Bridging the Gap in Software Engineering Education: Analyzing Learning Behaviors, Challenges, and Resource Effectiveness in Full-Stack Development

Swetha Rajeev

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Application

Dwayne C Brown, Chair

Mohammed S Seyam

Onyeka Emebo

May 13, 2025

Blacksburg, Virginia

Keywords: Full-stack development, Software tools, Learning Resources, Curriculum development, Software engineering education.

Copyright 2025, Swetha Rajeev

Bridging the Gap in Software Engineering Education: Analyzing Learning Behaviors, Challenges, and Resource Effectiveness in Full-Stack Development

Swetha Rajeev

(ABSTRACT)

In the rapidly evolving field of software engineering, the proficiency with which developers wield their tools significantly impacts their productivity and adaptability. Despite the critical role of these tools, there is a notable gap in research regarding the time investment required for learning them and the effectiveness of various learning resources. This study aims to bridge this gap by investigating the most commonly used full-stack development tools, quantifying the time developers dedicate to mastering these tools, and identifying the factors influencing their learning choices and durations. Through a comprehensive survey and analysis, we explore how developers and CS students discover new tools, decide which ones to learn, and the challenges they encounter during this process. We further evaluate the effectiveness of different learning resources, from formal education to self-directed online learning, in fostering tool proficiency. The findings aim to inform the development of a targeted curriculum that can equip engineers with the necessary skills more efficiently, thereby enhancing the overall quality of software engineering education and practice. Building on these insights, this research introduces "LearnMate," an interactive learning tool designed to provide curated learning paths, recommend efficient learning resources, and offer structured guidance tailored to different learning preferences. By leveraging survey findings, LearnMate aims to assist both students and professionals in acquiring proficiency with essential full-stack technologies in a structured and effective manner. This research not only contributes to

academic knowledge but also serves as a practical guide for curriculum developers, educators, and practitioners in the software engineering domain. By combining the findings from the survey recommendations with an interactive learning tool, this study paves the way for more effective and efficient learning pathways in the ever-changing technological landscape, ultimately improving software engineering education and professional training.

Bridging the Gap in Software Engineering Education: Analyzing Learning Behaviors, Challenges, and Resource Effectiveness in Full-Stack Development

Swetha Rajeev

(GENERAL AUDIENCE ABSTRACT)

In today's world, almost everything we do is supported by some form of digital technology, from the apps on our phones to the websites we browse daily. Behind these web applications and digital interfaces are full-stack developers — software engineers who work on both the front-end (the visible, user-facing part of a website or app) and the back-end (the server, database, and logic that make it function). These developers use specialized tools to build and improve digital experiences. However, selecting the right tools and learning to use them efficiently presents a significant challenge, as developers must navigate a vast and rapidly evolving technology landscape with limited guidance on which resources are most effective. Our research delves into the realm of full-stack development to identify the essential tools developers rely on, how much time they typically spend learning to use these tools effectively, and the factors influencing their learning decisions. We also explore the hurdles they face in this learning process and the resources they find most helpful, such as online courses, tutorials, or community forums.

Beyond analyzing these challenges, this study leads to the development of a learning tool designed to assist developers in selecting the most effective resources and structuring their learning process. This tool will offer personalized learning paths based on individual needs and preferences, helping both students and professionals acquire full-stack development skills

more efficiently. By understanding these aspects, we hope to make it easier for current and future software engineers to pick up the skills they need quickly and efficiently. Ultimately, our research contributes to making software education more accessible, structured, and aligned with industry needs, ensuring that the digital world continues to grow and improve in ways that benefit us all. This is not just about coding and programming; it's about understanding the backbone of our digital existence and ensuring that those who build it are well-equipped for the task.

Dedication

To my husband - for being my greatest source of calm, strength, and belief through every high and low. To my parents - for shaping my dreams with their love, sacrifices, and unshakeable faith in me. This work stands as a reflection of everything you have given me, and everything you inspire me to become.

Acknowledgments

I would like to sincerely thank Dr. Chris Brown, my advisor, for his unwavering support, guidance, and encouragement throughout this research journey. His mentorship has been instrumental in shaping both this work and my academic growth. I am also grateful to my committee members, Dr. Mohammed Seyam and Dr. Onyeka Emebo, for their time and participation on my committee. A special thanks to Megan Thomas for her valuable assistance over an entire semester, especially in supporting the study portion of this project. I would also like to extend my appreciation to the research group "Code World No Blanket," led by Dr. Chris Brown. Their thoughtful feedback and discussions at various stages of this project helped refine and strengthen my work. Finally, I am deeply thankful to everyone who encouraged and supported me along the way.

Contents

List of Figures **xi**

1 Introduction **1**

2 Review of Literature **5**

2.1 Gaps in Software Engineering Education 5

2.2 Decision-Making in Learning 6

2.3 Evolving Skill Requirements 7

3 Survey Methodology **10**

3.1 Survey Design, Population, and Data Collection Process 10

3.2 Pilot Testing 12

3.3 Data Analysis 12

3.3.1 Quantitative Data 13

3.3.2 Qualitative Data 15

3.4 Ethical Considerations 19

4 Results **20**

4.1 Demographics 20

4.2 Tool Usage and Learning Patterns 21

4.3 Time Investment 23

4.4 Challenges and Solutions 24

4.5 Effectiveness and Preferences of Learning Resources 26

4.6 Collaboration and Community Influence 28

5 Discussion **30**

5.1 Interpretation of Findings 30

5.1.1 RQ1: Essential Tools and Influences 30

5.1.2 RQ2: Time Investment 32

5.1.3 RQ3: Effective Learning Resources and Challenges 33

5.2 Contributions to the Field 35

5.3 Curriculum Development and Impact on Education 37

5.4 LearnMate: From Research to Application 41

5.4.1 LearnMate User Study: Evaluating Usability and Learning Impact . 50

6 Limitations **55**

7 Future Work **57**

8 Conclusions **59**

Bibliography **60**

Appendices **64**

Appendix A Surveys **65**

A.1 Survey for professionals 65

A.2 Survey for students 69

List of Figures

4.1 Learning Resources for Professionals 22

4.2 Learning Resources for Students 22

4.3 Time spent by professionals on learning new tools. 23

4.4 Time spent by students on learning new tools. 23

4.5 factors to consider when choosing a tool (Professionals) 27

5.1 Homepage of the LearnMate application. 42

5.2 Topic selection screen after choosing "Learn". 43

5.3 Learning interface for Frontend development. 43

5.4 Schedule Your Learning Path - initial screen. 44

5.5 Dropdown for selecting a time period. 45

5.6 Dropdown for selecting a learning path. 45

5.7 Example learning path with progress tracking for 1-week frontend plan. . . . 46

5.8 Form interface for submitting new learning content. 46

5.9 LearnMate AI input interface. 47

5.10 AI-generated response example for a CI/CD pipeline query. 47

A.1 Professional Survey – Page 1 65

A.2 Professional Survey – Page 2 66

A.3 Professional Survey – Page 3 66

A.4 Professional Survey – Page 4 67

A.5 Professional Survey – Page 5 67

A.6 Professional Survey – Page 6 68

A.7 Professional Survey – Page 7 68

A.8 Professional Survey – Page 8 69

A.9 Student Survey – Page 1 70

A.10 Student Survey – Page 2 70

A.11 Student Survey – Page 3 71

A.12 Student Survey – Page 4 71

A.13 Student Survey – Page 5 72

A.14 Student Survey – Page 6 72

A.15 Student Survey – Page 7 73

A.16 Student Survey – Page 8 73

Chapter 1

Introduction

In the rapidly evolving field of software engineering, development tools are not just accessories; they are integral to a developer's workflow. The proficiency with which developers use their tools significantly impacts their productivity [5, 21] and efficiency. For instance, integrated development environments (IDEs) such as Visual Studio or IntelliJ IDEA provide a comprehensive suite of tools that streamline the coding process[1]. A developer's ability to leverage IDE features such as intelligent code completion, automated refactoring, and seamless debugging can impact the time and effort required to implement complex software solutions [23]. However, the landscape of software development is constantly evolving, requiring engineers not only to master existing tools but also to continuously adapt to new and emerging technologies. Despite the critical role of these tools, there is limited research on how developers learn them, the time investment required, and the effectiveness of different learning resources.

Full-stack development encapsulates the comprehensive skill set required to build end-to-end software solutions, covering both front-end and back-end technologies. Full-stack developers work across the entire software development lifecycle, from designing the user interface to implementing the server-side logic and database integration. According to the Stack Overflow Developer Survey 2022[2] , full-stack developers make up the largest proportion of the developer community, accounting for 46. 82% of the respondents, followed by back-end

[1] https://www.xcubelabs.com/blog/a-comprehensive-guide-to-integrated-development-environments-ides/
[2] https://survey.stackoverflow.co/2022

developers (43. 38%) and front-end developers (25. 96%). This significant representation highlights the need to understand the learning behaviors, challenges, and tool proficiency of full-stack developers. By focusing on full-stack development, this study can provide broadly applicable insights to the wider software engineering community given the versatility and demand for full-stack skills in the industry.

