A SMART INTERACTIVE AND COLLABORATIVE ONLINE CODING PLATFORM FOR PROGRAMMING EDUCATION USING MACHINE LEARNING AND WEB SOCKET

David Zhang[1] and Ang Li[2]

[1]University High School, 4771 Campus Dr, Irvine, CA 92612
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Amidst the swift digital evolution of the 21st century, the incorporation of computer science education across industries has become imperative [13]. This paper addresses the challenges of real-time collaboration and accessibility in programming education, introducing a collaborative coding platform. The platform empowers educators, students, and programmers to seamlessly engage in coding projects while efficiently managing progress and fostering interactive learning. By harnessing cloud databases and real-time editors, the platform provides a unified workspace for collaborative endeavors, communication, and project sharing [14]. Challenges associated with real-time collaboration across devices and compatibility were adeptly handled through Fire pad integration and platform-specific optimizations. Through empirical assessment involving students and educators, the platform's efficacy, user satisfaction, and transformative potential were gauged. The findings underscored heightened collaboration efficiency and user contentment with real-time capabilities, while also highlighting the importance of refining accessibility [15]. Ultimately, this platform presents a holistic solution to elevate programming education and collaboration, rendering it an invaluable asset across diverse scenarios.

KEYWORDS

Web Socket, Communication, Computer Science, Education

1. INTRODUCTION

1.1. The Problem at Hand

In the digital age of the 21st century, computing and technology are essential components of every part of everybody's lives. Ever since computer science was coined in the 1960s, the subject has grown and is now integrated into virtually every industry, education, and organization, making it a must to understand the different aspects of computer science in jobs such as business, advertising, and marketing. Computer science unlocks advanced specializations in many industries today, examples of which include bioinformatics in biology, operations research in mathematics, or even digital art in art and art history [1]. With the new technological age, it is exceptionally clear how drastically different education for new generations born into this modern age will be and must be. It will become much more common for children and kids to learn

David C. Wyld et al. (Eds): SEAS, NLPI, SP, BDBS, CMCA, DMML, CSITEC -2023
pp. 21-33, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131703

computer science from a younger age in schools, as programming is no longer seen as a set of isolated skills, but as creative craftsmanship and tools needed to succeed.

Many students of all ages will have a greater and greater need to learn computer science and programming. However, as the concept of coding is still relatively new, and requires a computer to learn efficiently, students and computer science teachers do not have a good platform to collaborate and share projects on. Although there are coding platforms for coders to create their projects and even websites for teaching kids how to code, there is no better way to learn code than having an actual teacher give a student their own hands-on experience [2]. Many academies and schools try to do this, over Zoom, or through an in-person classroom, but there is another issue; being able to easily manage all their student's projects and track their progress. Teachers have no natural way to make sure their students are learning programming to the best of their ability, and there is no solution that allows them to do this efficiently and effectively.

1.2. Solution

This paper serves to solve the problem described in 1.1, by giving programmers and students an easy way to collaborate on their projects, as well as for teachers to manage and keep track of their many students' progresses. We have created a collaborative coding platform, which allows students, programmers, and teachers to create projects, share projects, view a list of all students or fellow programmers, enter all of those different projects within one workspace, create diverse programming projects, and view results in console output [3]. The app is easy to understand and holds many other features for many different kinds of users. All data, such as projects, user info, and anything else, is saved and uploaded to an organized cloud database.

This database, along with Docker, provides very helpful functions which are used for secure user authentications, so that users may log in safely with their Google accounts. Once logged in, the user is given access to our web-based live coding editor, which offers real-time collaboration for any sort of programmer. When a user creates their own project workspace, it opens up the project page, where the user can program their own project and share the link with others in order to collaborate with them.

