment was created on student computers. This necessitated the installation of git, Node.JS and related tools, and an IDE, such as the Desktop edition of VS Code. In addition, in the localized environment classes, GitHub Desktop [11] was installed to help students download starter code, and reupload their assignment submissions. This was done to avoid conducting additional lectures on the git Command Line Interface (CLI).

Over the course of a semester, students completed a total of 15 programming assignments. These included 12 regular programming assignments, a final project, a midterm test, and a final test. As there were more than 25 students in each class, by the end of each semester, Classroom had distributed and was managing almost 300 separate student repositories per class, around 1,200 repositories in total. Various pros and cons to both the online Codespaces environment and the localized development environment became apparent as each class progressed, and these will be analyzed in greater detail below. To begin with, Table 1 provides an overview of the two different development environments and software required for each.

### A. Codespaces Coding Environment

As the above table indicates, the localized development environment requires the installation of multiple software tools, whereas the Codespaces environment requires no setup apart from logging in to access the Codespace. Both GitHub Free and GitHub Pro accounts have a limited amount of free storage and core usage hours per month. Free accounts receive 15 GB of storage and 120 core hours per month, and Pro accounts receive 20 GB of storage and 180 core hours per month [12]. Verified GitHub Education accounts (using a school email address) are effectively upgraded to Pro accounts, providing the larger Codespaces capacity. However, both Free and Pro options provide more than enough storage and usage hours for students attending a three-hour per week lesson, including homework time, especially given that the entire lecture period is not spent in Codepsaces. In addition, instructors with verified Education accounts can enable Codespaces for their GitHub organization and select Codespaces as a supported editor for student assignments in Classroom [13].

In the Codespaces classes, when a student accesses a Classroom link and forks the template repository, the green [Code] button to the upper-right of the repository provides an option to “Create a new Codespace” in that repository. By default, Codespaces are limited to two

**Table 1.** Overview of software required for Node.JS backend development in both Codespaces and a localized development environment

| Software | Codespaces online environment | Localized environment |
|---|---|---|
| Version Control | git (built-in) | git (local installation) + GitHub Desktop |
| JavaScript Runtime Environment | Node.JS (built-in) | Node.JS (local installation) |
| Database | MongoDB Atlas (online) | MongoDB Compass + MongoDB Shell |
| IDE | VS Code (built-in) | VS Code (local installation) |
| Terminal | VS Code’s built-in Terminal tool | VS Code’s built-in Terminal tool |

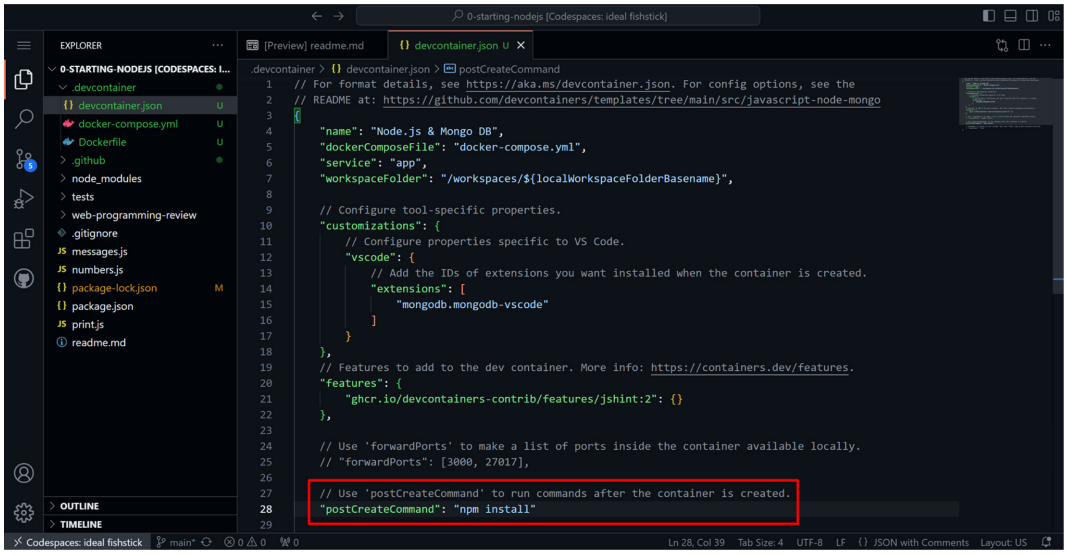


**Fig. 2.** A newly created `.devcontainer` file that contains standardized environment setup commands, including the MongoDB extension, JSHint feature, and `npm install` postCreateCommand.

active instances in a Free or Pro account at a time. These instances are automatically powered down after 30 minutes and automatically deleted after 30 days [14] of inactivity, although both settings can be adjusted in an account's personal settings page. However, it is good practice to remember to power down the Codespace at the end of each class to avoid overrunning the free capacity limits.

