Providing Technical Support to Sustain Student Motivation and Engagement in Software Engineering Project-Based Learning

Ahmad D. Suleiman
Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY, USA
as4300@rit.edu

David Shepherd
Department of Computer Science
Louisiana State University
Baton Rouge, LA, USA
dshepherd@lsu.edu

Jan DeWaters
Institute for STEM Education
Clarkson University
Potsdam, NY, USA
jdewater@clarkson.edu

Yu Liu
Electrical & Computer Engineering
Clarkson University
Potsdam, NY, USA
yuliu@clarkson.edu

Daqing Hou
Software Engineering Department
Rochester Institute of Technology
Rochester, NY, USA
dqvse@rit.edu

***Abstract*—In this research paper, drawing from our own and other computing instructors' experiences, we highlight common technical challenges faced by students in software engineering project-based learning (PjBL) and discuss ways in which instructors can support students in overcoming them so that motivation is summoned and sustained. Through the use of practical hands-on experiences, PjBL has been shown to be an effective educational approach. However, unless projects are intentionally designed and supported in a way that summons and sustains student motivation, PjBL is likely to fail to accomplish its goals. Several factors influence student motivation, including their perception of the project's value and how confident they are in their ability to complete it. In particular, challenges that students perceive as insurmountable during the project can significantly weaken their motivation. On the other hand, supporting students to overcome such hurdles can be troublesome, especially in large classes as well as classes with diversity in student backgrounds. To generalize from our own experience, we designed a questionnaire targeted at PjBL computing instructors that contained closed and open questions on technical challenges faced by students, support instructors provided to overcome such challenges, and lessons learned by instructors on the effectiveness of their support. A total of 47 responses were collected from instructors with diverse backgrounds in terms of courses taught, students' years, and class sizes. We categorized the technical challenges into three main categories, namely (a) challenges in installing and configuring software packaged tools, (b) lack of prerequisite knowledge, and (c) challenges while completing project tasks. In this paper, we present the survey results from the three categories of technical challenges, their frequencies, importance, and effective support strategies instructors use to alleviate them.**

***Index Terms*—Project based learning, Academic support, Survey, Technical Challenges**

I. Introduction

Through the use of practical, hands-on experiences, project-based learning (PjBL) has been shown to be an effective educational approach that fosters a deeper understanding of concepts in students, sharpens their critical thinking skills, and develops real-world problem-solving ability [1]. However, unless projects are intentionally designed and supported in a way that summons and sustains student motivation, project-based learning is likely to fail to accomplish the above-stated goals [2].

Although there are undoubtedly individual differences, several factors can affect students' motivation [2], including whether they believe the project is worthwhile, whether they believe they have the competence to complete it, and whether they focus on learning rather than on grades. In particular, challenges that students perceive as insurmountable during the project can significantly weaken their motivation. For instance, students often invest more time than necessary in installing and configuring packaged tools, programming languages, or libraries necessary for a project [3], [4], which can lead to a loss of interest and motivation even before commencing the project.

On the other hand, supporting students to overcome such challenges can be troublesome, especially in large classes, online classes, distributed classes, and classes with diversity in student backgrounds and preparedness, as well as diversity in technologies to be used by students.

In this study, we generalize from our own experience and that of other computing instructors by conducting a survey, targeting computing instructors using PjBL in their software engineering and other software-related courses. The survey questionnaire contains closed and open-ended questions regarding (1) technical challenges faced by students, (2) support instructors provided to overcome such challenges, and (3) lessons learned by instructors on the effectiveness of their support. We received a total of 47 responses from diverse instructors in terms of courses, students' years of study, and class sizes. This study answers the following key research questions:

- **RQ1:** What are the unnecessary technical challenges students face in software engineering project-based learning, their frequencies, and their relative importance?
- **RQ2:** What support strategies do instructors employ to address these challenges and the effectiveness of such support?

The rest of the paper is organized as follows: Section II outlines the theoretical framework of this study. Section III details the research methodology that guides this research; We present the survey result in Section IV and Section V. We discuss the results and highlight the study limitations in Section VI. Finally, we conclude the study in VII.

II. Theoretical Framework

Motivating students to learn in school is a topic of great concern for educators today, and motivating students so that they can succeed in school is one of the greatest challenges of education [5]. Motivation is a complex part of human psychology and behavior that influences how individuals choose to invest their time, how much energy they exert in any given task, how they think and feel about the task, and how long they persist at the task [6]. Dörnyei [7] outlines three main sources of motivation:

1) *Course-specific components*: the syllabus, teaching material, teaching method, and learning tasks.
2) *Teacher-specific components*: the teacher's behavior, personality, and teaching style.
3) *Group-specific components*: the dynamics of the learner group.

