SyncLink - Building An Expanded Cloud IDE Platform

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Charles Fang
Spring 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Technical Reviewer: Aidong Zhang, Department of Computer Science

Technical Reviewer: Matthew Dwyer, Department of Computer Science

Abstract

During the COVID-19 pandemic, the vast majority of in-person education transitioned to remote learning. The U.S. Census Bureau (2020) found that “nearly 93% of people in households with school-age children reported their children engaged in some form of ‘distance learning’ from home.” A significant drawback of remote education was the difficulty of providing and effectively using office hours. Remote office hour methods such as sharing the screen do not scale well with complex and lengthy programs.

SyncLink was designed to support remote collaboration among computer science students and help bridge the virtual gap with the teaching staff. Additionally, it can serve as a powerful tool for demonstrating introductory-level CS topics to students. This tool is a Cloud IDE platform with additional features such as virtual whiteboards, code editor embedded markup rendering, and a connected mobile scanner app. If successfully developed, SyncLink has applications in student collaboration, office hours, job interviews, and conceptual teaching.

Introduction

As this technical report was being written, the largest pandemic seen in nearly a century is interrupting our daily lives. In early 2020, various countries went through drastic shifts in lifestyle as the world became locked down due to COVID-19. Education systems were thrown into chaos as they were asked to transition from in-person learning to remote on short notice.

At the start of the pandemic, I was a Third Year student in UVA's Computer Science program. It was a difficult semester. One of the hardest classes I ever took was CS 4414: Operating Systems. There were many memories of the long hours spent grinding away in the office hour room at Rice Hall. Despite the struggles I faced with comprehending certain assignments, it was comforting to know that I had the opportunity to speak with teaching assistants and professors to receive help. This help came in many forms. On some days, it was an explanation drawn out on the classroom whiteboard with useful visualizations. On other days, a TA took a quick peek at my code and pointed out a misconception.

Many students depend on getting help from teaching assistants (TAs) and professors during office hours. I believe that it is a vital aspect of the educational process. In addition to trying to get a better understanding of recently taught concepts, students often need help with debugging their programs. A significant drawback of remote education is the difficulty of providing and effectively using office hours.

Nowadays, without access to a tablet, communicating a visual explanation is extremely tricky. The process of drawing on paper, taking a photo, and uploading is both tedious and time-consuming. In many of my classes, remote office hour debugging has consisted of sharing the screen and scrolling when asked to. Naturally, this method is not particularly effective when you have hundreds of lines of code to parse and ends up bottlenecking the entire process.

Additionally, screen share resolution issues made it difficult for TAs to read code and see where the cursor was placed.

SyncLink is a software tool proposal designed to assist with adjusting to remote office hours and code reviews. At its core, it is a Cloud IDE platform with a mobile app and other modules to facilitate the connected nature of software development.

Background

Traditionally, software development is done on local machines using Integrated Development Environments (IDEs) such as Eclipse or VS Code. Features such as syntax highlighting, code compiler, debugging tools, and hierarchy diagrams help increase productivity (Veracode, n.d.). As an individual developer, it is relatively simple to keep track of your own work. However, your work will only be saved locally unless you use online version control tools like GitHub or remote backups.

As software projects become increasingly complicated and widen in scope, the project team will likely consist of several members. A crucial aspect of software development is the idea of peer reviews. In a peer code review, other programmers check each other's code for mistakes and areas of improvement. Studies have shown that this practice can catch 60% of coding defects (Shull et al., 2002). When a student visits office hours, the process is not entirely unlike a code review.

To better understand the current status of UVA office hours, I spoke with a teaching assistant for CS 1111: Introduction to Programming about the difficulties and drawbacks of remorse learning in their class. She pointed out that many CS concepts are hard to understand unless you visually describe them. There's a lot of abstraction that can lead to confusion for new

students. After the transition to virtual learning, one of the “hardest things about being a TA is being unable to easily explain concepts visually.” In a more normal semester, the TAs would gather students around the classroom whiteboard and draw out an explanation of the topic. An important point that the TA stressed to me was that the most beneficial part was not the final image. It was the process of showing the student each individual step as you draw it. By looking at their reaction to your explanation, you can gauge their understanding.

