Ege Eğitim Teknolojileri Dergisi
Journal of Ege Education Technologies
e-ISSN: 2667-4270
Cilt:6 Sayı: 1, Aralık 2022, Sayfa: 17- 28
Araştırma Makalesi

Öğrencilerin Covid-19 Pandemisi Sürecinde Programlama Öğrenme Deneyimleri

Erkan ER[1], *Gamze SÖKÜCÜ*[2]

[1] *ODTÜ, Eğitim Fakültesi, BÖTE, erkane@metu.edu.tr, orcid.org/0000-0002-9624-4055*

[2] *ODTÜ, Eğitim Fakültesi, BÖTE, gamze.sokucu@metu.edu.tr, orcid.org/0000-0002-0140-0837*

Geliş Tarihi: 14.06.2022 *Kabul Tarihi: 21.07.2022* *Yayınlanma Tarihi: 31.12.2022*

Özet

Küresel COVID-19 salgını nedeniyle yükseköğretim kurumlarında uzaktan eğitime ani bir geçiş yaşanmıştır. Eğitim ve öğretim faaliyetlerinin devam edebilmesi için zaruri olan bu hazırlıksız geçişin öğrenciler üzerinde bazı olumsuz etkileri olmuştur. Öğrencilerin öğrenme süreçlerinin olumsuz etkilendiği derslerden biri de programlama dersleri olmuştur. COVID-19 acil durum uzaktan eğitim sürecini inceleyen birçok araştırma gerçekleştirilmiştir. Ancak, pandeminin programlama derslerine olan etkileri neredeyse hiç araştırılmamıştır. Bu nitel araştırma çalışması, alanyazındaki bu boşluğu dikkate alarak lisans öğrencilerinin üç teknik derste programlama öğrenme deneyimlerini incelemiştir. Çevrimiçi bir anket uygulanarak öğrencilerin derslerin güçlü ve zayıf yönlerine ek olarak potansiyel iyileştirmeler konusunda görüşleri elde edilmiştir. Açık uçlu sorulara verilen yanıtlardan elde edilen veriler, Temellendirilmiş Teori kullanılarak analiz edilmiştir. Analiz sonucunda, üç ana kategori altında (öğrenmeyi destekleyen ders özellikleri, öğrenmeyi aksatan ders özellikleri ve dersler için olası iyileştirmeler) çeşitli temalar belirlenmiştir. Elde edilen bulgular, esnek öğrenci merkezli öğretimin ve öğrencilerin sıklıkla pratik yapasına fırsat veren uygulamalı programlama aktivitelerinin önemini vurgulamaktadır. Özellikle, pandemi sebebiyle ciddi bir dönüşüm geçiren eğitim ortamları dikkate alındığında, adaptasyon problemi yaşayan öğrenciler başta olmak üzere tüm öğrenciler ile hoşgörüye dayalı ve yapıcı yaklaşım temelli ilişkilerin kurulmasının önemi ortaya çıkmıştır. Sonuç olarak, nispeten hızlı bir tempoda ve yoğun bir iş yükünde dahi, sistematik bir ders tasarımının, çevrimiçi öğrenmede fark yaratabilmektedir. Bu noktada, ders tasarımının etkililiğinde öğrenci merkezli öğretim, yapılandırmacı öğretim felsefesi ve öğretmenler ve öğrenciler arasındaki sıkı bağ önemli rol oynamaktadır.

Anahtar Kelimeler: *programlama eğitimi; covid-19; bilgisayar eğitimi; hibrit öğrenme*

Ege Eğitim Teknolojileri Dergisi

Journal of Ege Education Technologies

e-ISSN: 2667-4270

Volume:6 No: 1, December 2022, Pages: 17-28

Research Article

Listening to Students' Voices: Learning Programming in the Covid-19 Pandemic

Recieved: 14.06.2022 *Accepted: 21.07.2022* *Published: 31.12.2022*

Abstract

With the global COVID-19 outbreak, the higher education institutions have experienced an abrupt transition to remote education. Although this emergency transition was mandatory for the continuation of the academic activities, it has resulted in some negative effects on students. The programming courses have been one of those where students' learning experiences were considerably harmed. Although many studies investigated the effects of COVID-19 emergency remote teaching, there has been little or no focus on programming courses. Attending this gap, this qualitative research study examined students' experiences of learning programming in three undergraduate-level technical courses. An online survey was administered to obtain students' responses to three open-ended questions about the things they liked, disliked, and recommend changing in the courses taken. The data were analysed and coded using Grounded Theory, which yielded several themes under three main categories: course characteristics enhancing student learning, course characteristics hindering student learning, and the potential improvements. The results highlight the importance of flexible student-centred instruction and frequent opportunities for hands-on practice in in hybrid teaching of programming courses during the pandemic. Moreover, the results indicate that it is important for teachers to establish relations based on tolerance and constructive approach with all students, especially those who have adaptation problems, when teaching during the pandemic. An important implication of the findings is that even with a relatively fast pace and a heavy workload, a systematic course design makes a difference in online learning when supported with a student-centred instruction informed by constructive teaching philosophy and rapport between teachers and students.