The tools used in software development are essential components that enable full-stack developers to bring their ideas to life, impacting every stage from concept to deployment. Full-stack developers need to possess knowledge of both front-end technologies like HTML/CSS, React, Angular or Vue.js and back-end frameworks such as Node.js,Django or Spring Boot, making their tool usage and mastery more comprehensive and demanding. They must be proficient with tools for designing user interfaces, implementing server-side logic, and integrating databases, among other tasks. Recognizing the importance of these tools, this study seeks to investigate the most commonly used full-stack development tools, quantify the time developers dedicate to mastering them, and identify the factors influencing their learning choices and durations.

However, beyond simply analyzing these aspects, this research also introduces a learning-driven software engineering tool called LearnMate, designed to enhance the learning experience for both students and professionals. This tool is designed to address the common challenges faced by developers when learning new technologies by providing structured guidance tailored to individual needs. It offers personalized learning paths that help learners navigate the complexities of full-stack development by breaking down skills into manageable steps based on their experience level and career goals. Additionally, the tool incorporates recommendation systems that suggest the most effective educational resources, including interactive tutorials, official documentation, and video-based courses. By analyzing devel-

opers' learning behaviors, the tool optimizes time investment by suggesting efficient learning strategies, ensuring that users spend less time searching for resources and more time actively learning. Ultimately, this tool serves as an intelligent guide for both students seeking structured educational pathways and professionals looking to stay up-to-date with industry trends. By integrating data-driven insights with practical, real-world applications, it aims to bridge the gap between traditional education and the constantly evolving demands of software engineering[7], fostering a more effective and personalized approach to skill acquisition.

To guide this study, we explore the following research questions:

RQ1: What are the essential tools in full-stack software development, and what influences students and professionals when choosing which of these tools to learn?

RQ2: How much time do students and professionals invest in learning new software tools, and what factors affect this time investment?

RQ3: What learning resources do students and professionals find most effective and what challenges do they face when learning new tools for their projects or work?

With these research questions, this study aims to comprehensively explore how software engineers, including both students and professionals, approach learning essential tools in full-stack development. By examining their learning choices, time investment, and challenges faced, we aim to provide actionable insights for educators, curriculum developers, and industry professionals.

Through a comprehensive survey and analysis, we aim to understand how developers discover new tools, decide which ones to learn, and the challenges they face in this process. Moreover, this research also evaluates the effectiveness of different learning resources, ranging from formal education (such as university courses and bootcamps) to self-directed online learn-

ing(including interactive tutorials, documentation, and video-based courses).Our findings will contribute to the development of a targeted curriculum and targeted learning methodologies that equip software engineers with the necessary skills more efficiently. In their seminal work, Veena and Shastri (2013) [13] highlight the diversity of learning preferences, underscoring the need for tailored educational approaches. Our findings will contribute to this dialogue by focusing specifically on tool proficiency within full-stack development. By informing the development of a targeted curriculum, we intend to equip software engineers with the necessary skills more efficiently, thereby enhancing the overall quality of software engineering education and practice. Moreover, this knowledge is invaluable for corporate settings, where the optimization of training resources is of paramount importance [22]. This thesis provides guidance on the best practices in learning and the optimal time allocation for training initiatives.

This research extends beyond academic inquiry; it serves as a practical guide for curriculum developers, educators, and industry practitioners. By introducing LearMate, it directly addresses the need for more structured and effective learning pathways in software engineering. In a field where the only constant is change, providing structured learning guidance is essential for helping developers keep pace with evolving technologies. This study, along with the proposed tool, aims to bridge the gap between academia and industry, empowering developers to learn, adapt, and excel in the ever-changing technological landscape.

Chapter 2

Review of Literature

2.1 Gaps in Software Engineering Education

Research has long acknowledged the disconnect between the skills taught in educational settings and those required in the software engineering industry[9]. This discrepancy is particularly pronounced in the realm of full-stack development [8, 18], where the rapid evolution of technologies necessitates a continuous learning approach. Building on the work of Veena and Shastri (2013) [13], which calls attention to the diverse learning preferences, our review explores how different learning models address or fail to address this gap.

Oguz et al.[14] conducted an intensive study into the gaps between industry and education and concluded that the gaps were here to stay due to the fact that industry will continue to innovate. This persistent gap is further corroborated by Garousi et al.[9], who conducted a systematic literature review of 33 papers relating to the gap between software engineering education and industry needs. Their analysis revealed that out of 15 knowledge areas defined in the Software Engineering Body of Knowledge (SWEBOK), 11 showed large gaps and high relevance when considering only reports from the last five years. This indicates a widening gap between education and industry needs, underscoring the urgency of addressing this issue.

The challenge of keeping pace with industry demands is further complicated by the rapid

evolution of technologies. Dobslaw et al. [6] used a text-analysis tool to compare keywords from higher education course syllabi and job posts, finding that demand for cloud and automation technologies such as Kubernetes and Docker is rising in job ads but not as much in higher education syllabi. This mismatch highlights the need for more agile curriculum development processes that can quickly adapt to emerging industry trends.

While previous research has highlighted the persistence of these gaps and the need for continuous learning, our study uniquely focuses on the specific tools and learning resources used by full-stack developers. We aim to quantify the time investment required for mastering these tools and identify the factors influencing developers' learning choices and durations. Unlike earlier studies that broadly discuss educational gaps, our research provides a detailed examination of the most commonly used full-stack development tools and the effectiveness of various learning resources, thereby offering actionable insights for curriculum developers and educators to design more targeted and efficient learning pathways.

2.2 Decision-Making in Learning

The act of selecting which software engineering tools to learn is a multifaceted decision that can significantly impact a developer's career trajectory [2]. This reinforces the complexity of this decision-making process, highlighting the myriad of factors at play, including prevailing industry trends, project-specific needs, and individual career objectives.

A seminal study by Murphy-Hill et al.[12] explores the social dynamics of tool discovery, finding that peer interactions play a crucial role in how developers come across new tools . Their research presents a nuanced picture of the learning ecosystem, where developers often rely on their peers for knowledge acquisition, despite the presence of formal educational resources. This peer-to-peer exchange, while occasionally serendipitous, is shown to have a

profound influence on the tools that developers choose to incorporate into their skill set.

Building on this, Robillard et al. [15] conducted a study on how developers learn APIs, which are crucial tools in software development. They found that developers often struggle with learning new APIs due to insufficient documentation and examples. This highlights the importance of comprehensive learning resources in the decision-making process when choosing which tools to learn.

Furthermore, a study by Begel et. al [2] work on the struggles of new college graduates in their first software development job further emphasizes the importance of practical, hands-on experience with tools used in industry settings. These insights are invaluable, offering a window into the organic learning processes that often go unnoticed in structured educational environments. They underscore the need for educational programs to not only focus on technical skills but also on developing the ability to make informed decisions about tool adoption and learning strategies.

The research underlines the necessity for learning resources and curricula that not only provide technical knowledge but also foster an environment conducive to social learning and knowledge sharing. The literature offers insights into how developers prioritize learning objectives and how these decisions impact their professional development.

2.3 Evolving Skill Requirements

The rapid evolution of technology in the software industry creates ongoing challenges for academic programs trying to keep pace with industry needs [9].This technological acceleration has fundamentally transformed the skill requirements for software engineers, particularly in full-stack development, where both front-end and back-end expertise are essential [18].

Recent studies highlight the growing complexity of full-stack development [8]. In today's fast-paced environment, developers must be proficient in a range of technologies that span various layers of the software stack. Front-end development increasingly involves complex frameworks such as React, Angular, and Vue.js, which enable the creation of dynamic, responsive interfaces but require mastery of JavaScript and advanced design principles. Simultaneously, back-end development has expanded beyond traditional server management to include cloud infrastructure, microservices, and containerization tools like Docker and Kubernetes.

The integration of artificial intelligence and advanced manufacturing technologies has created new demands for software engineering education [10]. Modern software engineers must understand not only traditional programming concepts but also emerging technologies like cloud computing, artificial intelligence, and industrial automation [1]. This trend underscores the need for software engineers to continually learn new frameworks and adapt to emerging technologies.

Moreover, the rise of cloud-based and serverless architectures has led to an increased demand for skills in deploying and managing applications on various platforms [3]. Full-stack developers today are often required to have knowledge of DevOps practices, which integrate development with continuous deployment and monitoring [6]. This shift highlights a gap in current software engineering curricula, as many programs still focus primarily on programming languages and basic development skills without sufficient emphasis on deployment, scalability, and performance optimization.

Another dimension of evolving skill requirements is the integration of artificial intelligence (AI) and machine learning (ML) into full-stack applications [8, 19]. Recent research indicates that the swift development of AI, particularly Large Language Models (LLMs), is drastically altering various industries and necessitating a major change in the way software engineering is taught. As a result, full-stack developers are increasingly expected to understand the

basics of AI and ML to integrate data-driven features into their applications.

To address these evolving demands, several researchers recommend an industry-aligned, flexible curriculum structure [16]. Studies suggest that software engineering education must adapt and evolve to prepare software engineers for the emerging AI-driven environment [17]. This includes implementing industry-driven competency standards and incorporating real-world project experiences. The literature indicates that while academic programs have made strides in incorporating some new technologies, there remains a gap in bridging fundamental knowledge with hands-on experience in industry-standard tools.