In CS 1110, students are assigned pair programming labs on a weekly basis. Under this model, one partner will be the “driver” controlling the mouse and keyboard and the other partner will be the “navigator” examining the progress that is being made. The two roles should be switched every 15-20 minutes. The TA noted that students were having difficulty switching roles virtually. With in-person labs, the students could simply swap laptops and continue their work. Traditionally, having multiple developers work on the same project is simplified with the use of version control programs such as GitHub. With the wide variety of skill levels of students in introductory programming courses, this was not a viable solution because version control is only introduced to the curriculum in higher-level classes like Advanced Software Development. As such, many students resorted to primitive solutions such as emailing code or migrating to Google Documents during each and every switch. This time-consuming process is prone to errors and reduces the amount of time spent learning material.

Related Work

The discussion on related work will begin with Google Documents. To summarize, it is an online word processing software that is freely available as part of Google’s Office Suite. This software can be accessed out-of-the-box as an online web application or through downloadable

apps on mobile/desktop devices. Although it is not suitable for direct use as an IDE, there are several features from it that I would like to see implemented in SyncLink. To begin, users can collaboratively edit documents together in real-time with other users. Individual user's cursors and highlighted text are visible to everyone else. Additionally, any changes to the text are logged and viewable through the document history. Users can be invited to edit specific documents and there are tiered view/edit/share permissions. Part of its charm is that Google Documents is extremely quick to set up and easy to use.

In recent years, existing IDEs have begun to roll out modules that allow for collaborative editing. As an example, Atom's teletype program (2017) is currently in its beta phase and allows multiple users to work on the same code in real-time. The popular IDE Visual Studio Code's marketplace released an extension called "Microsoft Visual Studio Live Share" in 2018 which serves a similar role on their platform. Although these programs are extremely accessible, the requirement of the user downloading and installing these extensions still serves as barriers to entry. Additionally, they cannot completely fill the role of SyncLink. SyncLink is more than just a synchronized IDE. Its ability to cleanly integrate text, images, and other elements with programming language code sets it apart. That being said, this idea of integrated elements is not a novel feature of the technology landscape.

My initial ideas for SyncLink were drafted during Fall 2020. In early 2021, I used Google Colab and Jupyter notebooks in my Computer Vision course for the first time and realized there was significant overlap in the features. Jupyter notebooks are interactive documents that allow for the blending of live code, visualizations, images, and narrative text. With the official support of over 40 languages including, but not limited to, Julia, Python, R, and MATLAB, Jupyter notebooks fill a similar niche that SyncLink's embedded notes module was designed to capture.