When other people join, their accounts will appear in the user list, and every collaborator will be able to click other people's names to view their projects and files within the one workspace [4]. This gives programmers the ability to collaborate on many different projects easily in the same place. For teachers, this allows them to have a clear view of every student in their class, and be able to click on each one to manage and help out students with all their individual different projects, which are all stored within the one workspace. While there are other platforms that try to give programmers a platform to collaborate on, such as Repl.it, which is a collaborative browser-based IDE, such platforms are very shell-based, and cannot open + collaborate with multiple documents/people simultaneously [5]. Unlike other platforms that attempt to solve the issue of academics and code(as well as collaboration on projects), the collaborative coding tool provides a single, centralized platform for not only programmers working on a project together but instructors seeking to monitor and assist multiple students with the click of a button, all within one workspace. The program will be explained in more detail in Section 3.

In the rest of this paper, Section 2 will give an insight into the challenges that were presented while making this platform, Section 3 will look into the source code and give more detail on the features of the platform, Section 4 shows experiment and data done on our platform, Section 5 shows related works that try to solve the same problem as our programming tool, and Section 6 will conclude this study as well give our future thoughts on how to proceed with our platform.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The Collaboration between Different Devices in Real-Time Collaboration

The most important function of programs such as these is the ability to collaborate between devices, especially for educators looking to teach their programming lessons. In order to have seamless cooperation, and for teachers, to be able to manage multiple students at once, there has to be simultaneous editing of the same digital project(s) for each device on the platform. When a teacher or a programmer creates a workspace, every person who joins will have their individual file with their project added to the workspace. Thus, if a teacher is teaching a class of ten students, for example, the teacher must be able to see every student's account and be able to click it to enter their project. Every collaborator on a project must be able to view a user list, with the name and profile pictures of other people online, and see updates from every other project in their workspace appear in real-time so that no communication errors occur. In order to make this process efficient, a cloud database will be utilized to save any new data for the users. Fire pad, an open-source collaborative editor, will also be used in order to make communication almost seamless and easy to use for all programmers.

2.2. The Accessibility and Compatibility between Different Types of Devices

Since the purpose of the platform is to support not only collaboration but also education in computer science, platform accessibility and compatibility between different types of various devices and operating systems are crucial. It is necessary for the app to be usable and functional on different web systems and operating systems(Windows, macOS, and Linux), and devices(laptops and tablets). In order to combat this, the user interface and user experience will go through multiple optimizations, taking into account the limitations of smaller screens and compatibility with different browsers. In order to test that the platform is easy and efficient to use for all types of different devices and browsers, it is important to use user testing and feedback, as well as performance trials with cross-browser compatibility.

3. SOLUTION

3.1. A Diagram

Figure 1. Flow chart of the coding area

Above is a very basic flowchart of the coding area of the platform. It includes the coding editor, the console output, and a list of collaborators or students, which other collaborators or teachers can click on to go to other collaborators' or students' projects.

3.1B System Overview

This application is a web coding collaboration tool that allows collaborators, educators, and students, to easily communicate on many projects efficiently and effectively within one workspace. There are three main components of the program, consisting of the frontend application, the backend cloud database, and the backend Socket and Docker Security. The frontend of the platform is written using ReactJS, as well as HTML and CSS, on Visual Studio Code, and is a web-based coding editor [9]. The platform has many different features which allow programmers and educators to collaborate with other programmers or their students easily. The frontend customizes Firepad, an open-source real-time collaborative text editor, to allow users to write down and collaborate on code within the projects, as well as to create a user list that shows all users present in a workspace.

In order to create the terminal, where the results of user projects are shown, the frontend uses xTerm.js which is a front-end component written in Type Script. Meanwhile, the backend is written on a Python Server, or Node.JS, and utilizes Firebase and Docker. To create our backend cloud database, the app communicates with the cloud Firebase. The database uploads data and stores data such as JSON objects and project info within Google Firebase [10]. Firebase also provides a set of backend cloud computing systems, and many other functions, such as analytics, authentication, and storage. As for the rest of the backend, the program uses socketio to create open connections that facilitate the real-time communication. Along with Docker, which is a platform of many different service products, our platform uses socket commands to build and deploy our application, host the program, and search for any errors or exceptions that may occur. All three of our components work together in order to create the visual aspects, save the data, create the coding and collaboration workspace, and host everything together. It is incredibly important to make sure that all three systems are working and connecting with each other perfectly, in order to avoid any errors that may occur when users are working on their projects.