According to Self-Determination Theory (STD) [8], individuals are motivated when their needs for autonomy, competence, and relatedness are met. In the context of PjBL, project-specific components such as project autonomy, relevance, ownership, complexity, and support can fulfill these needs. For example, giving students control over their projects (autonomy) and ensuring they have the resources and support needed to succeed (competence) can significantly enhance motivation. Several other factors can affect students' motivation in PjBL, including whether they find the project interesting and valuable and whether they focus on learning rather than on grades. In particular, challenges that students perceive as insurmountable during the project can significantly weaken their motivation and confidence [2]. For example, students often invest more time than necessary in installing and configuring packaged tools, programming languages, or libraries necessary for a project [3], [4], which can lead to a loss of interest and motivation even before commencing the project. Another example is when students encounter technical issues caused by software bugs, which can undermine their confidence, subsequently impacting their motivation. Similarly, if projects are too difficult, students can easily lose confidence, as they expect continuous success from one task to another. While projects should ideally be challenging [9] for students, effective support should be provided to ensure they can overcome difficulties and maintain motivation.

However, supporting students to overcome such challenges can be troublesome, particularly in large classes. Large classes often suffer from reduced interaction, low engagement, lack of individual attention, social isolation, and a sense of disconnection [10]. It becomes impractical for instructors to monitor the success of every team project adequately. Moreover, large classes are not the only challenging environments for supporting PjBL. Online classes [11], [12], distributed classes [13], and classes with diverse student backgrounds, preparedness levels, and technology proficiency all present significant hurdles in providing effective support in PjBL.

Hence, instructors must be cognizant of these technical challenges and be equipped with effective methodologies to support students in overcoming them.

III. Research Methodology

In this section, we outline our research methodology, covering our questionnaire design, participant recruitment, data analysis, and the class background information for the survey responses.

A. Questionnaire Design

To address the research questions, we conducted an online survey using the questionnaire shown in Table I (condensed version). From our experiences with running PjBL courses in computing, we carefully designed the questionnaire to capture insights from computing instructors who used PjBL in their courses.

To understand the class background information of the instructors, we started by asking questions related to the type of course they implement PjBL on, the student's year of study, and course class size. These background details will help in contextualizing responses. We categorized the technical challenges into the following three main categories:

- **TC1** - *Challenges students face while installing and configuring software package & tools*: Such as programming language compilers, libraries, frameworks, web and database servers, etc., including any existing projects that require students to set up before starting their projects.
- **TC2** - *Lack of Prerequisite Knowledge*: Some students may lack the prior knowledge and experience needed to undergo the project, such as not being familiar with a programming language, version control systems like Git, or project management tool like Jira.
- **TC3** - *Challenges while completing project tasks*: Challenges students face along the way of completing project tasks, such as debugging, design implementation, etc.

For each of the above categories, we ask instructors how often their students face such challenges, the support they provide, and the effectiveness of such support.

The questionnaire comprised a balanced mix of closed-ended questions, aiming to quantify the frequency and perceived importance of challenges and support strategies, and open-ended questions, intended to capture nuanced qualitative

TABLE I
SHORTENED VERSION OF THE SURVEY QUESTIONNAIRE

Question	**Question Type**
Class Background Information	
In what course(s) do you apply PjBL? Enter course titles, e.g., Software Engineering	Open
What is the size(s) of your PjBL course(s)?	Multiple[Range]
What year are the students typically in?	Multiple[Year]
Challenges in Installing and Configuring Software Packages & Tools	
How often do your students encounter this challenge?	Single[Scale]
Elaborate on the specific packaged tools used or any existing project that your course project was based on.	Open
What support do you provide to students to overcome such challenges?	Multiple+Open
What lessons do you want to share with us on the effectiveness of providing this technical support?	Open
Lack of Prerequisite Knowledge	
How often do your students encounter this challenge?	Single[Scale]
Elaborate on the prerequisite knowledge needed for each project.	Open
What support do you provide to students to overcome such challenges?	Multiple+Open
What lessons do you want to share with us on the effectiveness of providing this technical support?	Open
Challenges while completing project tasks	
How often do your students encounter this challenge?	Single[Scale]
Elaborate on the particular challenges your students encounter when completing project tasks	Open
What support do you provide to students to overcome such challenges?	Multiple+Open
What lessons do you want to share with us on the effectiveness of providing this technical support?	Open
Other Challenges	
Describe the additional technical challenges your students face that are not captured by this questionnaire, the support you provide to students to overcome such challenges, and the lessons you want to share with us on the effectiveness of such support.	Open
Rank the relative importance of the three categories of challenges in determining students' success in PjBL.	Open

perspectives and experiences. The questionnaire was designed and collected using Google Forms.

B. Participant Recruitment

The study undergoes an Institutional Review Board (IRB) review process for approval, ensuring that ethical considerations are thoroughly evaluated and upheld throughout. The questionnaire was targeted at computing instructors who use projects in their software-related courses. We used two methods of participant recruitment, which are:

- Direct email invitation to instructors, with multiple reminders. The email list comprises of 313 email addresses, obtained from research papers in our ongoing work on systematic review on computing PjBL [14]. The review gathered 184 computing-related PjBL papers from ACM Digital Library, IEEE Xplore, EI Compendex, and Scopus, without any period limit. About 50 of the emails bounced back.
- Utilizing online social media platforms like X (formerly Twitter) and Reddit.