The configuration of a Codespace is managed with a `.devcontainer` file which can be customized to suit a given starter code template or coding environment. For example, various features like JSHint and extensions like MongoDB can be pre-installed in the environment, and the process of running `npm install` after Codespace creation can be automated in the `.devcontainer` file. Fig. 2 displays a sample of this type of configuration file.

A `.devcontainer` file like the one shown above is not necessary for the creation of a Codespace. However, it does provide additional environment customization features like those shown in the CS50 Codespaces example in Fig. 1. In this study, a customized `.devcontainer` file was not created and each student Codespace was run with the default settings. In the case of using MongoDB, the cloud-based MongoDB Atlas [15] was used instead of the local installation of MongoDB Compass, which requires additional setup if attempting to access it from Codespaces.

## B. Localized Coding Environment

In the local environment classes, with the exception of MongoDB Compass and MongoDB Shell, all the programs listed in Table 1 needed to be installed at the beginning of the class. This installation process, including an overview of assignment access and submission with GitHub Desktop, required an additional lecture. When MongoDB Compass and MongoDB Shell were installed halfway through the course, a second additional half-lecture was required.

When accessing a Classroom link to fork the template repository, students in the localized environment selected the "Download with GitHub Desktop" option in green [Code] button to the upper-right of the repository. Students need to authenticate with GitHub Desktop before downloading or resubmitting (uploading) assignment files. The process is straightforward on personal devices, but on shared computers, merge conflicts and accidentally overwriting another student's code overwriting are always possibilities. To avoid this, students must first logout of the previous account, and then login to their personal account, before accessing their assignment code. Additionally, it is recommended to use two separate download directories on the local computer for each student to avoid merge conflicts or overwriting another student's localized code. Such issues

will be addressed again in the Evalution and Analysis section that follows.

### C. GitHub Classroom for Assignment Access and Grading

In both the Codespaces environment classes and the localized environment classes, student assignments were managed by linking student GitHub account IDs with their names and student IDs from a manually input Class Roster provided by the university. In both types of classes, students accessed their assignments in the same way, with a "fork repository" link. These links need shared with students, and can be done easily with a class website built with GitHub Pages [9]. However, assignment submission and teacher grading are handled outside of Classroom.

When submitting assignments in the Codespaces classes, the implementation of git in the Codespaces VS Code IDE automatically tracks all changes that are made to the code. Deleted files and lines of code are marked in red, modified files are marked in yellow, and newly created files and folders are marked in green. After completion of a coding assignment, the git icon in the left sidebar highlights the number of changed files. When students click the icon, they enter a commit message and push the code back to the assignment repository. Submitting assignments from the localized environment is similar, except that the local installation of VS Code does not track file changes, but rather GitHub Desktop does. After completion of a coding assignment, students enter a commit message in GitHub Desktop and push the code back to the assignment repository.

When grading assignments, instructors have two options for running student code. The first option is to open the assignment code in a Codespace. Assignments created in the student Codespaces environments will work properly when accessed by the instructor in the same way. However, code that was created in the localized environment may contain `localhost` web server links and may not work properly in a Codespace. The second option is for an instructor to install the GitHub CLI and the GitHub Classroom CLI Extension to use the Classroom download link available in each assignment page. Running the CLI code provided by the Classroom download link, shown in Fig. 3, will automatically download all student repositories for that assignment into the folder the command is run from. The same is true for both Codespaces and localized environments, and the code from both environments should function properly in the instructor's localized environment.