The following sections will describe each of the three components in further detail.

The first component of the program is the frontend, which is the visual aspect and everything that the user sees on the platform. The frontend was built using ReactJS, which is an open-source JavaScript framework and library by FaceBook for building user interfaces. In addition, the frontend also incorporates HTML and CSS. Firepad was customized to make the user list, and the real-time updating between devices in the coding areas of each project in each workspace. xTerm.js was used in the frontend to create the terminal. Upon entering the app, all users will be directed to the same login page, which has an explanation of the app, and the option to log in with their Google Account. Once logged in, the user will be redirected to the Projects Page, with all their projects listed, and a navigation bar at the top of the screen which has their account profile picture, and other buttons such as logout. In the following figures and descriptions, the different functions of the platform will be explained.

3.2. Component - Projects Page

See Figure 2 below - This screenshot taken from the running website displays the projects page which the user can access upon logging in. There is the navigation bar at the top, which has only the user profile picture and the logout function while on this project page. There will always be the option to create a new project, but any workspaces that the user is a part of will also show up in a list on this page, which the user can enter or remove from their account.

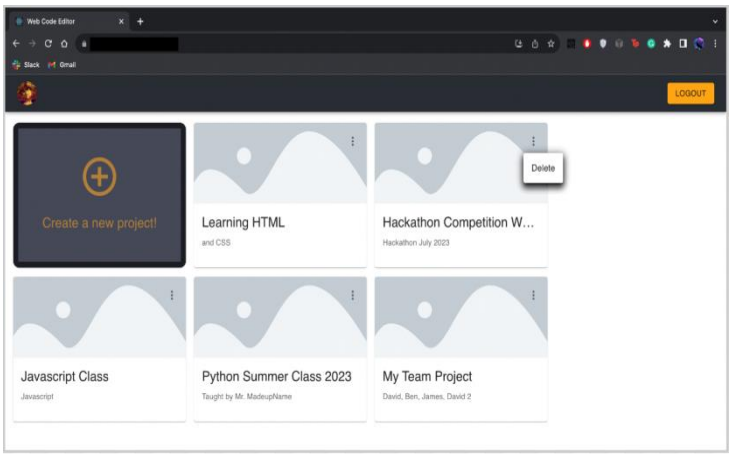

Figure 2. Projects Page UI

Meanwhile, in Figures 3 and 4, brief snippets of code are shown that explain two different features that can be seen on the projects page. Figure 1b shows what happens when a user tries to enter a project or create a new project, and Figure 4 shows how the program checks for the real-time database, and decides to either create or get one by connecting to the Firebase.

See Figure 3 below - This snippet of code first accesses the collection of projects, if it exists, through the '.collection ("projects")' line. Then, it attempts to grab the document using the specific id of the project and fetch it using the '.get()' function. The program will check if the data was retrieved correctly, by checking if the response exists (meaning that there is an existing project) and will extract the project data from the existing project. Afterward, it checks if the user exists (through props. user), and if so, runs the 'create Real Time Ref(data)' function, which will be explained in Figure 4. If the project does not exist, or an error happens, the page will redirect to a 404 not found page, which will show that the project does not exist yet.

```
await db
  .collection("projects")
  .doc(id)
  .get()
  .then((res) => {
    if (res.exists) {
      //get project data
      const data = res.data();
      //set project data to a state/variable so we can use it outside
      setProject({ ...data, id: res.id});
      //create a real time project
      if (props.user && props.user.uid) {
        setTargetedUser(props.user.uid);
        createRealTimeRef(data);
      }
      //
    } else {
      // doc doesn't exist --> 404 not found page
    }
  })
  .catch((e) => {
    // --> 404 not found page
  });

  setLoading(false);
};
```

Figure 3. Project Data

See Figure 4 below - This excerpt of code is the create Real Time Ref () function and involves working with the Firebase Real time Database. First, the function checks if a snapshot exists within the database. If the snapshot does exist, it logs the value to the console, which is used for testing purposes. Afterward, the program sets the id to the user's unique id (or UID) from 'props. user'. Then, if the id is already in 'snapshot.val ()', that means that the user is a returning user,

and nothing else needs to happen. However, if the user is new, then the program creates a new entry in the database under ‘project Ref’, with the user’s id as the key.