Keywords: *programming education; covid-19; computing education; hybrid learning*

INTRODUCTION

As technology has been constantly transforming society and industry, there is an increasing demand for professionals equipped with advanced technical knowledge beyond basic computer skills (Pothier, Wendy Girven Condon, 2019). To produce graduates with high competence and capability to shine in this digital era, higher education institutions offer an increasing number of technology courses such as machine learning, artificial intelligence, data science, and more importantly computer programming (Murphy et al., 2017). Learning a programming language has become an essential component of many undergraduate degrees (Amnouychokanant et al., 2021). While some introductory programming courses are offered during the freshman and sophomore years, relatively more advanced programming courses are introduced in the junior and senior years.

Although the educational opportunities in universities for learning a programming language is promising, the degree to which students benefit from these opportunities is highly questionable. Many research studies show that in programming courses a considerable number of students perform poorly and therefore fail to grasp the fundamentals of programming (Figueiredo & García-Peñalvo, 2021; Zingaro, 2015; Alturki, 2016). This pedagogical breakdown can negatively affect students' attitudes toward programming (Figueiredo & García-Peñalvo, 2021), and in long term can lead to the graduates without sufficient knowledge and the skill of programming, a highly demanded competence in current and future job markets (Kim & Lee, 2016).

Teaching programming courses is a difficult task for instructors not only because computer programming is relatively a difficult subject to teach (Alammary, 2019), but also because teaching needs to accommodate the student cohorts with diverse backgrounds (Medeiros et al., 2019), attitudes (Alturki, 2016), and skills (Malik et al., 2017). Not surprisingly, the traditional teaching approaches, which centre learning around teachers while favouring passive learning, have been found ineffective in programming courses (Horton & Craig, 2015). Novel pedagogical approaches that promote active learning are necessary to maintain student motivation, engagement and persistence in learning programming (Medeiros et al., 2019).

The global outbreak of the COVID-19 pandemic has impacted teaching and learning drastically (Pokhrel & Chhetri, 2021). As a measure to prevent the spread of COVID-19, most higher education institutions cancelled all face-to-face classes, and the faculty was forced to convert their face-to-face courses to online courses at very short notice (Bao, 2020; Hodges et al., 2020). Teachers tended to apply old traditional teaching methods in online environments, which led to ineffective bored lectures with minimal teacher-student interaction and communication (Güzel & Özeren, 2021). Students, who are not familiar with online learning, faced difficulties with concentrating on poorly designed online lessons and reaching out to their teachers and peers (Şengün, 2021).

Subject areas that involve hands-on practices such as computer science were affected the most by the transitioning to emergency remote teaching (Crick et al., 2020). Few studies have been conducted to investigate the impact of COVID-19 and emergency remote teaching on computer science education (Crick et al., 2020; Mooney & Becker, 2021; Deus et al., 2020). Although these studies provided important findings about the impact of pandemic on overall education in computer science, still very little is known about students' experiences in programming courses in particular.

PURPOSE OF THE STUDY

To listen to students' voices about their learning experiences in programming courses under the pandemic conditions, this study investigated the effects of different course characteristics on students' experiences of learning programming in multiple courses taught during the COVID-19 pandemic. An online survey was administered to collect qualitative data about the course characteristics favoured/unfavoured by the students and the possible improvements to support their learning more effectively. The following research questions guided this study:

1. What course characteristics enhanced students' learning experiences of programming?

2. What course characteristics hindered students' learning experiences of programming?
3. What could be the potential improvements in the programming courses for creating better learning experiences?

METHOD

This is a qualitative study focusing on students' experiences in programming courses during the COVID-19 pandemic. The data collection processes conducted in this research were approved by the METU Human Research Ethics Committee on February 15, 2022, with the document number 28620816.