Chapter 3

Survey Methodology

This section outlines the methodology employed in this research to investigate the learning patterns, resource effectiveness, and tool proficiency among full-stack software developers. The study aims to quantify the time investment required for learning development tools and to identify the factors influencing learning choices and durations.

3.1 Survey Design, Population, and Data Collection Process

The survey was designed as a cross-sectional study aimed at capturing a snapshot of current practices, preferences, and challenges related to learning and using web development tools among full-stack developers. To achieve comprehensive data coverage, two distinct surveys were developed: one targeted at students and the other at working professionals with experience in full-stack development. Both surveys shared a structured format, containing both closed-ended questions for quantitative analysis and open-ended questions for qualitative insights. The professional survey consisted of 21 questions, while the student survey had 22 questions. These surveys covered core areas including demographics, tool usage, learning resources, time investment, learning challenges, and resource effectiveness, with slight variations to tailor the questions to each group's specific context. For example, the professional

survey gathered insights on challenges related to project-based learning at work, while the student survey explored learning experiences in academic projects and internships.
The survey included questions on:

- Demographic Information: Age, gender, educational background, and professional experience.
- Tool Usage: Frequency and duration of use of various development tools.
- Learning Resources: Types of resources used (e.g., formal education, online courses, tutorials, community forums).
- Time Investment: Estimated time spent learning each tool.
- Learning Challenges: Difficulties encountered during the learning process.
- Resource Effectiveness: Perceived effectiveness of different learning resources.

To capture diverse perspectives, the professional survey was distributed through LinkedIn, email, and professional networks, particularly reaching out to colleagues from previous employment and friends working in various tech companies. Meanwhile, the student survey was shared with computer science graduate and undergraduate students at Virginia Tech via university email, relevant listservs, and classroom Slack channels. This approach ensured that data from both student and professional viewpoints were included, enriching the dataset with a wide range of experiences and insights.

A combination of convenience and purposive sampling was employed to ensure representation across different career stages and levels of experience in full-stack development. Convenience sampling involved leveraging accessible groups through personal and professional networks, while purposive sampling focused on individuals with specific backgrounds in software and

full-stack development. This combination was essential to capturing relevant data aligned with the study's focus.

The surveys were administered online through the QuestionPro platform, allowing easy access for participants and broad reach. The design prioritized user convenience, with each survey intended to be completed within 10-15 minutes, facilitating robust participation across both groups. This inclusive and varied distribution method enriched the study with comprehensive insights reflective of the full-stack development field at different career stages.

3.2 Pilot Testing

Before the full launch, the survey underwent a pilot test with a small group of 8-10 participants from the software engineering reading group at Virginia Tech. This group was chosen because the members have industry experience/ full-stack experience and substantial knowledge in software engineering, making them representative of the target audience. The pilot test aimed to validate the clarity and relevance of the survey questions and assess the timing required to complete the survey. Feedback from this phase was used to refine the questionnaire, ensuring that the questions were unambiguous and effectively aligned with the research objectives. Adjustments were made based on participants' suggestions to improve question clarity, remove any ambiguities, and ensure that the survey effectively captured the required data.

3.3 Data Analysis

The analysis for this study involved both quantitative and qualitative methods to ensure a comprehensive understanding of the collected data.

3.3.1 Quantitative Data

The survey was conducted using an online platform to ensure a broad reach and easy data collection. Closed-ended questions provided structured responses that could be statistically analyzed to derive meaningful insights. These questions were carefully designed to address the research objectives and included multiple-choice questions and single-response options to capture both demographic details and insights into key topics such as tool usage, resource preferences, time investment, and learning challenges.

Types of Closed-Ended Questions

- **Multiple Choice Question:**Most of the questions in both surveys utilized MCQs, where respondents selected one or more options from a predefined list. In the professional survey, 9 out of 21 questions were multiple-choice, while in the student survey, 12 out of 22 questions were multiple-choice. These questions gathered critical data points, such as demographic information (e.g., age, education, experience level), challenges faced during tool adoption, and factors influencing the choice of learning resources.
  - Example: ”What resources do you use to learn new tools?” with options like ”Online tutorials,” ”Video courses,” and ”Documentation.”

Statistical Analysis Techniques

The collected quantitative data were analyzed using various statistical methods to extract insights and answer the research questions:

- **Descriptive Statistics:**

  - **Frequencies and Percentages:** These were used to summarize responses, such as the percentage of participants preferring online tutorials or spending a specific amount of time on learning tools.
    * **Example:** "47.83% of professionals reported spending 1-3 hours weekly on learning new tools."
  - **Central Tendency Measures:** Mean and median values were calculated to identify the average time spent learning tools by students and professionals.

- **Inferential Statistics:**
  - **Two-Sample t-Test:** Applied to compare the time investment between students and professionals, providing statistical evidence of whether their time commitment significantly differed.
    * **Result:** The t-test ($t(106) = -1.03, p = 0.303$) revealed no significant difference, suggesting that both groups invest similar amounts of time in learning.

Visualization

Graphical methods were employed to present the analyzed data effectively:

- **Bar Charts:** Depicted comparisons of popular tools and preferred learning resources among the two groups.
- **Pie Charts:** Illustrated the distribution of participants' responses across different categories, such as time investment.

3.3.2 Qualitative Data

Open-ended questions allowed participants to elaborate on their experiences, providing deeper insights into their learning processes and challenges. The qualitative data collection focused on understanding the nuances behind the quantitative responses and uncovering rich, detailed insights. The qualitative questions targeted specific aspects of learning, including challenges, resource preferences, and collaboration. They were formulated to gather detailed narratives, which could help identify patterns and themes in the learning behaviors of both students and professionals.

Examples of Open-Ended Question

For Professionals:

- **Challenging Tools and Technologies:** *"Have you found any tools or technologies particularly challenging to master? If yes, why?"*
  This question aimed to uncover difficulties faced by professionals in mastering specific tools, offering insights into areas that require improved training or resources.

- **General Challenges:** *"What are the biggest challenges you face as a full-stack developer?"*
  This sought to identify overarching difficulties in the professional environment, such as time management, technological adaptation, or organizational constraints.

- **Web Frameworks:** *"What Web Frameworks have you used and what is the best resource to learn them?"*
  This question explored the tools professionals commonly rely on and the resources they found most effective for learning.

- **Front-End Tools:** *"What front-end tools have you used or are you using currently, and what is the best resource to learn them?"*

  This focused on understanding tool preferences for front-end development and identifying the resources found most effective.

- **Curriculum Recommendations:** *"Based on your experience in the industry, do you believe that the curriculum for aspiring developers should be altered?"*

  This question gathered feedback on how educational programs could be improved to align better with industry requirements.

For Students:

- **Tool Usage in Projects:** *"Which full-stack development tools have you used for your projects or internships?"*

  This question aimed to capture the tools that students have practical experience with, providing a perspective on current academic exposure.

- **Effective Learning Resources:** *"Did you find any particular learning resources or tutorials more helpful than others in understanding these tools? If yes, what made them effective for you?"*

  This question sought to understand what makes certain learning resources more effective from the student perspective.

- **Switching Tools:** *"Were there any instances where you had to switch to alternative tools during your projects or internships? If yes, please provide some examples."*

  This question explored adaptability and challenges associated with transitioning between tools.

- **Collaboration Experiences:** *"Have you ever collaborated with other students or*

professionals in using these tools? If yes, how did this collaboration affect your learning experience?"

This question sought to understand the role of teamwork and knowledge-sharing in enhancing the learning process.

- **Effective Learning Resource Characteristics:** *"In your opinion, what makes a learning resource (e.g., a tutorial, course, or documentation) effective? What elements contribute to an engaging and informative learning experience?"*

  This question sought to gather insights on what makes learning resources effective.

- **Suggestions for Peers:** *"What advice would you give to other undergraduate students who are about to embark on projects or internships that involve learning new tools?"*

  This question sought to collect practical tips and guidance for students preparing for real-world applications of their skills.

Data Analysis Process

The responses to these open-ended questions were analyzed using thematic analysis, a qualitative method that involves identifying, analyzing, and reporting patterns or themes within the data. The process included the following steps:

- **Familiarization with Data:**The qualitative responses were reviewed multiple times to gain a deep understanding of the content and context.

- **Initial Coding:**Key phrases and statements were highlighted, and initial codes were assigned to identify recurring ideas. For example, phrases related to resource effectiveness were coded under themes like "clarity," "hands-on practice," or "interactivity."

- **Theme Development:** The initial codes were grouped into broader themes, such as ”Challenges in Learning,” ”Collaboration Benefits,” and ”Resource Preferences.”

- **Cross-Comparison:** The data from professional and student surveys were compared to identify overlapping themes and unique perspectives. For instance, professionals emphasized workplace-related challenges, while students focused more on academic constraints.