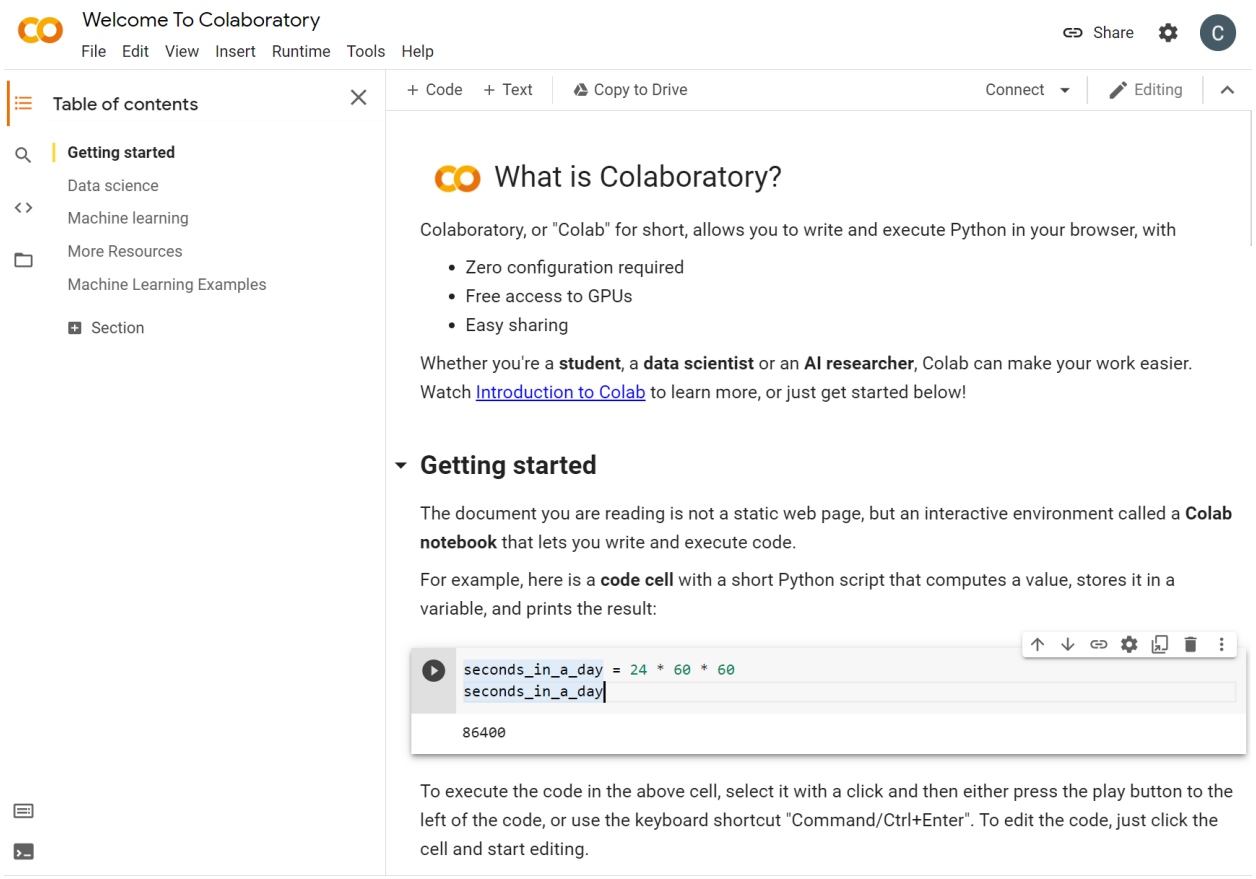

Figure 1: Google Colab Example

Google Colabatory, also known as Google Colab, is a web-hosted Jupyter Notebook service that is just as easy to set up as Google Documents. Unfortunately, Google Colab only has direct support for the Python language. As an interpreted language supporting read–eval–print loop (REPL), Python is well-suited for running individual code blocks. My computer vision assignments were completed in Google Colab. The assignment itself consisted of narrative text explaining the problem statement for the various code blocks breaking up the main program. After writing my code, I can run the program one code block at a time and any variables generated would be saved for future use. This was particularly useful for avoiding the need to rerun costly functions such as training a machine learning model when there is an error with

displaying the predicted output. A significant drawback is the lack of official support for other common languages such as Java or C++.

Finally, we should consider existing cloud IDE platforms. Acquired by Amazon Web Services (AWS) in 2016, Cloud9 was one of the first cloud IDEs to enter the field. The code editor, debugger, and terminal are accessible from a web browser and it supports over 40 languages including JavaScript, Python, C++, and Java. Coding environments can be quickly shared with other members of the development team and real-time collaboration is directly supported. There is no need to download or install any packages before beginning your project. While these are all very beneficial features, it is incomplete when considering its feasibility as a supplement to virtual learning.