In this study, a grounded theory approach was adopted. As stated by Glaser and Strauss (1967), grounded theory aims to systematically analyse the qualitative data of social research. Grounded theory enables the analysis of qualitative data in a way that leads to theories about human behaviour, rather than explaining the data analysis with existing theories. In other words, it, (Charmaz, 2006) asserts, grounds the theory with the data, which has no prejudice, nor any bias attributed to yet (Myers, 2013), before the theory construction takes place. Thanks to the grounded theory approach, researchers can identify emerging themes about a topic of interest in a social context and analyze the associations among them (Strauss & Corbin, 1990). By utilizing grounded theory in this research, we were able to perform a deep analysis of and derive new themes about students' learning experiences in the pandemic based on their open-ended responses.

Context and Participants

The context of this study was three programming courses taught by the same instructor in a Turkish university during the 2021 Fall semester. Although in each of these courses, a different programming language was taught (Course #1: Python, Course #2: C Sharp, and Course #3: Web Scripting Language), the very same pedagogical approach and course design was implemented in all. While Course #1 and Course #3 were the must courses, Course #2 was an elective course open to all departments. Lectures were conducted face-to-face and online at the same time. Students could opt to choose either option. Throughout the semester, mostly three to five students attended face-to-face classes while the majority joined online through Zoom. That is, although the hybrid modality was implemented, a substantial majority of the students preferred online participation. Further details about these courses are shared in Table 1.

Table 1: Principle course components and their distribution across courses

Component	Description	Course #1	Course #2	Course #3
In-class exercises	Several in-class exercises were included in the weekly lectures. During these exercises, the instructor led the code writing and shared his screen, while the students followed the instructor's guidance closely and wrote the same code on their computers. Students were required to submit their complete code as an assignment submission within the same day of the lecture.	11	11	12
Quizzes	On the lecture days, students were required to take a short quiz about the concepts learned on the same respective day. The quizzes consisted of 5 true/false or multiple-choice questions.	6	11	12
Lab assignments	Weekly lab assignments were an important part of the hands-on practice and assessment. Lab assignments were often built on top of the in-class exercises. Students were required to work on them individually.	10	12	9
Final project or assignment	Students were required to complete a final project or work on a comprehensive assignment (that involves developing a computer application) depending on the course.	1 final assignment	1 final assignment	2 projects
Communication and help-seeking	In all courses, Slack was used to support communication and help-seeking. Students were encouraged explicitly to post questions whenever they needed help and also to offer help to their peers. The instructor actively engaged with students' questions and answers in Slack. Replit, which is a web-based Integrative Development Environment (IDE), was used for collaborative code-writing in two courses. Replit allowed students to share their live codes with the instructor and to get instant feedback to improve their code.	Replit and Slack	Slack	Replit and Slack

The participants of this study were among the students registered in the courses mentioned above. Participation in the study was voluntary and students' consent was obtained beforehand. Among the participants were 37 female and 41 male students. While the majority of the participants were second (n=35), third (n=12) and fourth (n=23) grade students, there were also some fifth (n=7) and first (n=1) grade students. Most participants were from the department of Computer Education and Instructional Technology (n=72), and there were also students from other departments (n=6) such as Biology, Maths, and Physics.

Data Collection and Analysis

An online survey was developed and administered after the semester ended in order to collect the students' open-ended responses to three questions: (1) What did you like in this course?, (2) What did you dislike in this course?, (3) What changes do you suggest for improving this course? These survey questions correspond to the first, second, and the third research questions, respectively.

Descriptive qualitative data analysis was performed on students' open-ended responses to each of the survey questions separately. Open coding, axial coding, and selective coding, which are the steps in the grounded theory method, were performed (Strauss & Corbin, 1998). In the open coding phase, the student responses were read for the first time and were labelled with initial codes. Then, axial coding was performed to identify the connections between the initial codes. During this step, all responses were re-read to create broader abstract categories that encompass several connected codes. Lastly, selective coding was conducted to further refine previously identified categories. In selective coding, the categories were brought together in order to determine the core themes about students' experiences of learning programming.

Two researchers analysed the data independently and then came together to discuss the discrepancies in the emerging codes and themes. After the two iterations of refinement and discussion, 100% agreement was reached. Analysis of the collected qualitative data was performed using MAXQDA 2022.

FINDINGS

This exploratory study aimed to gain understanding of students' experiences in programming courses during the pandemic. The qualitative analysis resulted in several themes about the aspects of the courses that enhanced/hindered students' learning experiences and about the potential improvements in the courses for a better programming learning experience. The findings are presented for each research question separately as follows.