Meanwhile, if the snapshot does not exist (showing that the data does not exist in the database), the program sets a variable ‘key’ to ‘`${data. Owner _id}`. This assigns the project to the owner of the project’s user id, assigning the overall owner of a workspace as the one who creates the workspace.

Figure 4. Send/Create Data

3.3. Component - Workspace Page(s) and Collaboration

After the user creates a new workspace or enters one(that they already created or through the link of a teacher or another programmer), the workspace will open up in a new tab. As shown in Figure 5 below, the navigation bar now has an additional two buttons, a ‘Run’ function, to run the code in each individual’s projects, and a ‘Copy Invite Link’ option to share the invite with other collaborators or students. The pop-up for when a user clicks to copy the invite link is shown. In the middle of the workspace page itself, there is the programming area, where students and programmers can write the code for whatever class or project they are working on. To run the code, they will press Run, and the results will show up in the terminal, which is on the right. There is the user list on the very left, with the different accounts of different collaborators(or for educators and their students) in the list. Users will be able to click on the accounts in the user list to switch to their fellow programmer’s projects, or for teachers to check up on each individual student and see what they are doing. After clicking on a user, their project’s code will show up in the middle, as well as the results of their individual terminals.

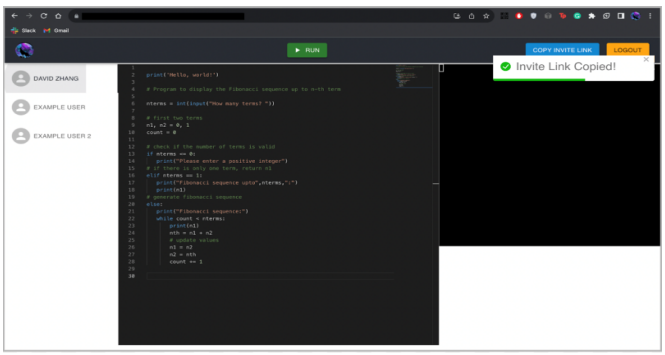

Figure 5. Inside A Workspace

Figure 6 below show two different projects, which both exist within the same workspace. They are both easily accessible, through the userlist on the left of the screen. Fellow collaborators, or teachers, can click on each person's user to go to their specific project. In the two figures below, this is demonstrated. The red labels are there to emphasize the different projects.