System Design

At the highest level, SyncLink is designed to provide a comprehensive platform that allows for collaborative coding and review to software engineers, teaching staff, and students. Users can begin using the software without needing to go through the hassle of downloading and installing a local environment. Projects would be accessible through the cloud and not limited to local devices.

System Requirements

Gathering system requirements is a key step in any major software development project. By codifying functional and non-functional requirements, expectations can be properly managed and scope creep can be avoided. These requirements were created after consulting UVA CS

teaching assistants and students for feedback on the most useful features for remote collaborative coding and learning.

Functional Requirements

The system shall:

- Have a tiered access system that allows owners to either share projects with specific accounts or allow the project to be accessible through a link
- Allow users to upload program files and begin editing their contents
- Support real-time editing and highlighting from multiple simultaneous users
- Contain common IDE features such as syntax highlighting and debugging tools
- Compile and execute files in a variety of common programming languages
- Import images drawn/captured on secondary devices
- Create an interface where code, images, and narrative text can co-exist

Non-Functional Requirements

The system shall:

- Minimize the amount of time it takes to start and complete the process of sharing a file with another developer
- Ensure that users can only access projects that they are authorized to do so
- Be accessible from web browsers without the need to download or install software

Key Modules & Features

These requirements are fulfilled through SyncLink's modules. Together, these modules form the basis of a collaborative coding environment.

Cloud IDE

- Code editor, compiler, and debugger
- Accessible via the web on different devices
- Multiple users can collaborate on code and view output in real-time
- Tiered Permissions

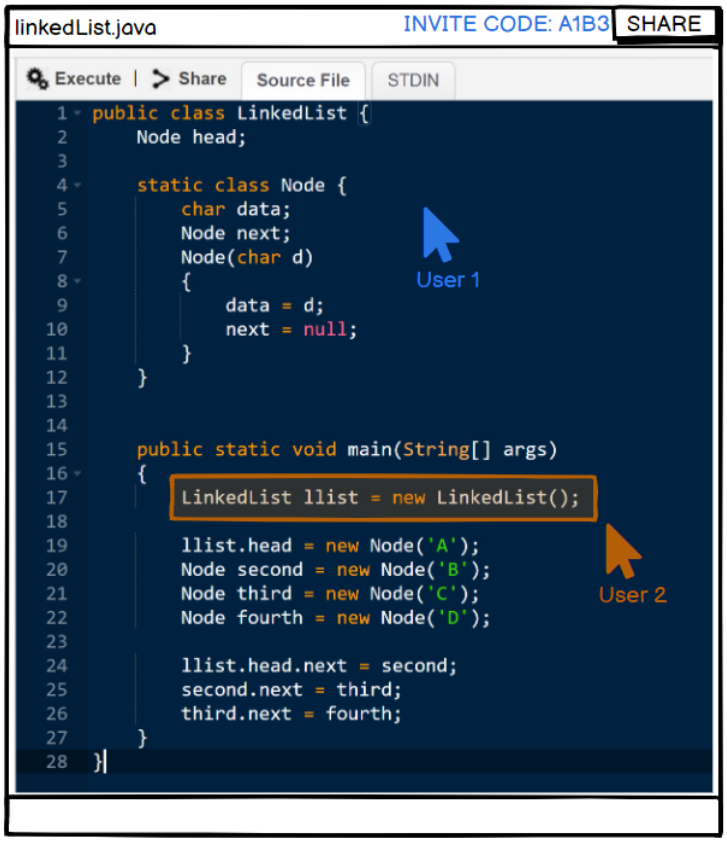

Figure 2: Real-Time Collaboration

This module is just a standard Cloud IDE similar to Cloud9. It is important to note that Cloud IDEs do suffer from certain drawbacks when compared to traditional IDEs (St-Pierre, 2010). For example, code is only available when there is an internet connection. In the event of an outage, developers would be unable to continue their work. Additionally, version-control

would be managed through the cloud IDE platform and users cannot choose their own software because the files are not local. That being said, the Cloud IDE module is well-suited for small projects such as CS programming assignments in lower-level courses. Its main benefit would be the ability to quickly show the teaching staff your files and begin collaboration without requiring too much setup time and knowledge.