Course Characteristics Enhancing Programming Learning Experiences

The emerging themes about the positive course characteristics along with the sub-categories and counts are provided in Table 2. The counts refer to the number of students who make a related comment in the corresponding theme and subcategory.

Among five themes, Teaching and Philosophy has been the most prominent (n=80). Within this theme, three categories emerged: (a) flexible student-centred approach (n=31), hands-on practice-based approach (n=26), and positive attitude toward students (n=23). Flexible student-centred approach mostly comprises students' comments about hybrid instruction, availability of lecture recordings, and the flexible due dates. Hands-on practice-based approach mainly included the effectiveness of in-class exercises and lab assignments on students' learning. Furthermore, students highlighted the importance of instructor attitude toward them as an important factor enhancing their learning experiences. As the last category in this theme, students indicated that throughout the course the focus had always been on their learning not on the grades. Below are two student comments that highlight and represent most, if not all, aspects of the first theme; "I really like our teacher's teaching method. I think his teaching method is very effective compared to other courses. I can even say that I learned a subject for the first time", and "The attitudes and understanding of our teacher were one of the most important factors that connected me to this course".

The second major theme was Course Design and Teaching Method. Under this theme, students commonly highlighted that the course was a motivating and engaging experience for them (n=12), and combined with the teaching method the instructor adopted, it created a productive learning environment (n=12). Linked with these subcategories, students also provided positive comments about the overall course design and teaching method (n=8). In the meantime, the most noteworthy response for this subcategory is as follows; "As someone who has never studied coding before, I really liked the structure of the course ", highlighting the suitability of course for even false beginners in programming.

Another important subcategory was systematic progression and organisation (n=8). Students indicated that the way the course progressed each week was well-organised, "…it goes step by step to increase the understanding" and "there was an overall orderly run". What made this progression, based on the students' responses, are "… in-class activities, quizzes and lab assignments … to reinforce the topic", with the just right level of challenge; "the course is not very easy but not too difficult as well."

Table 2. Emerging themes about the course characteristics enhancing programming learning experiences

Themes and categories	Count	Themes and categories	Count
CONTENT AND MATERIALS	**27**	**TEACHING PHILOSOPHY**	**84**
Rich learning materials and activities	8	Flexible student-centred approach	31
Effective online IDE	8	Hands-on practice-based approach	26
Good quality content	7	Positive attitude toward students	23
Useful learning activities	4	Emphasis on learning than grading	4
COURSE DESIGN AND TEACHING METHOD	**43**	**EFFECTIVE COMMUNICATION AND HELP**	**33**
Motivating and engaging	12	Effective help from instructor	17
Productive learning environment	12	Effective & prompt comm. with instructor	10
Overall good course design & teaching	8	Effective communication and help via slack	6
Systematic progression and organisation	8	**EFFECTIVE ASSESSMENT**	**6**
Timely available materials	3	Open-book assignments/projects	4
Effective teaching method	3	Weekly quizzes for self-assessment	2

The other prominent themes were Effective Communication and Help (n=33), and along with Content and Materials (n=27). Regarding the preceding theme, effective and prompt help from (n=17) and communication with (n=10) the course instructor were the most important factors as articulated by the students; "the instructor was helpful and made the class engaging". Additionally, students explicitly indicated that communication and help-seeking was enhanced via Slack (n=6); "another thing I love most was Slack. It was always with me", since "slack helped me(them) a lot in this(such) situation(s)". Regarding the latter theme, rich learning materials and activities (n=8), Effective online IDE (n=8), and good quality content (n=7) were noted as the aspects of the courses that enhanced student learning. The lab assignments specifically designed to be motivating, "I have always loved doing labs" and purposeful, "all of the homework was fun and useful" led to, as the participant responses reveal, a student friendly atmosphere; "I enjoy the lesson and become more eager to learn".

As the last theme, Effective Assessment was found to be the least prominent (n=6). Few students indicated the usefulness of open-book assignments or projects and weekly quizzes for their learning, "I also liked the quizzes are not very difficult, I get a sense of satisfaction whenever I got good score on the quiz" and so (open assignments and projects were) better than taking exams and reading guidelines."

Course Characteristics Hindering Programming Learning Experiences

The emerging themes about course characteristics that hindered students' learning experiences are provided in Table 3, along with the categories and counts. In the overall sense, compared to those enhancing students' learning experiences, the course characteristics that hindered student learning were considerably fewer.