Responses are anonymous and informed consent was obtained from all participants. Measures were implemented to safeguard the confidentiality and privacy of their responses.

C. Data Analysis

We received a total of 47 valid responses. The majority of these responses were due to the direct email invitation. The responses were exported to Google Sheets for data analysis and synthesis. A few instructors also reply to the invitation email, expressing their support for this research and interest in knowing more about the research outcomes.

Quantitative responses exported from Google Sheets were plotted on appropriate charts and tables. Additionally, we also extracted relevant and interesting themes from the qualitative responses, which are presented in the study results.

D. Class Background Information of Responses

Below is the result of the collected background information from the survey:

1) Type of Courses: Software engineering courses dominate the courses in the survey, which is our primary target. However, we got responses from other diverse computing courses, as shown in Table II. Introductory and advanced programming courses are also commonly reported. Responses also include software life-cycle courses such as software requirement and analysis, software architecture and design, modeling, software development, software quality assurance, security, and software evolution. Domain-specific courses are also reported, such as machine learning, cloud computing, high-performance computing (HPC), and data science. Other reported computing courses include information systems, research projects, software systems, remote teamwork, design thinking, formal methods, telematics applications and services, data analytics, industry experience, thesis seminars, studio projects, and programming language processors.

2) Student's Year of Study: The courses involve students from diverse study years, ranging from freshmen to graduate students, as shown in Figure 1. Students in their late academic years, such as juniors and seniors, tend to be more involved in PjBL initiatives. This increased involvement may be attributed to several factors, including a deeper understanding of course material and higher proficiency in technical skills due to prior programming experience from earlier courses. While it's less

TABLE II
DISTRIBUTION OF RESPONSES BY PJBL COURSE

PjBL Course	Count [a]
Software Engineering	22
Introductory/Advance Programming	7
Software Development	3
Cloud Computing	2
Data Science	2
Security	2
Software Quality Assurance	2
Software Requirements Analysis and Specification	1
Software Architecture and Design	1
Software Evolution Project	1
Machine Learning	1
High Performance Computing	1
Modeling	1
Mobile Applications Development	1
User Interface Design	1
Other Computing Courses	14

[a]**Note**: The count may not sum to the specified total of 47 because some responses report multiple PjBL courses

common for first-year students to participate in PjBL, it's not unheard of. Instructors incorporate PjBL into their introductory courses to introduce students to real-world problem-solving, teamwork, and critical thinking skills. However, the complexity and scope of projects assigned may be more limited compared to those assigned to upper-level students. Additionally, first-year students may require more support than other senior students due to their lack of experience.

3) Class Size: There are four class size ranges available on the questionnaire: under 25, between 25 and 40, between 41 and 60, and over 60. In computing courses, a class size below 25 can be considered small. Smaller class sizes facilitate better opportunities for students to ask questions, more focused instruction, and more effective assignment feedback. Class sizes between 25 and 40 are somewhat manageable. As the class size goes above 40, the class becomes harder to manage and more difficult to support. As illustrated in Figure 2, most of the class size of responses falls under the chaotic category.

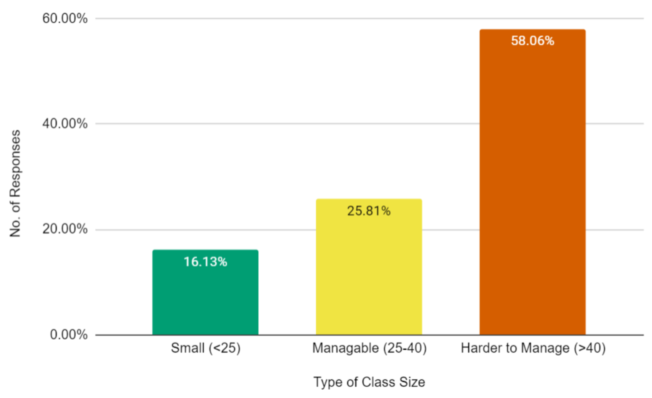

Fig. 2. Distribution of Responses by Class Size

IV. TECHNICAL CHALLENGES

In this section, we will then present the results of the three technical challenge categories students face and their frequencies. Subsequently, we present the relative importance of these categories in determining student success. Finally, we will discuss other reported technical challenges.

A. Frequency of Technical Challenge Categories

In all three categories of technical challenges, we asked instructors how often their students faced them. There were three selection options: never/rarely, occasionally/sometimes, or many/often. The result shows that all three challenges have a frequency of occasionally or often, as shown in Fig. 3.