Once a cloud project has been started, invitation links can be generated and shared with collaborators. There should also be options to generate "Anyone with this link can view" and "Anyone with this link can edit" links to increase ease of access for sharing files. For more private projects, invitations can be sent to specific emails and will require signing in through Google's SSO or another account system to be accessed.

Digital Whiteboard

- Different color pens
- Insert commonly used shapes and icons
- Save whiteboard contents as an image
- Link code to mobile application

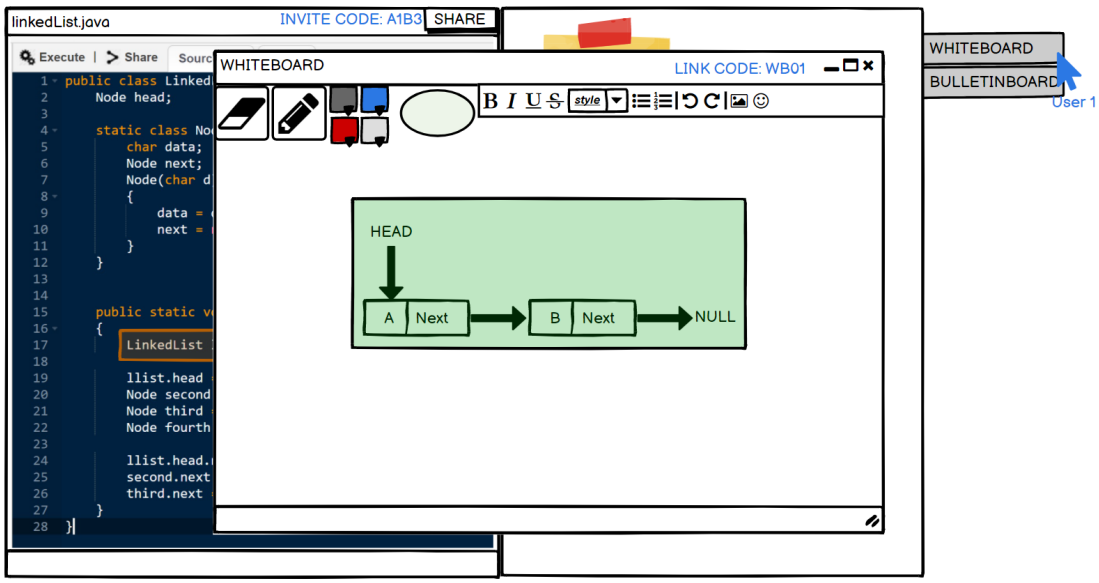

Figure 3: Drawing a Whiteboard Element

It can be quite beneficial to use an image to explain an idea or concept. When I spoke with other students during my own experiences as an APMA TA, I noted that a significant number of them were visual learners. I found that drawing out ideas would often improve my understanding or reveal a logical defect. One of the biggest drawbacks of going remote was losing access to the classroom whiteboard when getting help.

The Digital Whiteboard module serves as a virtual substitute. With its multiple colored pens, the user can distinguish and emphasize certain parts of the drawing. In certain classes, we tried to use Zoom's whiteboard and other virtual spaces to draw, but the lack of sensitivity on the thickness of the line and difficulties of using a mouse/touchpad made it hard to draw finer details and write legible text. To assist in the process, the user can also drag and drop common shapes such as rectangles and ovals into the drawing. Additionally, text boxes can be added to help label the drawing.

The contents of the digital whiteboard can be saved and inserted into the embedded notes/bulletin board for future reference. Finally, each whiteboard drawing has its own Whiteboard ID number to be used as a linking code to the mobile application. This will be discussed in further detail in a later section.