- **Validation:** The themes were reviewed to ensure they accurately represented the data, and illustrative quotes were selected to highlight key findings. For example, one student mentioned, “Collaborating with peers helped me tackle complex projects faster,” under the theme of “Collaborative Learning.”

The mixed method approach of analyzing the qualitative data offered nuanced insights into the diverse learning experiences of full-stack developers, shedding light on various challenges, preferences, and collaborative strategies. Participants frequently highlighted time constraints, insufficient documentation, and steep learning curves as significant obstacles in mastering new tools. Interactive and step-by-step tutorials emerged as a preferred resource, praised for their effectiveness in facilitating practical learning. Additionally, many respondents emphasized the importance of teamwork and peer support in overcoming challenges and enhancing their understanding of tools. This qualitative analysis complemented the quantitative findings, providing a comprehensive view of the learning processes, challenges, and preferences among full-stack developers.

3.4 Ethical Considerations

This research strictly adhered to ethical guidelines to ensure the rights, safety, and well-being of all participants were protected. Institutional Review Board (IRB) approval was obtained prior to the initiation of the study. The IRB review process ensured that the research design, survey content, data collection methods, and consent procedures met established ethical standards and minimized any potential risks to participants.

The following ethical measures were implemented during the study:

- **Informed Consent:** Participants were provided with an informed consent form that clearly outlined the purpose of the study, its objectives, and the nature of their involvement. The form also included information about the voluntary nature of participation, the approximate time required to complete the survey, and the fact that they could withdraw at any time without any consequences.
- **Confidentiality:** Data was stored securely to protect participants' privacy.
- **Right to Withdraw:** Participants were allowed to withdraw from the study at any time without any consequences.
- **Minimization of Risks:** The survey design minimized potential risks by avoiding sensitive or intrusive questions and ensuring the language used in the surveys was neutral and non-discriminatory. Participants were also provided with contact information for the research team should they have any questions, concerns, or complaints about the study.

By incorporating these ethical practices and securing IRB approval, this study prioritized transparency and data protection. These measures ensured compliance with established ethical research standards and reinforced the integrity and reliability of the findings.

Chapter 4

Results

The surveys aimed to capture current practices, preferences, and challenges associated with learning and using full-stack development tools among full-stack developers. Two sets of surveys were conducted: one among professionals and the other among students at Virginia Tech.

4.1 Demographics

The professional survey included responses from 71 participants, all of whom completed the survey, resulting in a 100% completion rate. The respondents had varying levels of experience, with the majority ie 31 participants(45.71%) having 1-2 years of experience, followed by 22 participants ie, (31.43%) with 3-5 years of experience. A smaller percentage had less than 1 year of experience i.e., 11 participants(15.71%), and 5-10 years of experience i.e., 5 participants (7.14%) of experience. None of the respondents reported more than 10 years of experience.

On the other hand, the student survey collected 63 responses, with 62 fully completed, resulting in a 98.41% completion rate. The respondents' academic levels varied, with participant (1.72%)being a freshmen, 2 participants (3.45%) sophomores, 7 participants (12.07%) juniors, 20 participants(34.48%) seniors, and the majority i.e., 28 participants (48.28%) being master's students. In addition to academic background, 70.69% i.e., 41 student respondents

reported having participated in internships or projects related to their field of study. These internships and projects provided students with practical exposure to tools and technologies commonly used in industry. Notable companies where students interned include Collins Aerospace, Amazon, Deloitte, ServiceNow, Cognizant, Peraton and Virginia Tech . This diverse professional experience adds depth to the analysis of the student survey responses, as it reflects varying levels of real-world application of full-stack development tools and resources.

4.2 Tool Usage and Learning Patterns

Professionals and students exhibited distinct patterns in their tool usage and learning. Among professionals, the most common full-stack tools were React, Node.js, Flask, Django, AWS services, PostgreSQL, and MySQL. Professionals predominantly favored video courses and online tutorials (49.62%) as their primary learning resources, followed by coding bootcamps (15.27%), social media platforms (16.03%), books (10.69%), and other sources like ChatGPT, Stack Overflow, and project documentation (8.40%).

Students, on the other hand, frequently used VS Code, React, Django, Spring Boot, MySQL, MongoDB, and Node.js. They showed a strong preference for online tutorials (46.59%), with documentation (25%), blogs and articles (26.14%), and other resources such as ChatGPT and video tutorials (2.27%) also being popular choices. They also demonstrated a willingness to explore new tools when existing ones were less efficient. Participant S23 noted that:

> *"In my current project, I used HTML, CSS, and JS to create front-end web pages, but then I found another tool called Reflex, a development tool for writing front-end code in Python, which made the process much easier."*

This quote highlights the exploratory nature of students' learning journeys, showcasing their

ability to adapt to new tools that better meet project requirements. It reflects the importance of flexibility and resourcefulness in the learning process, as students actively seek out tools that enhance productivity and improve outcomes. This behavior also aligns with the trend of trial-and-error learning, where students experiment with various tools to discover the most suitable ones for their tasks.

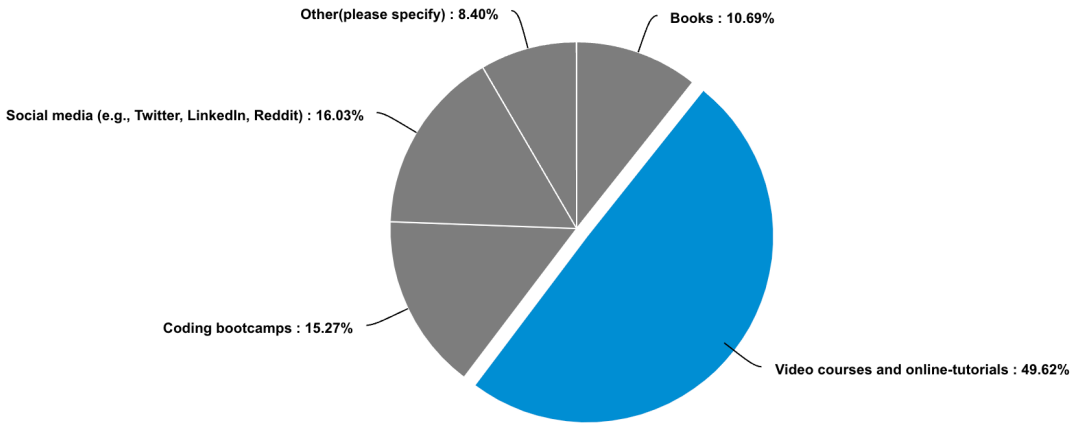


Figure 4.1: Learning Resources for Professionals

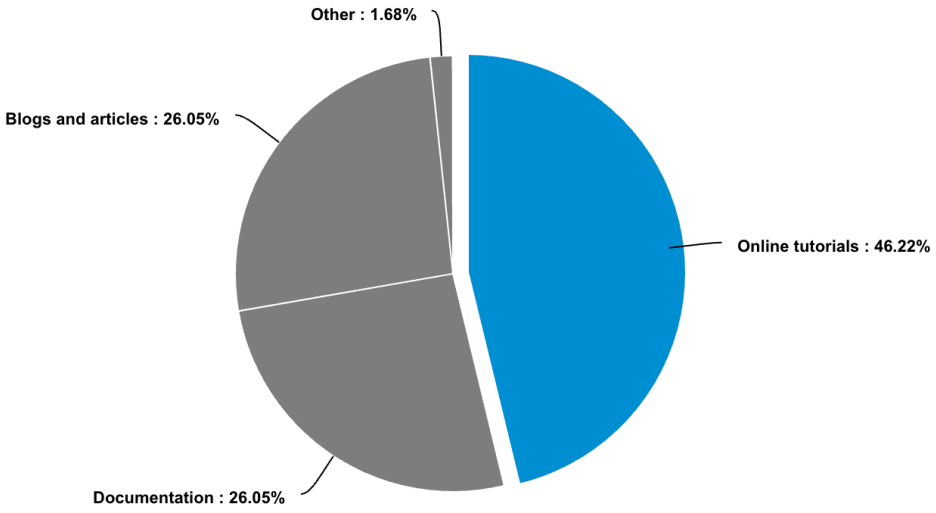


Figure 4.2: Learning Resources for Students

4.3 Time Investment

The time invested in learning new tools varied between the two groups. Professionals typically spent 1-3 hours per week on learning new tools (47.83%), with some dedicating 4-8 hours (24.64%) or more than 8 hours (11.59%). A smaller percentage spent less than 1 hour per week (10.14%), and 5.80% reported not spending any time at all. In contrast, students generally invested more consistent hours, with 59.52% spending 1-2 hours per week, 26.19% dedicating 3-5 hours, 9.52% less than 1 hour, and 4.76% more than 5 hours per week.