B. TC1: Installation and Configuration

Challenges students face while installing and configuring software packages & tools are the most reported, with a "many/often" frequency (about 60%). Instructors reported students having challenges with various software tools such as web servers, compilers, and interpreters for programming languages like Python and Java, integrated development environments (IDEs) like Eclipse, PyCharm, and Visual Studio, programming language frameworks like Django and NodeJS,

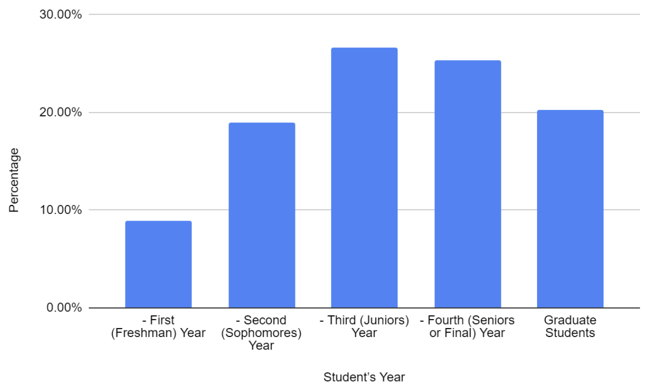

Fig. 1. Distribution of Responses by Student's Year of Study

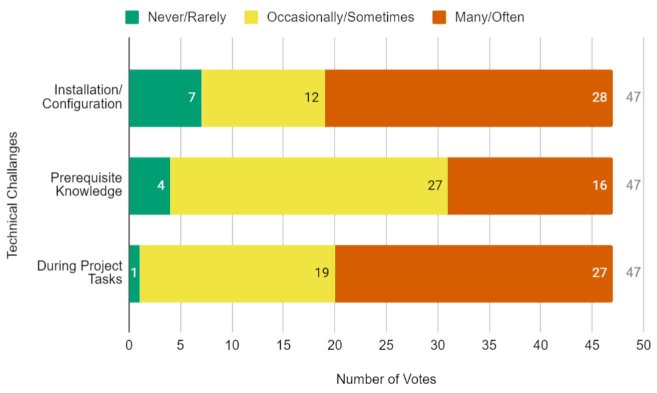

Fig. 3. How Often Each Technical Challenge Category is Faced by Students

and database management systems (DBMS) like MySQL and PostgreSQL. Other tools also reported include Git, Docker, and Kubernetes. The frequency of challenges with these tools can vary depending on various aspects. For example, projects that use a variety of tools are likely to have more challenges. Diversity in operating systems students use and software versions are other factors. Installation instructions provided by instructors are likely not to work for all operating systems and software versions. These issues can be even more problematic to support when students are open to selecting any technology stack, making it difficult for instructors to prepare ahead. Some instructors also note that it is challenging to get students to use the provided materials first because many try to help themselves by searching online or using ChatGPT, Stack Overflow, Google Search, and other resources before consulting the materials. A small number of responses (7) reported rarely having installation and configuration challenges. All of these responses pointed out that their project tools do not need installation, a straightforward installation process, or that they provide effective installation scripts and virtual machines.

C. TC2: Prerequisite Knowledge

Fewer instructors report a lack of prerequisite knowledge than in the other two categories. Occasionally, students lack the knowledge or skills needed to perform the course project. Instructors have reported a lack of prerequisite knowledge like programming basics, object-oriented design, Git/GitHub knowledge, SQL and databases, programming language syntax, software testing, and user interface design. These challenges tend to be more common in senior-level courses, such as capstone projects. This might be because these courses aim to integrate students' prior knowledge into a software engineering project. These challenges can also be prevalent in courses that involve more students from diverse backgrounds.

D. TC3: Project Tasks

Challenges while performing project tasks are also mostly reported with "many/often" frequency. Often, students face several challenges as they complete the assigned project tasks. Instructors have reported challenges while performing tasks such as code debugging, writing test cases, lack of proper project design, code implementation, version control issues, deployment, database issues, and documentation. Students can be easily frustrated by errors, which often take them longer than necessary to solve. Students sometimes also do not carefully read and evaluate error messages. An instructor also pointed out the issue with "feature creeping" [15], where students tend to think more about fancy features than the actual working of the project. Another issue with students arises from their desire to focus solely on generating code, often overlooking the necessary design and analysis efforts.

E. Importance Ranking of Technical Challenge Categories

We also asked instructors to rank the relative importance of overcoming the three categories in determining student's success. The result shows that challenges while performing

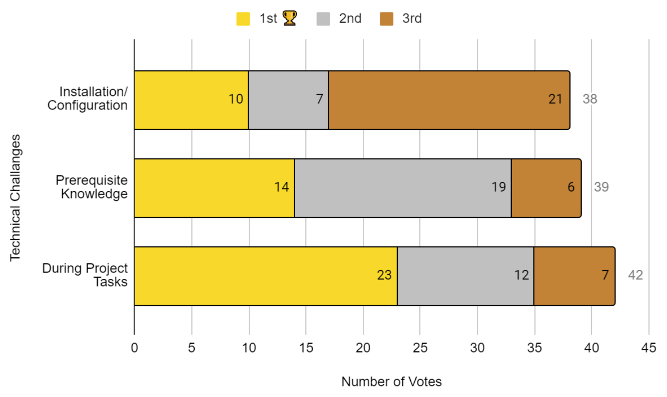

Fig. 4. Relative Importance of the Technical Challenge Categories.
Note: The responses may not sum to the specified total of 47 because of missing votes

project tasks (TC3) are considered the most important, as shown in Fig. 4. These types of challenges are highly unpredictable, especially in open-ended and greenfield projects, where students have numerous solution paths to explore. The variability inherent in such projects can lead to a wide range of technical hurdles that students may encounter, making it difficult for instructors to anticipate and effectively support every potential issue.