Embedded Notes

- Render markup language inside of the code editor window
- GUI to simplify markup language insertions
- Bulletin board for text, images, and other visualizations

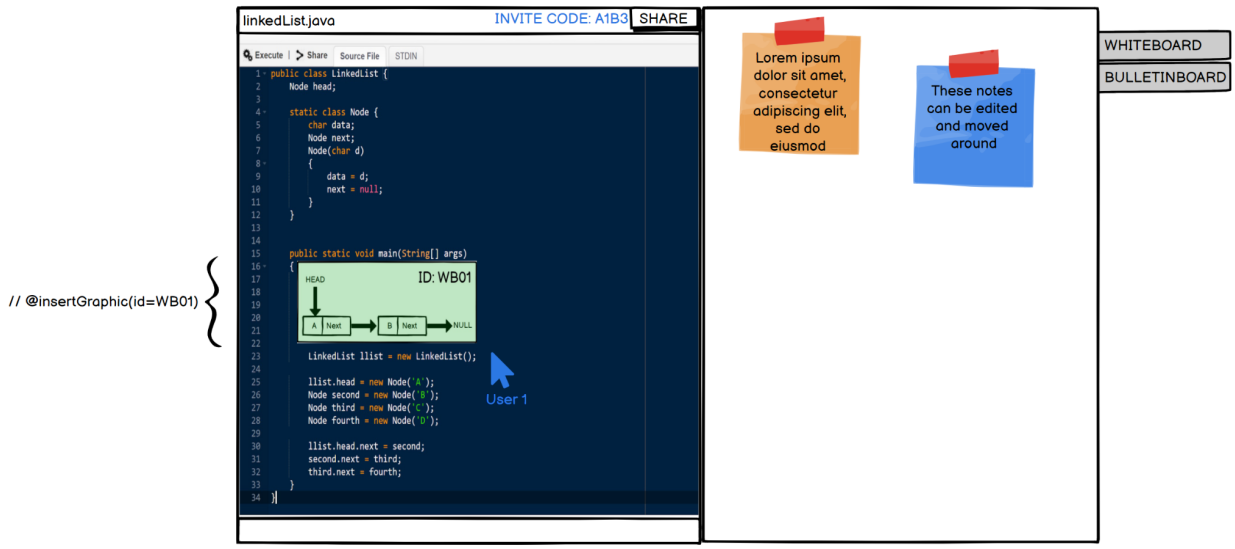

Figure 4: Embedding Images Within Code

Embedded Notes consists of two main features that exist in the code editor window and bulletin board tab. Comments are a crucial part of software development and are considered to be part of good coding practices. Documenting your code increases readability and maintainability. At its core, comments are just pieces of text with special syntax that tells the compiler/interpreter to ignore it. Documentation generators such as Javadoc use additional syntax to create HTML documentation from these comments.

However, it should be possible to directly embed these markup elements into the code editor window itself. Since these elements are in the same format as the coding language's native comment syntax, it would not be limited to specific interpreted languages. Similar to how Overleaf can render LaTeX code into a viewable format, SyncLink will render the markup language and expand the scope of comments to beyond just text strings. Images and tables and other visualizations can then be inserted as comments to a program file though referencing a specific ID. This will improve the narrative ability of traditional comments.

A GUI can help simplify the markup element insertion. For example, clicking on the "Table" symbol will open up an interface to set the number of rows and columns and enter values. When the interface is closed, the equivalent SyncLink comment will be inserted at the cursor location and rendered as a table. Clicking on this table will reopen the GUI.

The Embedded Notes module also contains a bulletin board style window synchronized across all users in the project. It is a home for text, images, and other visualizations that are not embedded within the code editor window. Users can pin and move the elements around.