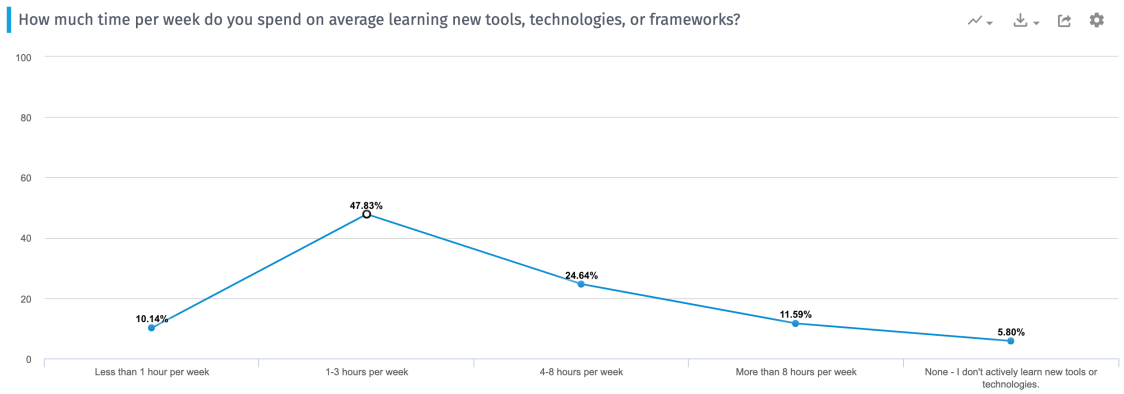


Figure 4.3: Time spent by professionals on learning new tools.

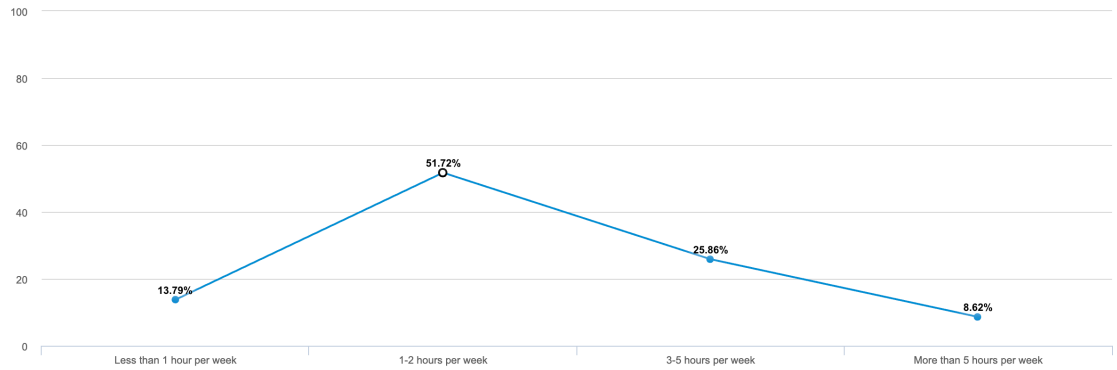


Figure 4.4: Time spent by students on learning new tools.

While students and professionals show slight variations in how their time is distributed across

different learning categories, the overall trends are similar: the majority of both groups dedicate approximately 1–3 hours per week to learning new tools. A smaller proportion in both groups invests either significantly less or significantly more time.

To determine whether these observed differences were statistically significant, a two-sample t-test was conducted. The results indicated no statistically significant difference in the average weekly learning time between students and professionals ($t(106) = -1.03$, $p = 0.303$).Since the p-value is greater than the conventional threshold of 0.05, we cannot reject the null hypothesis, suggesting that the time investments made by both groups are comparable.(Here 106 refers to the degree of freedom).

These results imply that, despite operating in different environments—academic for students and professional for working individuals—the actual time spent learning new tools is broadly similar across both groups. Although students and professionals may differ in the context and motivations behind their learning, the amount of time they dedicate each week does not significantly vary. This finding highlights the shared need for continuous upskilling among both emerging and experienced developers, driven by the constantly evolving landscape of new tools and technologies.

4.4 Challenges and Solutions

Professionals and students both encountered challenges when learning new tools, although the type and frequency of these challenges varied between the two groups. For professionals, the primary challenges included time constraints (14.54%), lack of clear documentation (15.96%), and a steep learning curve (13.12%). As Participant P55 remarked:

"One of the biggest issues is trying to keep up with the learning process while also

> *having time to complete my normal duties at work. It is hard for me to learn new technologies through courses while also being actively engaged in finishing my features for my sprint."*

This quote exemplifies the difficulty professionals face in balancing learning with their regular job duties, reflecting how the demands of ongoing projects can limit their ability to explore new tools. Additionally, they faced issues such as insufficient training resources (11.35%), incompatibility with current systems (9.57%), and uncertainty about the long-term viability of tools (8.87%).

To address these challenges, professionals prioritized learning that aligned with immediate project needs (21.20%), career advancement (15.20%), and peer recommendations (10.40%). They relied heavily on formal training programs and online courses, with 49.62% using video courses and online tutorials as their primary learning resources.

Students reported the complexity of technology (31.58%), lack of time (39.47%), insufficient documentation (10.53%), and limited resources (18.42%) as significant barriers. One student remarked:

> *"Managing coursework alongside learning new tools for internships can feel like juggling too many things at once. Sometimes, even finding the right resources can be challenging."*

This quote highlights the difficulty students face in balancing multiple commitments, emphasizing the importance of time-efficient learning strategies and accessible resources. They tackled these challenges by referring to other online tutorials (25.64%), seeking help from peers or mentors (21.37%), engaging in hands-on practice (22.22%), trial and error (23.08%), and using official documentation (7.69%). These approaches helped them manage the complexities and constraints they faced in their learning processes.

Overall, both groups demonstrated a proactive approach to overcoming their learning challenges by leveraging available resources, prioritizing their learning time, and seeking support from peers and formal education.

4.5 Effectiveness and Preferences of Learning Resources

The surveys underscored the preference for online tutorials and video courses as primary learning resources among both groups. To better understand this preference, we also asked participants what factors make these learning resources effective for mastering new tools. Professionals highlighted functionality and features (25.96%), ease of use and learning curve (23.56%), integration and compatibility (20.67%), reliability and support (17.79%), and cost and value proposition (12.02%) as the most critical aspects in their evaluation. The students' survey results showed similar observations, with 64.29% finding their learning resources effective and 30.95% rating them as very effective. They considered learning style (29.32%), level of experience (24.06%), content quality (23.31%), cost (16.54%), and availability of support (6.77%) when choosing resources.

Participant S27 expressed the usefulness of interactive learning platforms, saying:

> *"I found LinkedIn Learning videos pretty useful as the teachers tend to give you exercise files that allow you to follow along with them."*

Another participant S39 further emphasized the value of detailed guidance, stating:

> *"I personally found online video tutorials to be the most effective because they walk you through the entire process step-by-step, which really helps when you are trying to learn a new framework or tool for the first time."*

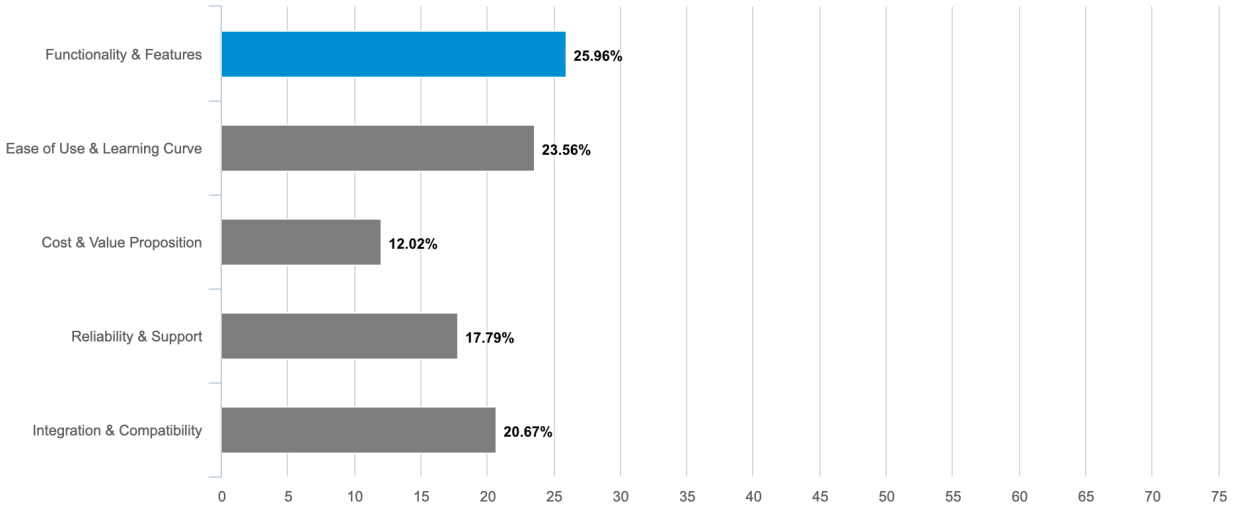


Figure 4.5: factors to consider when choosing a tool (Professionals)

Additionally, the importance of hands-on learning was reiterated by participant S46 who shared:

> *"Hands-on tutorials, especially those featuring live coding or code walk throughs, captivate my attention and enhance my learning experience."*

These quotes emphasize how engaging, hands-on content enhances the learning experience by allowing students to apply new concepts in real time. They also highlight the importance of practical resources that combine theoretical instruction with interactive elements, making learning both efficient and effective.