Despite being the most frequently reported, challenges in installation and configuration (TC1) are generally perceived as the least important by most instructors. However, there are some varying opinions, with a few instructors considering it the most important. These opinions usually come from courses where the installation and configuration of tools are part of the learning objectives.

F. Other Reported Challenges

We asked instructors to describe any additional technical challenges their students face that are not covered by our three categories. The responses consist of a range of challenges, some of which are technical in nature and others non-technical.

Instructors pointed out that students lack skills on how to organize and manage software engineering team projects. Another issue arises from a lack of design thinking and exploration of solution space. Instructors also highlighted difficulties regarding getting information about the project domain, especially during requirements gathering from an external client or the teacher. Students often lack business etiquette when talking to clients. Matching project requirements and stakeholder needs is also an issue for students. Students struggle with critiquing their own work and conducting thorough validation and verification processes. Writing appropriate documentation for system design, architecture, and implementation is another common challenge. For projects that require special equipment, the availability of this equipment is a challenge, especially if they have to share it with other project groups. Presentation and demonstration of project artifacts have also been reported. With recent advancements in generative AI [16], an instructor reported that students are relying on ChatGPT to do all their project work.

While this study mainly focuses on technical challenges, instructors have pointed out several non-technical challenges that can directly or indirectly contribute to students' inability to succeed. For example, some instructors mention that students struggle with time management. They tend to only work on their project close to the deadline, giving them little to no time to effectively design, implement, test, and debug their work. Students also do not contact instructors, making it impossible for instructors to be aware of the needed support. Team dynamics are another issue, starting with team formation and project allocation. Students struggle to effectively collaborate due to personality clashes, differences in expertise, and a lack of discipline. Communication and scheduling are other major issues, especially in online classes, where students struggle to communicate with their team members effectively and agree on a common time for project meetings. Some students dominate project tasks, excluding others from meaningfully participating in the project, while others disappear (free riders) or fail to contribute, leaving their teammates with additional work. Keeping track of teams with neurodivergent and slow students is also a reported issue.

V. Provided Support

We asked instructors what support they provide to students in each of the three technical challenge categories and the effectiveness of such support. In each case, we listed some support strategies based on experience for instructors to select from. In addition, an open-ended response can be provided. Subsequently, we will highlight the effectiveness of these provided supports and highlight other support strategies from the open-ended responses in each category.

A. Supporting TC1: Installation and Configuration

To alleviate challenges while installing and configuring packaged tools, instructors need to provide appropriate support. Table III lists the initial support strategies and the number of respondents who implement them in their PjBL courses.

1) Online Resources: Providing links to online resources, like written tutorials and videos, is the most common strategy. These resources usually have detailed instructions on how to properly install and configure the necessary packaged tools. This strategy is less challenging for instructors as it does not require their active participation. On the other hand, they might not always work for all students with different operating systems and software versions. Instructors need to update them in a timely manner.

2) Teaching Assistants: The next common support strategy is having a teaching assistant offer support to students in troubleshooting installations. This can be during office hours, via email, or through other communication channels such as Slack and Zoom. This approach can be relatively costly, especially for large classes, considering that teaching assistants have to be hired and trained to support such challenges.

3) Peer Support: Peer support [17] is another common strategy instructors employ, where students get assistance and guidance from their peers who may have successfully completed the necessary installation and configuration of tools. In a team, one or two people (especially the team leader) will typically complete the installation successfully and then assist other members. Peer support fosters collaboration, knowledge sharing, and a sense of community among students. This approach is hardly costly, as instructors just need to encourage students and provide appropriate means of communication for students to share common challenges with specific tools.

4) Laboratory Activity: While not as common compared to the first three strategies, designing a dedicated laboratory activity for the installation and configuration of packaged tools is effective for instructors. It allows students to complete their installation within the laboratory period and get immediate support from the facilitators (i.e., the instructor and/or teaching assistant). However, because of schedules, instructors do not necessarily have the time to conduct such activities during class time, especially in projects with several technology stacks that might require multiple laboratory sessions.

5) Computer with Tools Installed: Some instructors provide a computer with all the necessary software packages and tools installed. While this approach eliminates installation challenges, it prevents students from having a crucial learning opportunity in configuring these tools. Moreover, the devices might not always be fully available to students, either within the project time frame or after. Cost is also an issue because not every institution can afford to provide good computer devices to a large number of students for their project work.

6) Virtual Machines: Virtual machines (VMs) are sometimes provided to students, containing all the necessary software packages and tools installed. This is cost-effective compared to computer devices. However, students still need to install and configure virtualization software, such as Oracle VirtualBox. Using VMs consumes a lot of CPU and memory resources. Not all students with computer devices can seamlessly use them. Sharing the prebuilt VM image can also be a challenge, as it can have a very large storage size.