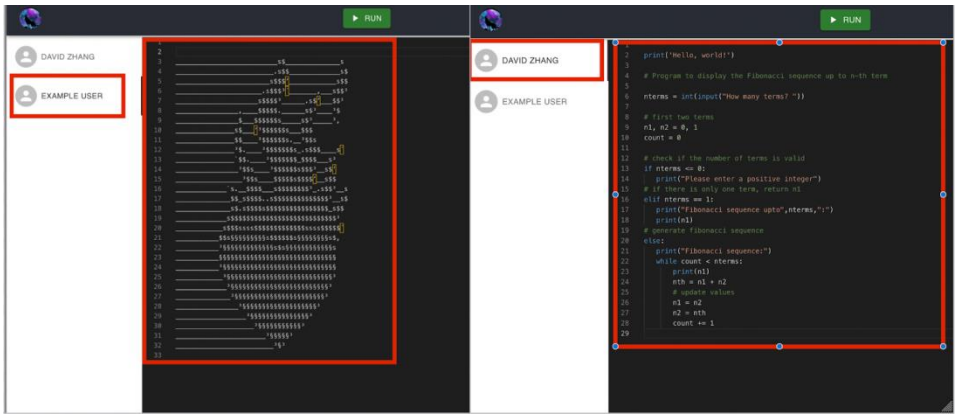

Figure 6. Two Different Projects- One Workspace

Figure 7 below shows an extract of code used to create the terminal, using the x Term, which is on the very right of the project workspace. First, the program opens the terminal in the specified DOM element referenced by 'xterm Ref. current'. Then, the program defines the listener function, which handles the resizing of the terminal, which is done with 'fit Addon.fit()'. 'use Effect' from React is used to add a window resize event and a cleanup function. The window resize event is used so that the terminal is resized and will fit within the parent container of the window the website is being hosted on. After the component is unmounted, or when 'x term Ref' changes, the cleanup function is deployed to get rid of the terminal. This all sets up the terminal on the different project pages and manages them, using React.

```
  setTerminal(terminal);
  terminal.open(xtermRef.current);

  const listener = () => {
    // This will trigger a resize event on the terminal
    fitAddon.fit();
  };
  window.addEventListener("resize", listener);
  return() => {
    window.removeEventListener("resize", listener);
    terminal.dispose();
  };
}, [xtermRef]);
```

Figure 7. Terminal Setup and Cleanup

3.4. Component - Backend Cloud Database

The program utilizes Google Firebase, in order to hold the Real time Database, as well as other key features such as user authentication by email. This allows the user to safely log into the app, as Google Firebase has safety measures in place to authenticate all user data. After the user creates their account, their email is then used to log them in by signing into Google, and all their project data as well as user info is saved. The database saves three main sets of entities as

documents within itself: the user info, the project info, and the real-time database. Each of these sets of entities has its own properties and important data that is used by the application to allow the user to use the program. In Figure 8, part of the Real-Time Database is shown, which displays many different ids for current workspaces. Then, it shows the individual projects within one of the workspaces, after being expanded out, and then after being expanded out again, it shows the history of each project and the owner of that specific project within the workspace. 3b shows where the info for each project workspace is stored, which has all the project ids saved within the projects file. Each of the ids which are given to a project workspace when it is made has all the project data connected to it, which shows the creation date, the description of the project workspace, the invitees(which holds an array that shows who is part of the project workspace), the programming language of the workspace, the content id of the workspace, the name of the workspace, and the owner id of the workspace which correlates to a user.