Professionals emphasized the effectiveness of accessible, well-structured platforms, particularly highlighting YouTube tutorials, Udemy courses, and official documentation. For example, Participant P55 shared: Participant P64:

> *" For learning, official documentation, tutorials on platforms like YouTube, and interactive coding websites like Codecademy and Frontend Masters are highly recommended."*

These findings highlight that both students and professionals prefer learning platforms that offer flexibility, hands-on experiences, and accessibility, ensuring they can progress at their own pace while mastering the tools they need.

4.6 Collaboration and Community Influence

Collaboration with peers and colleagues was highlighted as a significant enhancer of the learning experience for both professionals and students. 31 respondents indicated that working in groups, sharing knowledge, and solving problems collectively were beneficial. As one student noted:

> *"Collaborating with my peers and labmates has made many coding challenges simpler as we were able to collectively combine our knowledge, and our shared knowledge was greater than any one individual. Also, bugs are easier to find with a lot of eyes looking at something."*

This quote exemplifies how collective knowledge and teamwork not only enhance learning outcomes but also streamline problem-solving efforts, reflecting the collaborative nature of software development. Another student emphasized the importance of collaboration:

> *"Collaborating with my classmates made understanding new frameworks so much easier. Sometimes, a quick explanation from a peer saves hours of struggling on my own."*

This illustrates how quick exchanges with peers can alleviate individual challenges and accelerate the learning process. Importantly, professionals also recognized the critical role

of community and collaboration. In the professional survey, 7.09% of respondents identified Lack of Community Support as a key challenge when learning and adopting new tools, highlighting the importance of having access to active peer communities and expert forums. Additionally, when asked how they evaluate the credibility of learning resources, 11.5% of professionals cited Recommendations from Peers as a factor influencing their choice. Online communities and forums like Stack Overflow and GitHub played a crucial role in problem-solving, providing access to tutorials, extensions and plugins, peer learning, and recommendations. These findings reinforce that across both students and professionals, collaboration, peer input, and active community participation significantly influence the learning journey. Access to a knowledgeable community not only improves the quality of learning but also provides ongoing support for solving technical challenges.

Chapter 5

Discussion

The results of this study provide valuable insights into the learning behaviors, preferences, and challenges faced by full-stack developers. By examining the survey data collected from both professionals and students, we can better understand the factors influencing the adoption of essential tools, the time investment required for learning, and the effectiveness of various learning resources. This discussion will interpret the findings in relation to the research questions, highlight the contributions to the field of software engineering education, and explore the implications for curriculum development. Furthermore, it will describe the design and development of LearnMate, a web-based application created to address the gaps identified through the research and to support more effective and personalized learning experiences for both students and professionals.

5.1 Interpretation of Findings

5.1.1 RQ1: Essential Tools and Influences

The survey results identified several essential tools commonly used by both professionals and students in full-stack development. Among professionals, React, Node.js, Flask, Django, AWS services, PostgreSQL, and MySQL were the most frequently used tools. These tools are popular due to their functionality, performance, and widespread industry adoption. For

example, React is favored for its flexibility and efficiency in building dynamic user interfaces, while Node.js is valued for its ability to handle asynchronous operations and high scalability. AWS services are indispensable for their robust cloud infrastructure and diverse service offerings, which are crucial for deploying and managing applications.

Students reported frequent use of VS Code, React, Django, Spring Boot, MySQL, MongoDB, and Node.js. VS Code is particularly popular among students for its lightweight, yet powerful features, and extensive extensions that support a variety of programming languages and frameworks. Django and Spring Boot are appreciated for their comprehensive documentation and ease of use in setting up backend services.

The choice of these tools is influenced by a combination of factors including functionality, ease of use, integration and compatibility with existing systems, reliability, support, and cost. For instance, the integration capabilities of a tool determine how well it can work with other tools and systems within a developer's workflow. Ease of use and a low learning curve are crucial for quick adoption, especially in fast-paced development environments.

Additionally, peer recommendations, industry trends, and specific project requirements significantly influenced developers' decisions on which tools to learn. Peer recommendations often provide practical insights and proven efficacy of tools in real-world scenarios. Industry trends guide developers towards tools that are in demand, ensuring their skills remain relevant. Specific project requirements dictate the need for particular tools that best fit the project's technical needs and constraints. These findings align with previous research by Murphy-Hill et al. (2015) [5], which highlights the importance of social dynamics and peer interactions in the discovery and adoption of new tools.

5.1.2 RQ2: Time Investment

The analysis of survey responses reveals that students and professionals invest a comparable amount of time each week in learning new software tools. While there are slight differences in how their time is distributed across time categories, the overall pattern shows that both groups most commonly dedicate between one to five hours per week to learning. A two-sample t-test confirmed that the difference in average time investment between students and professionals is not statistically significant ($t(106)=-1.03, p=0.303$), indicating that both groups, despite differences in context and motivation, tend to allocate similar amounts of time toward upskilling.

For students, learning time is often driven by academic structures. Course assignments, lab work, and project deadlines frequently require the adoption of new tools, motivating students to allocate time within their weekly academic schedules to acquire these skills. Additionally, the university environment may provide access to peer support, guided resources, and instructor-led learning, all of which can influence how students organize and spend their learning time. However, some students also indicated the need to explore tools independently, especially when academic instruction lacked coverage of industry-relevant technologies, thereby increasing their self-directed learning time outside the classroom.

Professionals, on the other hand, often engage in continuous learning as a means of staying current with evolving industry standards. While some learning may be formally integrated into job roles especially in organizations that support ongoing training or experimentation with new technologies much of the learning happens independently. Professionals frequently rely on self-paced resources, online tutorials, documentation, and peer communities to learn during personal or downtime. Unlike students, whose learning is often assignment driven, professionals may tailor their learning goals based on job demands, project needs, or career

development objectives.

Several key factors influence the amount of time both groups dedicate to learning. These include the complexity of the tool or technology, the quality and accessibility of learning resources, and the degree of urgency or relevance to their current goals. For instance, tools with steep learning curves or poor documentation may require a greater time investment, whereas highly rated tutorials or community-driven platforms can accelerate the learning process. Personal factors such as motivation, learning style, and time management skills also play a critical role.

These findings reinforce the idea that although students and professionals operate in distinct environments—academic versus workplace—their time investment in learning is shaped by a mix of institutional, personal, and contextual factors. The shared challenge of keeping up with rapidly evolving technologies underscores the need for time-efficient, flexible, and high-quality learning resources that can adapt to a range of learning styles and time constraints. Educational institutions and training providers can better support both groups by offering modular, self-paced learning paths, structured resource recommendations, and opportunities for peer interaction and mentorship.

5.1.3 RQ3: Effective Learning Resources and Challenges

Both professionals and students reported a strong preference for online tutorials and video courses as their primary learning resources. These resources are considered highly effective due to their accessibility, structured content, and practical, hands-on approach. However, students particularly valued the step-by-step guidance and visual demonstrations that support foundational understanding, while professionals emphasized the flexibility to skip to relevant sections and apply concepts directly to real-world tasks. Online tutorials and video

courses allow learners to progress at their own pace, revisit complex sections, and directly apply what they learn through interactive coding exercises and projects.

Other valuable resources include documentation, blogs, articles, and community forums like Stack Overflow and GitHub. Official documentation provides authoritative information and comprehensive coverage of a tool's features. Blogs and articles often offer insights, tips, and best practices from experienced developers. Community forums facilitate problem-solving through peer support and shared knowledge.

The main challenges faced by developers when learning new tools include the complexity of technologies, lack of clear documentation, time constraints, and limited access to high-quality learning resources. Complex technologies with steep learning curves can be daunting and require significant effort to understand and use effectively. Inadequate or poorly organized documentation can hinder learning by leaving gaps in understanding or providing unclear instructions.

Time constraints, whether due to work deadlines or academic schedules, limit the amount of time developers can dedicate to learning new tools. Limited access to high-quality learning resources, such as up-to-date tutorials or expert guidance, can further impede learning progress. To overcome these challenges, developers often rely on trial and error, hands-on practice, peer support, and seeking help from mentors or online communities. These strategies help bridge gaps in knowledge, provide practical experience, and offer support and encouragement. These findings underscore the need for comprehensive, high-quality, and easily accessible learning resources tailored to the diverse needs of developers.

5.2 Contributions to the Field

The findings of this study contribute significantly to the field of software engineering education by offering in-depth insights into the learning behaviors, preferences, and challenges faced by full-stack developers. By identifying the most commonly used tools, such as React, Node.js, Flask, Django, AWS services, PostgreSQL, MySQL, VS Code, Spring Boot, and MongoDB, and understanding the factors influencing their adoption, this research provides valuable data that can be used to develop targeted curricula and learning resources. These tools are critical in the industry, and by ensuring that educational programs focus on these technologies, institutions can better prepare students for the demands of the job market.

Furthermore, the study delves into the factors that influence developers' choices regarding which tools to learn. These factors include functionality, ease of use, integration capabilities, reliability, support, cost, peer recommendations, industry trends, and project-specific requirements. By understanding these factors, curriculum developers can design educational programs that not only cover the technical aspects of these tools but also address the practical considerations that are important to developers. For example, incorporating case studies, real-world examples, and project-based learning can make the educational content more relevant and engaging, thereby enhancing the learning experience.