7) Container Environments like Docker: Providing Docker container environments with prebuilt images is very similar to virtual machines. It is a modern approach and very commonly used in the software engineering industry. They usually take up

TABLE III
Support Provided to Overcome Challenges During Installation and Configuration of Project Tools or Existing Project

Support Provided	**Count** [a]
Share online resources such as written or video tutorials	32
A teaching assistant offers support to students	29
Peer support from fellow students	27
A dedicated laboratory activity to support students	15
Provide a computer with all the necessary software installed	9
Use a container environment such as Docker	9
Share a Virtual Machine with all the necessary software	6

[a]**Note**: The count may not sum to the specified total of 47 because some responses report multiple support strategies

less computer resources compared to VMs and can be easily shared with students. However, as one of the responses noted, installation is part of the learning process, and merely offering Docker containers would miss that opportunity.

8) External Support: In the open-ended sections, some instructors mentioned they have industry mentors, project customers, or other faculties that help students with installation and configuration issues. Considering their extensive experience with these tools, they can be of great assistance. However, a lot of student projects do not involve outside partners, mostly because there are not enough mentors or clients who are willing to help out.

9) Other Support Strategies: Several other support strategies were reported in the open-ended section. Instructors give direct support during office hours, during dedicated class time, on online platforms like Discord and Slack, and a special tutorial for installation tasks. For the simple programming and initial exercises, some instructors developed a browser-based tool that allowed the students to get started with programming right away. Some instructors recommend open-source tools with free student licenses, such as AWS Cloud and Google Cloud, which help bypass the installation and configuration problems. Another instructor provides comprehensive platform-sensitive installer scripts to students, which took many years to learn all of the issues that can go wrong and program the installer scripts to handle them.

B. Supporting TC2: Prerequisite Knowledge

Instructors need to provide appropriate support to students who lack the necessary prerequisite knowledge. Table IV lists the initial support strategies and the number of respondents who implement them in their PjBL courses.

1) Online Resources: Providing links to online resources, like written tutorials and videos, is also the most common strategy when supporting students with a lack of prerequisite knowledge. Some instructors developed a set of video lessons themselves. While this approach relies on students' ability to self-learn from these resources, it does save them time in finding them themselves.

2) Peer Support: Peer support is another common support strategy in overcoming a lack of prerequisite knowledge. Instructors ask and encourage students to get help from team members and organize peer support within the project team's scope. Some instructors believe that most such prerequisites seem very dramatic at first but can quickly be overcome through teamwork with more knowledgeable students. For this reason, it is recommended to manually balance project groups when they are formed, perhaps by using a technical survey at the start of the course.

TABLE IV
SUPPORT PROVIDED TO OVERCOME LACK OF PREREQUISITE KNOWLEDGE

Support Provided	**Count** [a]
Online resources such as written or video tutorials	32
Peer support from fellow students	29
A teaching assistant offers support to students	22
Give additional lectures on prerequisite knowledge	19
A dedicated laboratory activity is designed	10
Ask students to drop the class and take a prerequisite course	2

[a]**Note**: The count may not sum to the specified total of 47 because some responses report multiple support strategy

3) Teaching Assistants: Teaching assistants can always be used in addition to other support strategies to support students with a lack of prerequisite knowledge. Graduate teaching assistants or students who have been through the project previously are well-positioned to provide support within the limits of their own expertise while not doing the project in their place. As previously indicated, this approach can be rather expensive because teaching assistants must be employed and trained to support such challenges.

4) Lecture on Prerequisite Knowledge: Creating additional class lectures on prerequisite knowledge is another method of supporting students with a lack of prerequisite knowledge. It is also a way to remind students of what is required of them and help them refresh their prior knowledge before or during the project activities. With a tight lecture schedule, instructors need to structure the semesters so that students go into the project with a strong baseline of capability. It seems reasonable to ensure that before students are required to apply concepts, skills, and tools to the project, they have been introduced and given practice with them.

5) Laboratory Activity: Some instructors also design dedicated laboratory activities for students to practice required project knowledge. To make sure everyone has the necessary basics, some will make the laboratory session mandatory for all students. Some instructors give students the opportunity to attend one or two workshop-style laboratory activities to gain a better understanding of the requirements process before beginning the project.

6) Drop class and Take Prerequisite Courses: In rare cases (2), instructors suggest that students lacking prerequisite knowledge should drop the class and take a prerequisite course instead. To avoid these, instructors need to fix prerequisite classes so we can ensure they have the right exposure and enforce these prerequisites prior to course enrollment. However, students may have undergone the prerequisite course but may still face challenges. One respondent recommended the implementation of a ramp-up coding pre-assignment at the beginning of the semester, in which students must pass or drop the course. This serves as an effective screening mechanism, despite rare instances of failure.

7) Other Support Strategies: Several other support strategies have been reported by instructors in the open-ended section, such as industry workshops, mentorship from an open-source community, office hours, in-hand tutorials, and support from other faculty members. Extra time can sometimes be given to students with these knowledge gaps. One respondent mentioned it could be beneficial to have project goals that are flexible and can be adjusted based on the student's experience. This may introduce inconsistency or uncertainty in the learning experience, and while it may create a feeling of unfairness among students it does promote equity by providing accommodations for those who need them. Another instructor conducts a short quiz or a hands-on coding activity to assess the entire class's capability and then plans tutorials to incrementally bring them up to speed with the project. Assignments are another strategy employed by instructors. As reported by one of the respondents, some students complain about the intensive six weekly assignments used, but many more find the assignments helpful in completing the project.