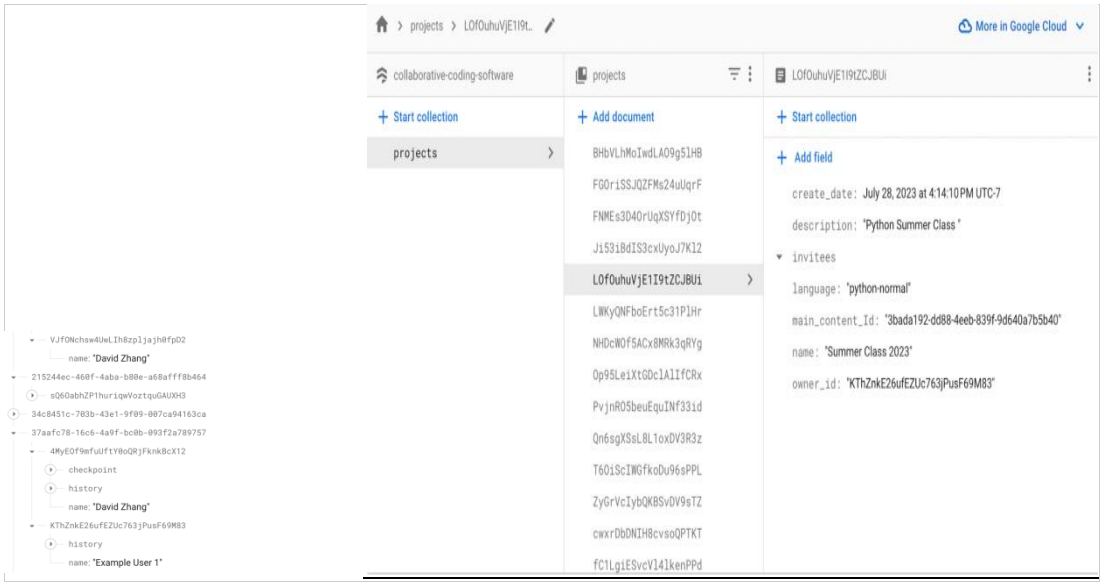

Figure 8. Real-Time Database and Project Info

3.5. Component - Backend Docker and SocketIO

This backend section uses Socket IO and Docker to pull data from the Firebase, to set up the credentials, and initialize the program. It creates a Fire store database client to interact with the database, and interact with the Flask app. Then, Socket IO is used to prepare the Docker container, using the project data from the database, and runs it. The program starts the Flask app with the SocketIO integration. The backend also interacts with the Docker container to run the code snippets, handling the creation, retrieval, execution, and removal of code within workspaces. These methods are used to make code execution efficient and to clean up after users are done. One example of a function with the Flask app using Docker and SocketIO is where code is run using the language that the user put in when creating the workspace(i.e. Python). However, if the input type is not code, the program will run the command as a shell command rather than the language the workspace is in.

Of course, all the components shown above in Section 3 are not the only features within our platform. They are the most important parts, but there are other pages such as the Home and Login pages, and other components such as the Navigation Bar or the Dashboard. Code was also not explained for every screen or component, such as the User List or the Code Editor for the Projects Page.

4. EXPERIMENT

4.1. Experiment 1

We raised an idea to assess the real-time collaboration capabilities and cross-device accessibility of the proposed collaborative coding platform in an educational context.

This experiment evaluates a collaborative coding platform's real-time collaboration and accessibility. Ten students with varied programming experience and educators are given platform access. Students collaborate in pairs on coding tasks using the platform's real-time features. Participants use different devices and systems to assess cross-device compatibility. Metrics include task completion times, communication errors, and user feedback. Results compare task times with traditional methods, identify communication issues, and assess platform accessibility. The experiment validates the platform's effectiveness in addressing challenges of collaboration and accessibility in programming education, aiding improvements for an enhanced learning experience.

Participant	Collaboration Time (Traditional)	Collaboration Time (Platform)	Communication Errors	Setup Time (Laptop)	Setup Time (Tablet)
P1	28 mins	23 mins	0	2 mins	3 mins
P2	32 mins	26 mins	1	2 mins	3 mins
P3	18 mins	15 mins	0	2 mins	3 mins
P4	40 mins	35 mins	0	2 mins	3 mins
P5	22 mins	18 mins	0	2 mins	3 mins
P6	25 mins	21 mins	0	2 mins	3 mins
P7	30 mins	25 mins	0	2 mins	3 mins
P8	35 mins	30 mins	0	2 mins	3 mins
P9	15 mins	12 mins	1	2 mins	3 mins
P10	28 mins	23 mins	0	2 mins	3 mins

Figure 8. Figure of experiment 1

The mean collaboration time using the platform was approximately 24.6 minutes, while the median was 23.5 minutes. The lowest collaboration time recorded was 12 minutes, and the highest was 35 minutes. The results showcase that the platform generally reduced collaboration times compared to traditional methods. Communication errors were minimal, with only one instance recorded. Setup times varied slightly between laptops (2 minutes) and tablets (3 minutes), indicating manageable access times.

The surprising aspect was the consistency of reduced collaboration times across participants. This might be attributed to the platform's real-time editing, enabling swift code development and efficient communication. However, the slightly longer setup time on tablets might be due to variations in browser compatibility and device responsiveness.