The research also highlights the importance of understanding the learning behaviors and preferences of full-stack developers. By analyzing how much time developers invest in learning new tools and the types of resources they find most effective, the study provides insights that can help optimize educational programs. For instance, professionals typically spend 1-3 hours per week on learning new tools, while students may invest more consistent hours. Educational programs can be tailored to accommodate these differences, offering flexible learning paths that cater to the time constraints of professionals and more intensive modules for stu-

dents. In addition to technical skills, the study underscores the significance of social learning and peer support in the learning process. Social dynamics and peer interactions play a crucial role in how developers discover and adopt new tools. Educational programs that incorporate collaborative and community-based learning opportunities can leverage these social aspects to enhance learning outcomes. Group projects, peer reviews, and online forums can facilitate knowledge sharing and provide a support network for learners, making the learning process more interactive and supportive.

The study also addresses the challenges that developers face when learning new tools, such as the complexity of technologies, lack of clear documentation, time constraints, and limited access to high-quality resources. By identifying these challenges, the research provides a foundation for developing targeted interventions that can help learners overcome these obstacles. For instance, creating detailed, user-friendly documentation, offering time management strategies, and providing access to expert mentors can significantly improve the learning experience. Moreover, the insights gained from this study have broader implications for continuous professional development in the software engineering field. As technology evolves rapidly, there is a need for lifelong learning to keep up with new tools and practices. Training programs that emphasize self-directed learning, provide access to the latest resources, and encourage continuous skill development can help professionals maintain their competencies and advance their careers.

5.3 Curriculum Development and Impact on Education

The findings from this study have significant implications for curriculum development and the broader field of software engineering education. By identifying the most common tools and understanding the factors influencing their adoption, educational institutions can design curricula more aligned with industry demands. This alignment ensures that graduates are better prepared to enter the workforce and meet the expectations of employers.

Survey responses strongly supported the need for curricular changes. Out of 71 participants, 47 respondents (66.2%) expressed that the current curriculum needs to be updated to better reflect industry trends and tool usage. Eleven participants (15.5%) reported neutral opinions, suggesting that while they did not see immediate issues, improvements could still be beneficial. The remaining participants felt that the existing curriculum was effective, often emphasizing that components like capstone projects provided valuable real-world preparation. These responses highlight a clear majority sentiment that greater alignment with evolving industry practices is necessary to enhance graduate readiness.

As one participant P49 remarked:

> *"One challenge I faced was that the courses didn't provide enough exposure to tools that are actively used in industry. I had to spend extra time outside my classes learning relevant tools through tutorials and online resources."*

This quote highlights the gap between academic curricula and industry requirements, emphasizing the need for more relevant and up-to-date instruction. It reflects the burden placed on students to independently learn critical tools outside their coursework.

One of the key takeaways from this study is the importance of practical, hands-on learning experiences. The preference for online tutorials, video courses, and interactive coding exercises among both professionals and students underscores the need for curricula that prioritize practical application over theoretical instruction. Incorporating **project-based learning**, where students can work on real-world projects using industry-standard tools, can enhance their understanding and proficiency [20].

Student participants offered practical advice for others starting projects or internships, with participant S35 stating:

> *"Start early and focus on one tool at a time. It can get overwhelming if you try to learn everything at once. Find tutorials that align with your learning style, and don't hesitate to ask for help from peers or mentors—it makes a huge difference."*

This advice reinforces the need for educational programs to offer structured, manageable learning paths supported by accessible resources and opportunities for peer collaboration. In light of these findings and the previously discussed gaps, addressing curriculum shortcomings through targeted updates can better equip students with industry-relevant skills and reduce their reliance on supplementary self-learning.

Specific recommendations include integrating courses focused on widely used frameworks and technologies such as React.js, Node.js, Docker, and Kubernetes; offering **project-based learning** opportunities where students build applications using modern development tools; and embedding design principles and software architecture fundamentals into core coursework. Additionally, updating capstone projects to reflect real-world team collaboration practices and providing electives on cloud platforms like AWS or Azure can further align academic preparation with current industry demands.

Furthermore, as another participant P57 noted:

> *"Based on my industry experiences and undergraduate curriculum, I think I did not receive enough instruction or exposure to good design principles. Professionally, I found that often any projects I was assigned to or inherited suffered from poor design decisions made early in the process."*

This quote emphasizes the gap between academic instruction and industry requirements, highlighting the need for better integration of core design principles into the curriculum. Addressing these gaps will ensure that graduates can contribute meaningfully to professional projects from the start, reducing the impact of poor design decisions.

Moreover, the study highlights the role of **peer recommendations and social learning** in the adoption of new tools. Educational programs can leverage this by fostering a collaborative learning environment. Group projects, peer reviews, and participation in online communities can facilitate knowledge sharing and provide a support network for students [4, 11]. Such collaborative approaches not only enhance learning outcomes but also mirror the collaborative nature of the software development industry.

The identified challenges, such as the complexity of technologies and insufficient documentation, point to the need for curricula that include **comprehensive and clear instructional materials**. Providing detailed documentation, step-by-step guides, and access to expert mentors can help students navigate complex technologies and reduce the learning curve. Additionally, offering workshops and seminars on time management and effective learning strategies can help students balance their academic responsibilities with the need to learn new tools.

The findings also suggest the need for **flexible curriculum structures** that can accommodate different learning styles and schedules, particularly within academic programs. Specifically, curriculum improvements should include:

- **Offering modular courses** that allow students to focus on specific technologies or topics (e.g., front-end development, cloud computing, DevOps) aligned with their career interests.

- **Embedding regular project-based assignments** throughout the curriculum to promote continuous hands-on learning, rather than concentrating practical experiences solely at the capstone stage.

- **Incorporating multiple learning formats**, such as video tutorials, interactive coding exercises, and technical documentation, to support diverse learning preferences and enhance engagement.

- **Providing flexible elective options** so that students can specialize in industry-relevant technologies (e.g., electives on React.js, Kubernetes, cloud deployment), tailoring their education to specific career paths.

- **Linking academic projects to real-world industry problems**, encouraging students to apply industry-standard tools and workflows consistently throughout their academic journey.

In addition, curriculum developers should establish **regular review processes** to monitor emerging industry trends and update course content accordingly. Maintaining alignment with current technologies, frameworks, and practices ensures that students graduate with skills that are both relevant and competitive in the evolving job market.

These improvements collectively aim to build practical, industry-aligned competencies throughout students' educational experiences, better preparing them for professional success upon graduation.

In conclusion, by integrating these insights into curriculum development, educational institutions can create more effective and relevant programs that prepare students for the dynamic field of software engineering. This approach not only enhances the employability of graduates but also contributes to the continuous professional development of current industry professionals.

5.4 LearnMate: From Research to Application

Building upon the insights gathered through this study, the **LearnMate** application was developed as a web-based tool designed to bridge the gap between software engineering education and real-world industry requirements. The design of **LearnMate** was directly shaped by the patterns identified through survey responses, especially regarding learning preferences, tool adoption challenges, and the effectiveness of various educational resources.

Key features include:

- **Personalized Learning Paths**: Users can select their experience level and time commitment (e.g., 1 week, 1 month, 3 months), and LearnMate generates a tailored roadmap with recommended tools and frameworks.
- **Progress Tracking and Scheduling**: The platform provides interactive task lists and visual progress trackers to help users stay on track and accountable.
- **Integrated Mini-Projects**: For every tool or concept, LearnMate suggests mini-projects that reinforce hands-on learning and real-world application.
- **Resource Quality Filtering**: Based on community feedback and ongoing review, only high-quality tutorials, documentation, and videos are recommended to users.

- **LearnMate AI (AI Assistant)**: An intelligent assistant embedded within the platform, LearnMate AI helps users clarify doubts, suggests relevant resources in real time, and assists with basic code debugging based on user queries.

LearnMate offers curated and customizable learning paths for full-stack development, helping users—both students and professionals—navigate their learning journeys more efficiently. It emphasizes structured progression and resource quality while adapting to individual goals, and timelines.