According to Table 3, the main issue reported to be the ineffective lecturing (n=17). In particular, students considered the pace of lecture very fast (n=12), "I didn't like the fact that the pace of the in-class exercises are very fast sometimes, it is hard to follow and to catch up, I get confused most of the time and have to watch the class recording later", which, in return, affected their in-class participation and experience negatively. Next, the difficulty level (n=10) and the workload (n=11) were found to be other principal factors hindering students' learning experiences, "since the subject of web design is quite comprehensive and detailed, I can say that I had difficulties in some labs". Under these two themes, labs were the main activity that students experienced difficulty with submitting on time. In particular, having multiple tasks at a time to complete was not favoured by the students: "having one lab, one in-class activity and one quiz that I have to work on every week sometimes put a lot of stress on me in terms of time."

Table 3. Emerging themes about the course characteristics hindering programming learning experiences

Themes and categories	Count	Themes and categories	Count
INEFFECTIVE LECTURING	**17**	**DIFFICULTY LEVEL**	**10**
Fast pacing during lectures	12	Difficult labs	8
Intense lectures	2	Difficult projects	1
Long lectures	1	Difficult course in general	1
Noninteractive async labs	1	**COURSE WORKLOAD**	**11**
Passive nature of in-class code-writing	1	High course workload in general	5
POOR CONTENT/ACTIVITIES	**7**	Demanding labs	4
Useless quizzes	5	Tight schedule with many topics	1
Insufficient material	2	Demanding final project/assignment	1

The other theme emerged was Poor Content/Activities (n=7). Under this theme, quizzes, administered weekly on the lecture day, were considered to be affecting the learning experience in a negative way, i.e., "I think weekly quizzes were not beneficial to me. I did not learn anything from them."

Potential Improvements for the Programming Courses

Six themes were identified from the analysis of students' suggestions for improving the programming courses, as shown in Table 4. The most prominent theme was The Mode of Instruction (n=15). The categories under this theme indicate that some students preferred to move fully to face-to-face teaching for lectures (n=6) and labs (n=4). Some students also asked to have at least some sessions face-to-face (n=4), while few suggested reducing the pace of instruction (n=2); "slowing down in some weeks would be better."

Table 4. Emerging themes about the possible improvements in the programming courses

Themes and categories	Count	Themes and categories	Count
UPDATING THE ACTIVITIES	**6**	**UPDATING THE COURSE MATERIALS**	**10**
Group work/projects	3	Covering new topics	4
Additional labs	1	Providing additional resources	3
Faster feedback for labs	1	Improving the existing materials	2
Problem-based approach in lectures	1	**THE MODE OF INSTRUCTION**	**15**
REDUCING WORKLOAD	**13**	Moving to fully face-to-face	6
Less number of assignments	4	Having some sessions face-to-face	4
Minimal number of quizzes	4	Having labs face-to-face	3
Less in-class exercises	3	Reducing the pace of instruction	2
Having a single project	2	**TIMING OF THE ACTIVITIES**	**5**
OVERALL COURSE SCHEDULE	**5**	Giving more time for assignments	2
Having two sections	3	Giving earlier access to materials	2
Longer lectures	2	Starting the projects early	1

Reducing the Workload was the second major theme. Under this theme, students suggested having a smaller number of assignments (n=4) and either reducing the number or removing the quizzes all together (n=4). Some recommended fewer in-class exercises during the lecture (n=2) and having only a single project for the semester (n=2); "maybe one of the assignments could be removed from the final part, or the number of webpages that we had to design could be decreased". The other important theme was Updating the Course materials (n=10). This theme contained four specific categories which indicated students' suggestions for including new topics (n=4), expanding the course materials with additional resources, and improving the existing learning materials (n=2); “some user experience topics”, or “some other libraries can be covered briefly.”

The remaining three themes were relatively minor. Under the Updating the Activities theme (n=6), students mostly recommended having opportunities for group work (n=3) and gave some suggestions for improving lab and lecture experiences such as having a problem-based approach and providing prompt feedback. Regarding the Timing of the Activities theme (n=5), students mainly commented on having longer due dates for assignments as “time spans can be more flexible” and earlier access to materials such as course video recordings; “accessing the module before the lesson would have made it easier for me to learn.” In the last emerging theme, Overall Course Schedule (n=5), students recommended having multiple sections for the courses (n=3) and allocating more time for lectures in the schedule (n=2); “please increase the time allocated for this course so the instructor does not have to rush through the content and the students can understand better.”