C. Supporting TC3: Project Tasks

Instructors need to provide appropriate support to students during project tasks. Table V lists the initial support strategies and the number of respondents who implement them in their PjBL courses.

1) Peer Support: Peer support is one of the most commonly used strategies to overcome project task challenges. One of the purposes of the PjBL course is to encourage students to cooperate with their team members by building a collaborative environment from the start of class so students can help each other. Instructors use collaborative tools like Slack to promote peer support and ask students to use the search tool to check if other teams have encountered similar issues. Providing other central collaboration sites for posting questions and reading answers also helps. Having students do troubleshooting together shows them how real-world problems are solved, even if they don't have the required experience themselves yet. A lot of instructors believe peer support is often helpful.

2) Teaching Assistants: Teaching assistants are also commonly used to support students. Some even pair a teaching assistant with each team to provide constant technical support. Office hours or consultation times are usually arranged so students can contact the necessary channel for assistance. It is usually better if the teaching assistants are graduate students. However, senior students or students who excelled in the course previously as teaching assistants can also be used to offer support. Teaching assistants' competence and enthusiasm for supporting students matter a lot. Again, this approach can be rather expensive, because teaching assistants must be employed and trained to support such challenges.

3) Solution Hints and Technical Support: Instructors also support students with comprehensive hints to problems that get them' stuck, either on-demand or timely. Instructors prefer giving hints to students at this stage and letting them figure out the answers to their problems. Also, because there are no perfect answers for a design or implementation, some believe it is best to give broad directions and let them choose their options. As pointed out by one of the respondents, providing solution hints can be tricky. If you say too much, you give away the solution, and staying vague often increases the confusion rather than helping to overcome it.

TABLE V
SUPPORT PROVIDED TO OVERCOME CHALLENGES WHILE COMPLETING PROJECT TASKS

Support Provided	**Count** [a]
Peer support from fellow students	28
A teaching assistant offers support to students	28
Provide solution hints and other technical support to students	26
Online resources such as written or video tutorials	21

[a]**Note**: The count may not sum to the specified total of 47 because some responses report multiple support strategy

4) Online Resources: Online resources are not as widely used to support project tasks when compared to the other two challenges. Providing links to online resources may not always address the specific problem a student is encountering. However, clarifying that these resources serve as a starting point and encouraging students to explore further on their own can empower them to find tailored solutions.

5) Managed Communication: Challenges with project tasks often arise due to poor time management by students. From the open-ended sections, instructors highlight that students don't contact them on time, and sometimes they even need to do some follow-up. Instructors use managed communication techniques as a result, such as holding weekly or biweekly meetings, reports, presentations, or reviews. Another helpful approach recommended by one respondent is organizing the project into clearly defined milestones and ensuring that each is checked off the list before moving on to the next. We believe that this can be an important strategy, whose effectiveness can be explored in future research. Regular reminders of project schedules and expectations can also be helpful.

6) Other Support Strategies: Instructors have also used other support activities, like giving students special training, industry workshops on design and best practices, and doing special laboratory activities on design comparison and criticism. Some also provide code templates to students and encourage the use of generative AI tools like ChatGPT.

VI. DISCUSSION AND LIMITATION

While the surveyed instructors have identified other challenges in PjBL, we believe that the three technical challenge categories we proposed represent the major ones, with the remaining being non-technical in nature.

Various strategies are available to help students overcome challenges in installing and configuring software packages and tools. Supporting students with installation problems early on saves time and allows them to focus on the project tasks, which are the true learning objectives. When stuck on the setup step, they can be demotivated for a substantial part of the course. It can be effective to make the students aware ahead of time of what technical stacks they need in the upcoming semester, which will provide them with sufficient time to self-study these packages and environments before the project.

While some advocate for direct support to alleviate the lack of prerequisite knowledge and challenges during project tasks, others prefer to encourage students to self-support themselves, expecting a high level of autonomy and promoting self-directed learning, especially given the abundance of tutorials available on the Internet. Through this self-support process, students are expected to learn a lot, gaining a deeper understanding of the practical applications of theoretical concepts.

When students take responsibility for solving their problems, they become more effective learners. Initial struggles are part of the learning process, and by the end of the semester, they will overcome them. Instructors should refine the project goals based on the difficulties encountered by the teams, ensuring the project is appropriately scoped for the available time and the students' experience levels.

We noted two limitations in our study design. First, the survey questionnaire was iteratively designed based on the authors' experiences, which may have introduced bias. Future research should follow established processes from the literature for formulating, refining, and selecting survey questions. Second, while the 47 responses provided initial insights, a larger sample size is necessary to generalize the findings more effectively and perform statistical analysis.