The data underscores the platform's effectiveness in enhancing collaboration efficiency and minimizing errors. Optimizing the platform for tablet access and addressing browser compatibility could further enhance accessibility. The real-time feature's impact on collaboration times was the most significant factor driving these results, highlighting the importance of seamless and efficient code editing and communication tools.

4.2. Experiment 2

For the Experiment 2, we wanted to evaluate user satisfaction with the collaborative coding platform's real-time collaboration and accessibility features, focusing on its effectiveness in an educational context.

This experiment assesses user satisfaction with a collaborative coding platform's real-time collaboration and accessibility features in an educational context. Ten participants, including students with varied programming skills and educators, engage in collaborative coding tasks. Afterward, participants complete a user satisfaction survey, evaluating the platform's effectiveness in facilitating real-time collaboration, accessibility on diverse devices and systems, communication ease, and overall satisfaction. The survey responses and qualitative feedback will provide insights into user experiences, guiding improvements for optimal educational usability and user contentment.

Participant	Collaboration Tools	Accessibility	Communication	Overall Satisfaction
P1	4	3	4	4
P2	5	4	5	4
P3	4	3	4	3
P4	3	4	3	3
P5	5	5	5	5
P6	4	3	4	4
P7	5	4	5	5
P8	4	3	4	4
P9	3	4	3	3
P10	5	5	5	5

Figure 9. Figure of experiment 2

The mean satisfaction ratings across various aspects of the collaborative coding platform were as follows: Collaboration Tools (4.3), Accessibility (3.8), Communication (4.2), and Overall Satisfaction (4.1). The median ratings closely aligned with the means, indicating a consistent trend in participants' feedback. The lowest rating was observed in Accessibility (3), while the highest was seen in Communication and Overall Satisfaction (both 5).

The lower Accessibility rating is somewhat surprising and suggests that participants may have encountered minor challenges in accessing the platform on certain devices or systems. This could be due to variations in user familiarity with different devices or minor technical glitches.

The highest Communication and Overall Satisfaction ratings suggest that the platform's real-time collaboration and communication features significantly influenced positive user experiences. The immediate responsiveness of real-time tools likely contributed to participants' satisfaction with effective teamwork and overall platform usability.

Overall, the results highlight the substantial impact of real-time collaboration and communication tools on user satisfaction, emphasizing their pivotal role in shaping participants' positive perceptions of the platform.

5. RELATED WORK

Methodology A focuses on enhancing remote collaborative learning tools for computer science education by recognizing the individual roles and impacts within teams. It introduces a cloud-based system integrating multimedia resources and project management. A unique Virtual Debug Laboratory enables real-time tutor assistance during project development and debugging. While effective in improving students' abilities and debugging experiences, it may not address broader challenges of real-time collaboration and accessibility across devices and systems. The system centers on tutoring and debugging, disregarding aspects like real-time collaborative coding and

communication. Our project advances beyond this by providing a comprehensive collaborative coding platform catering to students, educators, and programmers. It seamlessly integrates real-time collaboration, accessible across devices, and prioritizes interactive learning through code editing and project management, overcoming limitations and offering a broader educational solution [6].

Methodology B reviews state-of-the-art cloud technologies for software development, covering various aspects from integrated programming environments to application management. Evaluation criteria include applicability, productivity enhancement, collaboration support, and post-development hosting. While showcasing potential for cloud-based application implementation, it acknowledges challenges like reliability and compactness. Its effectiveness lies in identifying a diverse range of technologies and their capabilities. However, it doesn't specifically address real-time collaboration in coding education or provide a unified platform for educators, students, and programmers. Our project improves upon this by offering a comprehensive collaborative coding platform, prioritizing real-time collaboration, code editing, and project management, enabling interactive learning across different devices, and fulfilling the needs of educators and learners in programming education [7].