DISCUSSION

Since the global the COVID-19 pandemic started in December 2019 in Wuhan, China, the higher education institutions have undergone a major shift to online and hybrid instruction. This shift has led to numerous problems with teaching and learning, especially in courses that require regular hands-on practices such as computer programming. Identifying good and bad practices in teaching programming during the pandemic from students’ viewpoint can help practitioners redesign their courses to create the optimal learning conditions for the students.

First, the findings of this study highlighted the importance of a flexible student-centred approach in learning programming during the pandemic. Students mostly appreciated the hybrid modality since they were able to attend lectures without physical constraints and watch the video recording of the lectures later whenever they missed any sessions. Extended deadlines were also found to be important for student learning in the pandemic. Thus, incorporating an optimal level of flexibility in teaching during the pandemic might create a positive impact on engagement and achievement levels, given the high number of students who experience difficulty with online learning (Şengün, 2021) and suffer from the stress of managing heavy course load during the COVID-19 pandemic (Çetinkaya & Asıcı, 2021).

Next, the findings indicate that the frequent opportunities for hands-on practice yielded a positive learning experience. Besides the weekly labs, students highly acknowledge the in-class exercises, which allowed students to write code in order to develop some basic computer applications following the guidance of the instructor. Such hands-on teaching can allow students to apply abstract programming concepts in practice toward solving some real-world problems with potential immediate feedback from instructors (Rhudy & State, 2016). In a programming course where hands-on teaching was implemented, Handur and his colleagues (2016) noted an improvement in students’ performances, learning, and logical thinking abilities, and a promising increase in interactions between teachers and students. In the same vein, the findings of our study provide further evidence toward the effectiveness of hands-on practices during the lectures in programming courses.

Additionally, students noted the positive influence of the instructor’s constructive attitude and the explicit focus on their learning. In line with this, students also indicated that the course provided a productive learning environment, and they were quite motivated and engaged. Thus, a safe and positive learning atmosphere played a critical role in creating an encouraging and stimulating environment where students focused on the mastery of the programming concepts. These findings are crucial given the vital role of students’ motivation and attitudes in learning programming (Alturki, 2016). Programming is often perceived as a difficult subject area by students, and this perception, in return, affects their motivation, engagement, and achievement negatively (Santos et al., 2013). According to the results of this study, instructors should explicitly show their positive attitudes toward student learning and design their courses in a way that motivates and encourages the diverse student cohorts to learn programming.

Another important finding, as stated by the students, was the importance of effective communication with the course instructor and his unwavering support throughout the course. The use of the Slack application facilitated communication in all courses. The instructor responded to each and every inquiry from students throughout the semester and aimed to provide specific feedback on students' work-in-progress codes as needed, which led to a positive learning experience as students mentioned. This finding suggested that feedback, which is strongly associated with student learning (Panadero et al., 2018), should be integral to teaching of programming courses. Previous studies also showed that continuous feedback can support novice programming students' learning processes and can make a considerable impact on the dropout rates in programming courses (Vihavainen et al., 2011). On the other hand, findings did not provide enough evidence about the collaboration among students although Slack was introduced as well as encouraged also to promote help-seeking between students. One reason for the lack of collaboration in this study might be the instructor's promptness in addressing students' questions. Instructors should incorporate new strategies to promote peer support in online help-seeking and discussions. For example, one strategy might be gamification, which is found to promote student engagement in online discussions (Ding et al., 2018).

Moreover, according to the results, some students indicated that the pace of instruction during the lectures was too fast to follow. Additionally, they mentioned that the course workload was heavy, and the assignments were difficult. Although these students were not majority, they are likely to be the novice programmers. Collaboration with peers might be one effective strategy to support their learning processes. Research shows that collaboration in programming courses can lead to a considerable improvement in students' performances and pass rates (Vihavainen et al., 2011). When collaborating with peers, students tend to feel more engaged and motivated and, therefore put more effort in completing programming tasks (Alturki, 2016). To promote collaboration with peers, the instructor could design specific collaboration activities both during the lecture and outside the lecture. For example, during the lecture students can work in pairs using the same computer and switch between the roles of writing the code and monitoring the peer and warning about the errors.

From a theoretical lens, these findings highlight the importance of self-directed learning (Hiemstra, 1994). In self-directed learning, students have the principal responsibility for their learning process, which includes planning and management of their time and resources, and regulation of their motivation and engagement in a way that maximises their learning and achievement (Hiemstra, 1994). Learners with better self-directing skills are more likely to succeed in online education (Chu & Tsai, 2009). Given that students took multiple online courses at the same time, such skills have played an important role during the pandemic. The findings of this research regarding the flexibility, student-centred approach, and effective communication with the instructor provide some evidence towards students' needs for supporting their self-directed learning attempts.