Mobile App

- Document scanner/photo uploader
- Touch-based drawing surface
- Connected to Cloud IDE platform via accounts or quick invitation code

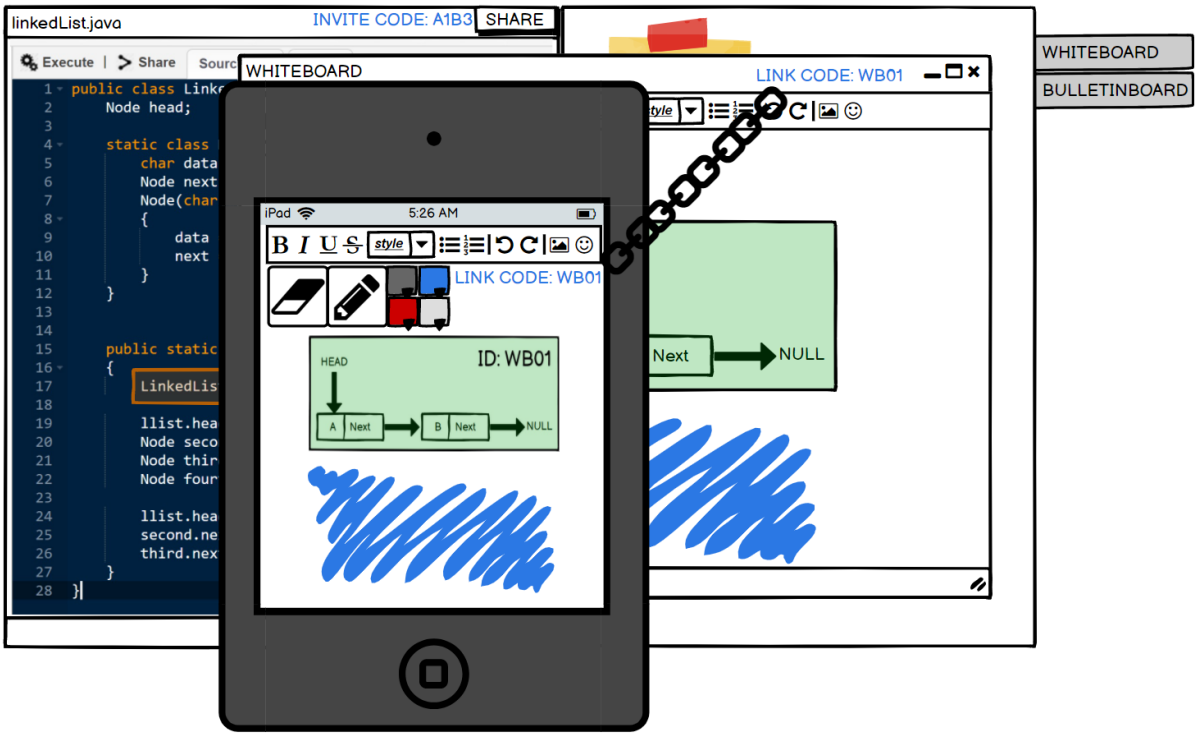

Figure 5: Linking Mobile App With Whiteboard Instance

The mobile application for smartphones and tablets is the final module of the collaborative system. As mentioned above, mobile devices are difficult to properly draw with unless there is a large surface area or specialized hardware such as a stylus or drawing tablet. Using the linking code mentioned in the Digital Whiteboard section, the user can instantly access the whiteboard instance on their device. With the use of a touchscreen or stylus, the user will have finer control over the details of their sketch.

The easiest way to quickly draw something is with a pencil and paper. However, getting that image from the desk into an office hour call can be a bit of a time-consuming process. First, the user needs to take a photo of the paper. The photo then needs to be emailed, uploaded to Google Drive, or sent through another communication app before reaching the computer. On the computer, the user must find the photo and enable screen sharing before continuing. The context switching leads to greater levels of distraction and is not a smooth process. By integrating a document scanner and photo uploader into the SyncLink ecosystem, the middleman can be removed. After scanning the paper image or selecting a file to upload, the SyncLink App will generate an image ID and automatically add the image to the Embedded Notes Bulletin Board.