VII. Conclusion

In conclusion, this study gives insight into the critical technical challenges encountered by students in PjBL environments and the strategies employed by instructors to alleviate them. The theoretical framework emphasizes the multifaceted nature of motivation in PjBL, highlighting the significance of project-specific components alongside course, teacher, and group-specific factors. Technical challenges, such as installation and configuration issues, a lack of prerequisite knowledge, and project tasks, significantly impact student motivation and confidence. Effective support strategies are essential to mitigate these challenges, particularly in large classes and other diverse learning environments. Through a comprehensive research methodology involving a survey of computing instructors, this study provides the following two key contributions:

- Categorizing technical challenges faced by students in computing PjBL courses, their frequencies, and the relative importance of overcoming these challenges in determining students' success.
- Highlighting effective strategies to support students in overcoming such technical challenges based on feedback and insights gleaned from a survey of PjBL practitioners, and the effectiveness of such support.

Supporting students demands a lot of time, and all strategies of support have their benefits and limitations. Some surveyed instructors believe that it is important to rely on the drive/eagerness of individual students to solve their own issues. They also believe that students will eventually figure it out, which will make them better software engineers. Others believe that many students easily get stuck and cannot move forward if they are not offered support. Emotional support and managing students' expectations also matter a lot. It is important to get feedback from the students about the difficulties regarding project activities. Soft issues such as time management also play a big factor and can often be harder to resolve than technical ones.

Onward, instructors must remain aware of these technical challenges and be equipped with effective strategies to support students in alleviating them. Instructors need to give students their time, expect them to contribute, and provide time to steer student efforts if they drift. Instructors also need to decide what is best for them based on the course learning objectives and specific learning environment. This will ultimately enhance the success of PjBL initiatives in software engineering and computing education in general.

Acknowledgment

This work was partially supported by the U.S. National Science Foundation Awards DUE-2111318 and DUE-2111294.

References

[1] S. Bell, "Project-based learning for the 21st century: Skills for the future," *The clearing house*, vol. 83, no. 2, pp. 39–43, 2010.
[2] P. C. Blumenfeld, E. Soloway, R. W. Marx, J. S. Krajcik, M. Guzdial, and A. Palincsar, "Motivating project-based learning: Sustaining the doing, supporting the learning," *Educational psychologist*, vol. 26, no. 3-4, pp. 369–398, 1991.
[3] D. Davenport, "Experience using a project-based approach in an introductory programming course," *IEEE Transactions on Education*, vol. 43, no. 4, pp. 443–448, 2000.
[4] A. Adorjan and M. Solari, "Software engineering project-based learning in an up-to-date technological context," in *2021 IEEE URUCON*. IEEE, 2021, pp. 486–491.
[5] J. Filgona, J. Sakiyo, D. Gwany, and A. Okoronka, "Motivation in learning," *Asian Journal of Education and social studies*, vol. 10, no. 4, pp. 16–37, 2020.
[6] R. Ramli, "The effect of learning motivation on student's productive competencies in vocational high school, west sumatra," *International Journal of Asian Social Science*, vol. 4, no. 6, pp. 722–732, 2014.
[7] Z. Dörnyei and E. Ushioda, *Teaching and researching motivation*. Routledge, 2021.
[8] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being." *American psychologist*, vol. 55, no. 1, p. 68, 2000.
[9] S. Aldabbus, "Project-based learning: Implementation & challenges," *International Journal of Education, Learning and Development*, vol. 6, no. 3, pp. 71–79, 2018.
[10] C. Mulryan-Kyne, "Teaching large classes at college and university level: Challenges and opportunities," *Teaching in higher education*, vol. 15, no. 2, pp. 175–185, 2010.
[11] M. Adil, I. Fronza, and C. Pahl, "Software design and modeling practices in an online software engineering course: The learners' perspective." in *CSEDU (2)*, 2022, pp. 667–674.
[12] S. Motogna, D. M. Suciu, and A.-J. Molnar, "Exploring student challenges in an online project-based course," in *Proceedings of the First International Workshop on Designing and Running Project-Based Courses in Software Engineering Education*, 2022, pp. 10–14. [Online]. Available: https://doi.org/10.1145/3524487.3527361
[13] J. D. Herbsleb and D. Moitra, "Global software development," *IEEE software*, vol. 18, no. 2, pp. 16–20, 2001.
[14] Anonymous, "Systematic literature review on project-based learning in computing education," 2024, unpublished Manuscript.
[15] B. Elliott, "Anything is possible: Managing feature creep in an innovation rich environment," in *2007 IEEE International Engineering Management Conference*. IEEE, 2007, pp. 304–307.
[16] S. Bengesi, H. El-Sayed, M. K. Sarker, Y. Houkpati, J. Irungu, and T. Oladunni, "Advancements in generative ai: A comprehensive review of gans, gpt, autoencoders, diffusion model, and transformers." *IEEE Access*, 2024.
[17] S. Mead, D. Hilton, and L. Curtis, "Peer support: a theoretical perspective." *Psychiatric rehabilitation journal*, vol. 25, no. 2, p. 134, 2001.