CONCLUSION

During the COVID-19 pandemic, the entire world has experienced an irreversible shift in terms of centuries-old teaching and learning habits. This study has been a qualitative attempt to understand the experiences of students in programming courses redesigned with hybrid instructions during the pandemic.

Implications

The findings of this study highlight the importance of several pedagogical approaches in effectively supporting student learning in programming courses. Adopting a flexible student-centred teaching approach while implementing problem-based learning in a hands-on manner is found to contribute to the learning in the best possible way. In the meantime, it increases motivation and creates engagement, which leads to better learning experiences. Thus, this forced move to emergency remote education, when supported by the teachers with a well-educated teaching philosophy and pedagogical approaches, can have a serious potential and so much more to offer for the students even in the midst of a pandemic. Based on the findings of this study, the following principles are suggested for designing effective programming courses:

- An effective communication channel should be established to facilitate and encourage asking questions and discussions.
- Instructors should not reply to students' questions too soon or too late for promoting peer interactions but not leaving the students' questions unattended.
- The instructors should explicitly transmit their positive attitudes toward students and the focus on student learning not the grades.
- The lectures should include in-class hands-on exercises by which students can immediately apply the theory in practice and receive quick feedback from the instructor and peers.
- Instructors should provide flexibility in a way that promotes student-centred learning (e.g., video recordings for students missing the live lectures, extending deadlines, etc.)

Limitations and Future Research

Along with the insights it provided, this study has several limitations. First, it has made use of a single type of data, which results in not having much breadth in terms of capturing students' experiences. For example, quantitative data could be collected through a questionnaire about students' perceptions about various course components to bring additional insight about how they perceived different elements in the courses throughout the semester. Secondly, the data collected from the three programming courses were given by the same instructor with the same instructional design. Although this was helpful to focus on the effectiveness of a very specific course design and pedagogical approach, richer findings could be obtained if the courses with different designs were included. Therefore, future research can collect several types of data in multiple distinct contexts to obtain a more comprehensive understanding of student engagement and learning in programming courses. Moreover, although the students reported a positive learning experience overall, the effectiveness of the programming courses on students' performance is not known. More experimental research is needed to close this gap.

ACKNOWLEDGEMENTS

We thank Prof. Dr. Soner Yildirim for his constructive feedback on the manuscript.

REFERENCES

Alammary, A. (2019). Blended learning models for introductory programming courses: A systematic review. *PLoS ONE*, *14*(9), 1–26. https://doi.org/10.1371/journal.pone.0221765

Alturki, R. A. (2016). Measuring and improving student performance in an introductory programming course. *Informatics in Education*, *15*(2), 183–204. https://doi.org/10.15388/infedu.2016.10

Amnouychokanant, V., Boonlue, S., Chuathong, S., & Thamwipat, K. (2021). A study of first-year students' attitudes toward programming in the innovation in educational technology course. *Education Research Internationa*, *2021*.

Bao, W. (2020). COVID-19 and online teaching in higher education: A case study of Peking University. *Human Behavior and Emerging Technologies*, *2*(2), 113–115.

Çetinkaya, M., & Asıcı, E. (2021). Covid 19 pandemisinin 12 sınıf öğrencilerinin üniversite sınavına hazırlık sürecine yansımaları. *Millî Eğitim Özel Eğitim ve Rehberlik Dergisi*, *1*(2), 62–94.

Charmaz, K. (2006). *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis.* Sage.

Chu, R. J., & Tsai, C. C. (2009). Self-directed learning readiness, Internet self-efficacy, and preferences for constructivist Internet-based learning environments among higher aged adults. *Journal of Computer Assisted Learning*, *25*(5), 489–501.

Crick, T., Knight, C., Watermeyer, R., & Goodall, J. (2020). The impact of COVID-19 and "emergency remote teaching" on the UK computer science education community. *Proceedings of UKICER '20: United*

Kingdom & Ireland Computing Education Research Conference, 31–37.

Deus, W. S. de, Fioravanti, M. L., Oliveira, C. D. de, & Barbosa, E. F. (2020). Emergency remote computer science education in Brazil during the COVID-19 pandemic: Impacts and strategies. *Brazilian Journal of Computers in Education (Revista Brasileira de Informática Na Educação – RBIE), 28*(2020), 1032-1059.