Procedure

To illustrate an example of SyncLink being used to facilitate communication between students and teaching staff, please consider the following scenario. A CS 1110: Introduction to Programming student is working on their linked list Java implementation assignment and is running into difficulties with certain sections of the code. The student uploads their Java file to the SyncLink Cloud IDE website and activates the “Anyone with this link can edit” share setting. Afterward, the student joins the TA’s office hours over Zoom and gives them the URL link. The

TA is able to immediately access the project and the real-time capture of cursor/highlighted regions allows for the TA's attention to be directed towards the relevant code section.

Since the TA is able to see the entirety of the project from their own browser, they will not need relay instructions to the student about scrolling up and down to view different sections of the code like you would if this was done through screen sharing. After recognizing an issue with a code block, the TA highlights the region and asks the student if they can identify the mistake in their logic. The TA decides that a visualization would be beneficial for the student to recognize what was wrong with their conceptual understanding. If the TA has the mobile app downloaded, they can use the linking code to enter the project's whiteboard instance and take a picture of a quick paper sketch that they just drew. This image will automatically appear on the bulletin board and can be used as a supplement to the TA's explanation.

Results

The SyncLink platform was shown to TAs from CS 1110: Introduction to Programming and CS 2110: Software Development Methods. One TA commented that "I wish I had this platform this semester" and that "it would have been very beneficial." The shared highlighting and visible cursor location make it simple to point out specific sections of code. On occasion, the TA will want to be able to type something on the student's program for reference, and the shared IDE nature would complement that quite well.

Additionally, I spoke with a 4th Year CS student for feedback on SyncLink. He described the system as sounding useful and that the general functionality is good. This student mentioned that he previously had a technical interview using a cloud IDE and was surprised that he did not see its use in school before. It was beneficial to be able to show someone what you are writing as you are writing it.

Conclusion

At its core, SyncLink is a tool designed to help supplement remote collaboration in software development. By using the SyncLink ecosystem, developers can seamlessly work together in peer reviews. Educators can create files that guide students through difficult programming concepts using visuals and narrative text integrated into the code itself. Students can quickly connect with teaching staff and demonstrate their programs. In particular, COVID-19 caused a sudden shift towards remote education and this tool can mitigate some of the learning gaps caused by remote office hours. While many of the individual SyncLink modules have existing industry competitors, the combination of these features is a potent force.

Future Work

This proposal sets the foundation for the future implementation of the tool. As the next step, it would be beneficial to consult with additional members of the teaching staff to see what supplementary modules would be useful. Once a minimally viable product is developed, further refinement of the design could be done through a beta testing program involving UVA CS students.

As a comprehensive platform, SyncLink has a modular design and is suitable for expansion into different technical areas. As students search for full-time jobs and internships, they will almost certainly run into technical interviews. SyncLink's shared editing features would be helpful for interviewers to watch potential hires work through a coding challenge. The embedded visual elements in the code could help elaborate on the problem statement. Additional modules could be built to explicitly support conducting technical interviews.

References

Shull, F., Basili, V., Boehm, B., Brown, A., Costa, P., Lindvall, M., Port, D., Rus, I., Tesoriero, R., & Zelkowitz, M. (2002). What we have learned about fighting defects. *Proceedings Eighth IEEE Symposium on Software Metrics*, 0. https://doi.org/10.1109/metric.2002.1011343

St-Pierre, B. (2010, April 14). *What are the advantages / disadvantages of a Cloud-based / Web-based IDE?* Stack Overflow. https://stackoverflow.com/questions/2640993/what-are-the-advantages-disadvantages-of-a-cloud-based-web-based-ide

Veracode. (n.d.). *Integrated Development Environment*. https://www.veracode.com/security/integrated-development-environment#:%7E:text=The%20overall%20goal%20and%20main,and%20standardizing%20the%20development%20process.