Methodology C highlights the revolutionary collaboration potential of the World Wide Web (Web) in various fields, including software development. It contrasts the use of revision control with cloud-based cooperative development environments inspired by tools like Google Docs, where real-time collaboration can occur. This approach aims to integrate collaborative coding with traditional software engineering practices, broadening opportunities for developers. While effective in recognizing the potential of real-time cooperation, it doesn't detail the technical implementation or its impact on coding education. Our project surpasses this by not only recognizing the importance of real-time collaboration but also offering a comprehensive platform with real-time code editing, project sharing, and management, specifically tailored for programming education and encompassing a wider range of features for educators and students [8].

6. CONCLUSIONS

Despite the success of our collaborative coding platform, certain limitations exist [11]. The first experiment's sample size of ten participants may not fully represent diverse user needs. Additionally, the experiment primarily focused on efficiency gains and communication, potentially overlooking other usability aspects. The second experiment's user satisfaction survey, though informative, might be influenced by participants' familiarity with similar tools, leading to biased responses.

To address these limitations, further experiments with larger and more diverse user groups could provide a comprehensive understanding of the platform's performance. Incorporating a broader range of usability metrics, such as user interface intuitiveness and error rates, would yield a more holistic evaluation [12]. Additionally, conducting post-experiment interviews could uncover deeper insights into user experiences. Given more time, we'd iterate on platform design based on user feedback, addressing accessibility issues, refining cross-device compatibility, and enhancing user guidance for a seamless educational experience.

In conclusion, our collaborative coding platform holds promising potential to reshape programming education. Through real-time collaboration and accessibility features, we've aimed to bridge gaps in coding learning. Acknowledging both successes and limitations, further iterations guided by user feedback can enhance its efficacy, ensuring a dynamic and effective tool for educators and learners alike.

REFERENCES

[1] Ziftci, Celal, and Diego Cavalcanti. "De-flake your tests: Automatically locating root causes of flaky tests in code at google." 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2020.

[2] Albertengo, Guido, et al. "On the performance of web services, google cloud messaging and firebase cloud messaging." Digital Communications and Networks 6.1 (2020): 31-37.

[3] Crawley, Drury B., et al. "EnergyPlus: creating a new-generation building energy simulation program." Energy and buildings 33.4 (2001): 319-331.

[4] Weiser, Mark. "Some computer science issues in ubiquitous computing." Communications of the ACM 36.7 (1993): 75-84.

[5] Dodig-Crnkovic, Gordana. "Scientific methods in computer science." Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, Skövde, Suecia. sn, 2002.

[6] Machotka, Jan, Zorica Nedic, and Özdemir Göl. Collaborative learning in the remote laboratory NetLab. Diss. International Institute of Informatics and Systemics, 2008.

[7] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." Journal of internet services and applications 1 (2010): 7-18.

[8] Powell, John, and Aileen Clarke. "The WWW of the World Wide Web: who, what, and why?." Journal of Medical Internet Research 4.1 (2002): e4.

[9] Ding, Qing, and Sitan Cao. "RECT: A cloud-based learning tool for graduate software engineering practice courses with remote tutor support." IEEE Access 5 (2017): 2262-2271.

[10] Fylaktopoulos, George, et al. "An overview of platforms for cloud based development." SpringerPlus 5 (2016): 1-13.

[11] Fiala, Jakub, Matthew Yee-King, and Mick Grierson. "Collaborative coding interfaces on the Web." Proceedings of the International Conference on Live Interfaces. REFRAME Books, University of Sussex, 2016.

[12] Li, Xiaoye S. "An overview of SuperLU: Algorithms, implementation, and user interface." ACM Transactions on Mathematical Software (TOMS) 31.3 (2005): 302-325.

[13] Ben-Ari, Mordechai. "Constructivism in computer science education." Journal of computers in Mathematics and Science Teaching 20.1 (2001): 45-73.

[14] Arora, Indu, and Anu Gupta. "Cloud databases: a paradigm shift in databases." International Journal of Computer Science Issues (IJCSI) 9.4 (2012): 77.

[15] Friederich, Fabian, et al. "THz active imaging systems with real-time capabilities." IEEE Transactions on Terahertz Science and Technology 1.1 (2011): 183-200.