## IV. Evaluation and Analysis

Various pros and cons became apparent when teaching

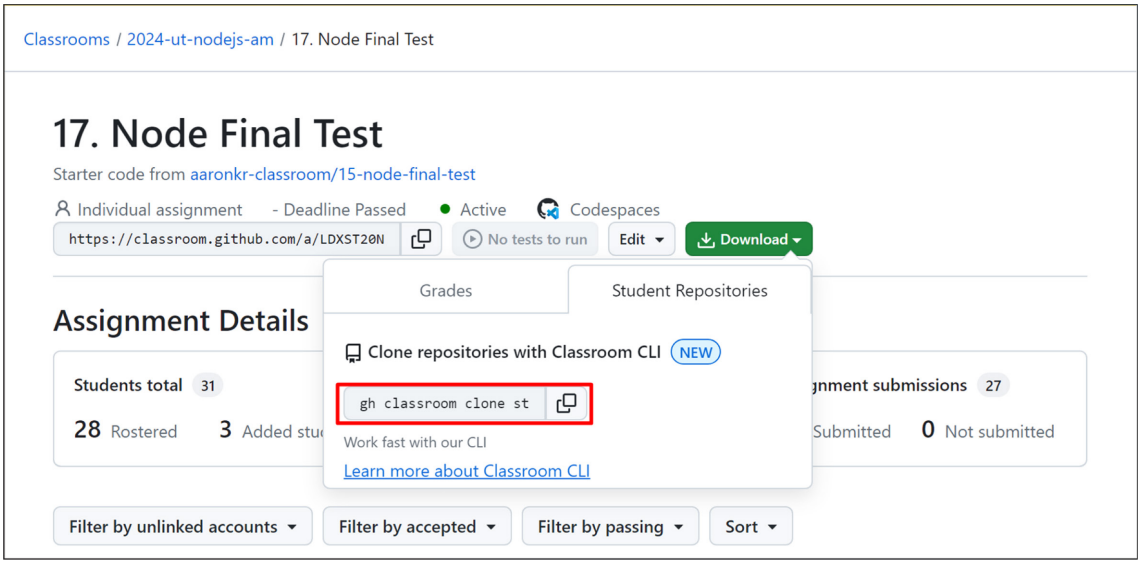


**Fig. 3.** A GitHub Classroom CLI student repository download link.

courses using both the Codespaces and localized environments. These will be explained in more detail below, but can largely be divided into technical problems and troubleshooting, assignment submissions, and offboarding students to their own devices at the conclusion of the class.

First, environment-based technical problems and troubleshooting were greatly reduced in the Codespaces environments compared to the localized environments. In localized environments, for example, various installation problems arose with nearly every new program installation. Such problems included students downloading the wrong installer, the wrong version number, or an archive file instead of the installer; students selecting the wrong settings during the installation process, which required a reinstall or manual settings adjustment; installer programs failing to update the Windows $PATH variable; and students failing to run the correct CLI commands after installation, for example to run the MongoDB server or install NPM packages. Additionally, in the localized environment classes, an additional lecture was required at the beginning of the semester just to set up the initial coding environment.

On the other hand, in Codespaces, all required program installations, versions numbers, and post create commands can be set by the instructor in advance with either the default Codespace settings or a `.devcontainer` file. This reduces the confusion and setup time for students, and more course material can be covered. In the Codespaces classes, an additional two lectures of course material was able to be covered compared to the localized environment classes.

Second, assignment access and submission in the localized environment classes quickly became a challenge when two students used the same computer, such as in morning and afternoon classes. For example, the morning students usually had no trouble accessing and submitting assignments, but if the afternoon students forgot to logout of GitHub Desktop and re-login with their own accounts, merge conflicts or accidental overwriting of the previous student's code became common occurrences. In many cases, students using shared computers opted to *not* share the computer. In other words, if a morning student used the computer lab computer, the afternoon student chose to bring their own device, and vice versa.

On the other hand, in the Codespaces environment, all student assignment commits and submissions are handled in their individual Codespaces, so merge conflicts and accidental code overwrites are exceedingly rare. Even in such cases that such a problem occurs, it is only a merge conflict or code overwrite with the student's own code, not someone else's.

Third, a possible advantage to the localized coding environment classes is that because students had already struggled through the installation and setup process for each required technology, offboarding them to their own devices after the class concluded was mostly unnecessary. However, students using Codespaces did not already have experience installing and setting up localized versions of the programs, and were ill-prepared to do so at the end of the course. In any case, Codespaces students are still able to continue coding in Codespaces after the class concludes, so long as they don't exceed their Codespace capacity. Or, if they desire to learn how to set things up on their own devices, a few well-written online tutorials should be enough to help them do so. Even so, it may be beneficial at the end of the course to provide Codespaces students an additional video lecture or a list of curated tutorial links to help them setup their own devices if so desired.