Ding, L., Er, E., & Orey, M. (2018). An exploratory study of student engagement in gamified online discussions. *Computers and Education*, *120*. https://doi.org/10.1016/j.compedu.2018.02.007

Figueiredo, J., & García-Peñalvo, F. (2021). Teaching and learning strategies for introductory programming in university courses. *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21)*, 746–751. https://doi.org/10.1145/3486011.3486540

Glaser, B. G., & Strauss, A. L. (1967). *The Discovery of Grounded Theory. Strategies for Qualitative Research*.

Güzel, İ., & Özeren, E. (2021). Covid-19 pandemi sürecinin lisansüstü eğitim faaliyetlerine etkisi. *Electronic Journal of Education Sciences*, *10*(20), 167–185.

Handur, V., Kalwad, P. D., Patil, M. S., Garagad, V. G., Yeligar, N., Pattar, P., Mehta, D., Baligar, P., & Joshi, G. H. (2016). Integrating class and laboratory with hands-on programming: Its benefits and challenges. *Proceedings of the 2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education, MITE 2016*, 163–168. https://doi.org/10.1109/MITE.2016.47

Hiemstra, R. (1994). Self-directed learning. In T. Husen & T. N. Postlethwaite (Eds.), *The International Encyclopedia of Education* (2nd editio). Pergamon Press.

Hodges, C., Moore, S., Lockee, B., Trust, T., & Bond, A. (2020). *The difference between emergency remote teaching and online learning*. Educause Review Website. https://er.educause.edu/articles/2020/3/the-difference-between-emerge ncy-remote-teaching-and-online-learning.

Horton, D., & Craig, M. (2015). Drop, fail, pass, continue: Persistence in CS1 and beyond in traditional and inverted delivery. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 235–240.

Kim, S. W., & Lee, Y. (2016). The effect of robot programming education on attitudes towards robots. *Indian Journal of Science and Technology*, *9*(24), 1–11.

Malik, S. I., Mathew, R., & Hammood, M. (2017). PROBSOL: A web-based application to develop problem-solving skills in introductory programming. In A.-M. A. & Curran K. (Eds.), *Smart Technologies and Innovation for a Sustainable Future* (pp. 295–302).

Medeiros, R. P., Ramalho, G. L., & Falcao, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, *62*(2), 77–90. https://doi.org/10.1109/TE.2018.2864133

Mooney, C., & Becker, B. A. (2021). Investigating the impact of the covid19 pandemic on computing students' sense of belonging. I. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21*, 612–618.

Murphy, E., Crick, T., & Davenport, J. H. (2017). An analysis of introductory programming courses at UK universities. *The Art, Science, and Engineering of Programming*, *1*(2), 2–23.

Myers, M. D. (2013). *Qualitative research in business and management* (2nd ed.). Sage.

Panadero, E., Jonsson, A., & Alqassab, M. (2018). Peer feedback used for formative purposes: Review of findings. In A. Lipnevich & J. K. Smith (Eds.), *The Cambridge Handbook of Instructional Feedback*.

Pokhrel, S., & Chhetri, R. (2021). A literature review on impact of COVID-19 pandemic on teaching and learning. *Higher Education for the Future*, *8*(1), 133–141. https://doi.org/10.1177/2347631120983481

Pothier, Wendy Girven Condon, P. B. (2019). Towards data literacy competencies: Business students, workforce needs, and the role of the librarian. *Journal of Business & Finance Librarianship*, *Online*. https://doi.org/10.1080/08963568.2019.1680189

Rhudy, P. M., & State, P. (2016). Integrated development of programming skills using MATLAB within an undergraduate dynamics course. *Proceedings of ASEE's 123rd Annual Conference & Exposition*.

Santos, A., Gomes, A., & Mendes, A. (2013). A taxonomy of exercises to support individual learning paths in initial programming learning. *Proceedings of 2013 IEEE Frontiers in Education Conference (FIE)*, 87–93.

Şengün, G. (2021). Farklı üniversitelerde öğrenim gören üniversite öğrencilerinin COVID- 19 pandemi sürecine yönelik görüşleri. *Karamanoglu Mehmetbey Educational Research*, *3*(2), 94–102. https://doi.org/10.47770/ukmead.943044

Vihavainen, A., Paksula, M., & Luukkainen, M. (2011). Extreme apprenticeship method in teaching programming for beginners. *SIGCSE '11: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 93–98.

Zingaro, D. (2015). Examining interest and grades in computer science 1: A study of pedagogy and achievement goals. *Transactions of Computer Education*, *15*(3).