## V. Conclusion

In this paper, two semesters of four university level Node.JS backend programming classes were compared. Two classes used a localized coding environment that required an additional almost two class lectures to setup and troubleshoot. Additionally, git merge conflicts and accidental overwriting of other students' code on shared devices were other downsides to this approach. The second two classes used a standardized, containerized, cloud-based Codespaces environment. In these classes, two additional lectures of course material were able to be covered due to the reduced amount of necessary setup and troubleshooting time. Additionally, student feedback for Codespaces was quite positive, with most students expressing a greater enjoyment and confidence in coding when less technical issues arose.

In conclusion, Codespaces presents an effective method to create standardized, accessible-from-anywhere, cloud-

based coding environments easily on a per-assignment basis. This reduces difficulties for both students and teachers, it enables teachers to cover more relevant course material, and it bolsters student engagement and enjoyment in the class, as well as confidence in coding. At the same time, GitHub Classroom is an effective tool to distribute Codespace template starter code to students, and also help teachers better organize and grade assignment submissions. In this study, GitHub Classroom was used to manage over 1,200 student assignment repositories containing submissions for 15 different assignments in each class.

## References


[1] GitHub Codespaces, https://github.com/features/codespaces.

[2] GitHub Classroom, https://classroom.github.com.

[3] O. Laadan, J. Nieh, and N. Viennot, "Teaching operating systems using virtual appliances and distributed version control," in *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, pp. 480-484, 2010. https://doi.org/10.1145/1734263.1734427.

[4] D. Malan, "Moving CS50 into the cloud," 2010, in *15th Annual Conference of the Northeast Region of the Consortium for Computing Sciences in Colleges*, 2010. https://www.researchgate.net/publication/42246066_Moving_CS50_into_the_Cloud.

[5] D. Malan, "Reinventing CS50," in *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, pp. 152-156, 2010. https://doi.org/10.1145/1734263.1734316.

[6] D. Malan, "Containerizing CS50: Standardizing Students' Programming Environments," in *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024*), July 8-10, 2024, Milan, Italy. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653567.

[7] D. Malan, "Standardizing Students' Programming Environments with Docker Containers: Using Visual Studio Code in the Cloud with GitHub Codespaces," in *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education,* vol. 2, pp. 599-600, 2022. https://doi.org/10.1145/3502717.3532164.

[8] D. Malan, J. Carter, R. X. Liu, and C. Zenke, "Providing Students with Standardized, Cloud-Based Programming Environments at Term's Start (for Free)." in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education,* vol. 2, pp. 1183, March 2023. https://doi.org/10.1145/3545947.3569611.

[9] A. Snowberger and C. H. Lee, "A workflow for practical programming class management using github pages and github classroom," *Journal of Practical Engineering Education*, vol. 15, no. 2, pp. 331-339, 2023. https://doi.org/10.14702/JPEE.2023.331.

[10] GitHub Pages, https://pages.github.com.

[11] GitHub Desktop, https://desktop.github.com.

[12] GitHub Docs, "About Billing for GitHub Codespaces," Accessed June 20, 2024. https://docs.github.com/en/billing/managing-billing-for-github-codespaces/about-billing-for-github-codespaces.

[13] GitHub Docs, "Using GitHub Codespaces with GitHub Classroom," Accessed June 20, 2024. https://docs.github.com/en/education/manage-coursework-with-github-classroom/integrate-github-classroom-with-an-ide/using-github-codespaces-with-github-classroom.

[14] GitHub Docs, "Understanding the Codespace Lifecycle," Accessed June 20, 2024. https://docs.github.com/en/codespaces/getting-started/understanding-the-codespace-lifecycle.

[15] MongoDB Atlas, https://www.mongodb.com/products/platform/atlas-database.

**Aaron Daniel Snowberger**_Regular Member

2006 : University of Wyoming, USA, Computer Science, B.S. degree
2011 : Full Sail University, USA, Media Design, M.F.A. degree
2024 : Hanbat National University, Korea, Information & Communications Engineering, Ph.D.
2010 ~ 2023 : Jeonju University, School of Liberal Arts, Professor
2023 ~ Present : Jeonju National University of Education, Dept. of Computer Education, Lecturer
〈Research interests〉 Computer Vision, Natural Language Processing, Image Processing, Machine Learning, and Software Education

**Kangsoo You**_Regular Member

August 2005 : Jeonbuk National University, Dept. of Image Engineering, Ph.D.
March 1996 ~ August 2006 : Jeonju University, School of Liberal Arts, Visiting professor
September 2006 ~ Present : Jeonju University, School of Liberal Arts, Professor
〈Research interests〉 Image Processing, Computer Education, SW/AI Education, Data Science Education, Robot Utilization Education