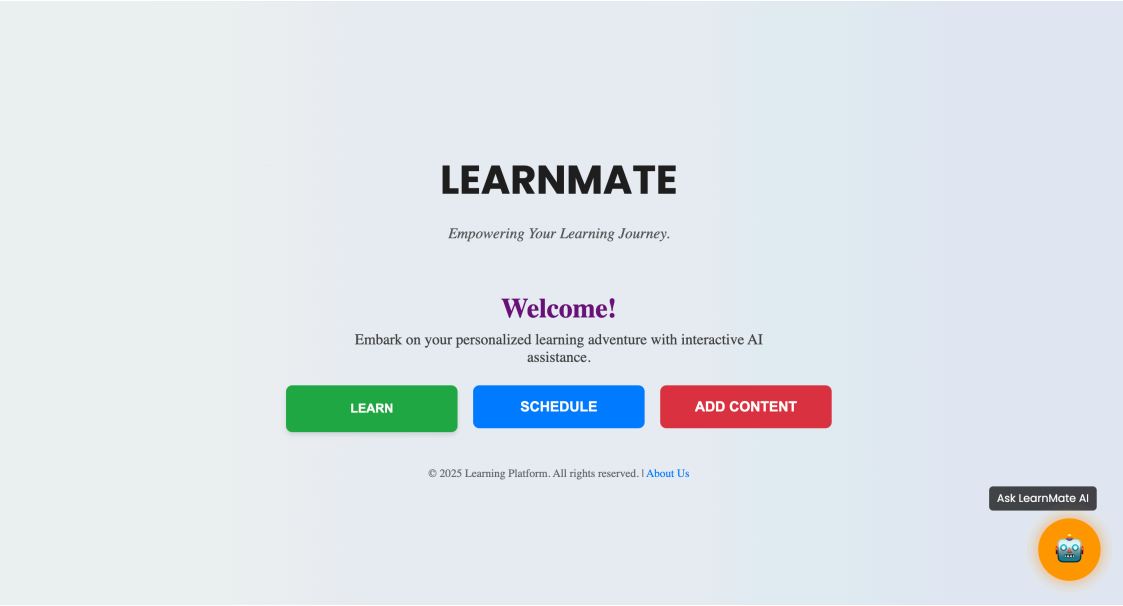


Figure 5.1: Homepage of the LearnMate application.

Upon clicking the **Learn** option on the homepage, users are directed to a topic-based learning interface.

Figure 5.2: Topic selection screen after choosing ”Learn”.

Users can choose a development area such as Frontend, Backend, Full Stack etc. This allows them to dive directly into learning a specific toolset or domain based on their goals.

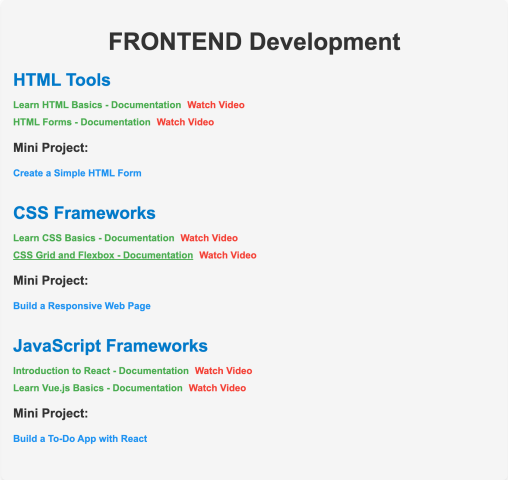


Figure 5.3: Learning interface for Frontend development.

Each topic (e.g., HTML, CSS, JavaScript) provides hand-picked links to high-quality documentation and top-rated video tutorials available online (Free resources). After completing each section, learners are presented with a related mini project to apply their knowledge and

reinforce hands-on learning.

By clicking on the **Schedule** option on the homepage, users can generate a learning plan tailored to their timeline and goals.

Figure 5.4: Schedule Your Learning Path - initial screen.

The scheduling interface prompts users to choose a time period (e.g., 1 week, 1 month, 3 months) and a learning focus area such as Frontend, Backend,Data Science etc.

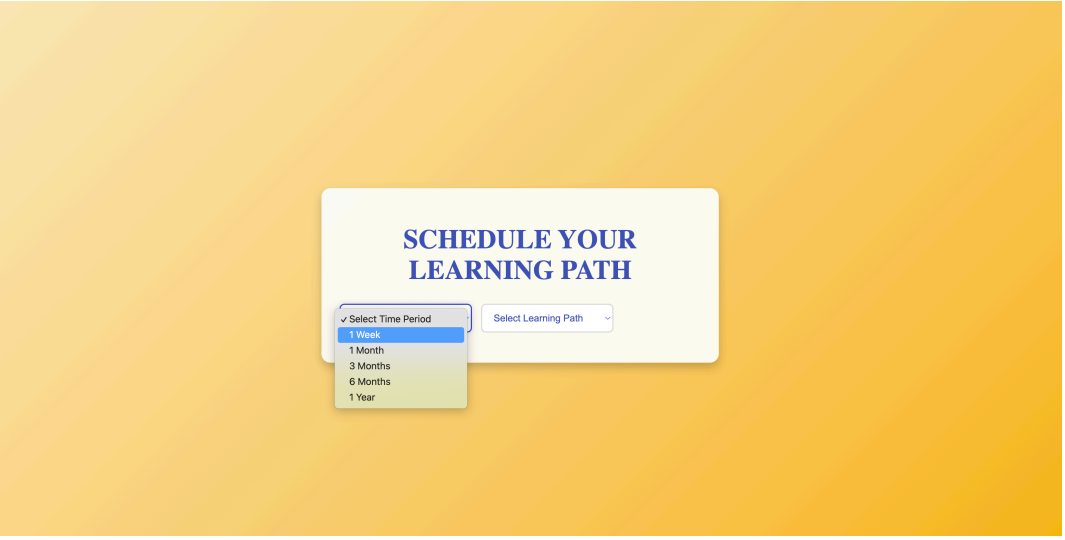


Figure 5.5: Dropdown for selecting a time period.

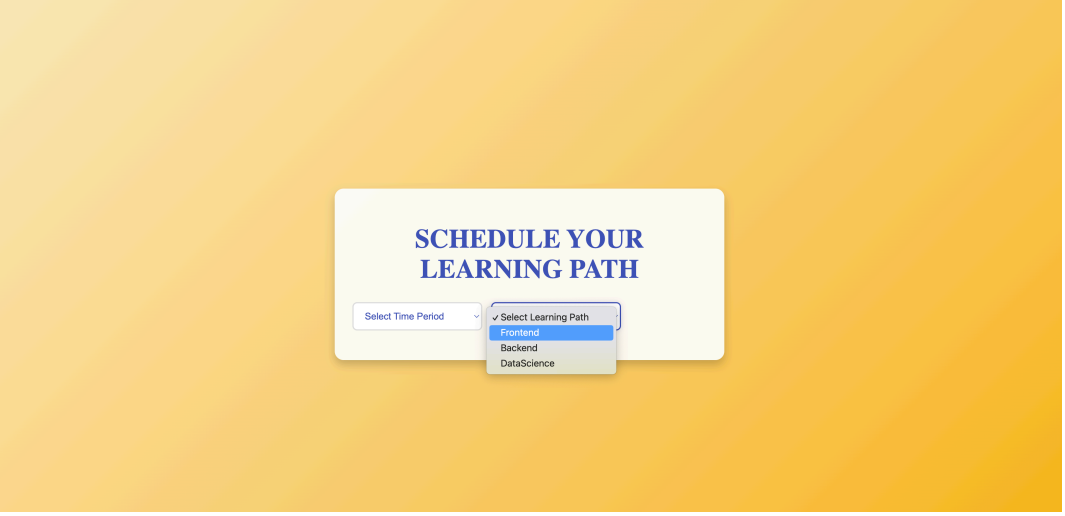


Figure 5.6: Dropdown for selecting a learning path.

Once both filters are selected, LearnMate dynamically generates a curated set of video and documentation-based resources with an estimated duration for each. Users can track their progress directly on the page.

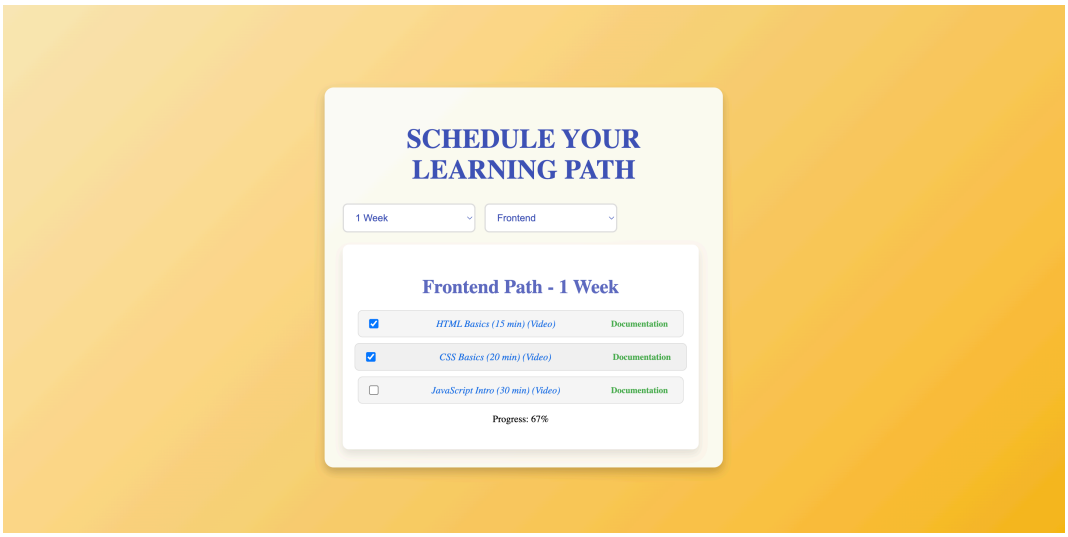


Figure 5.7: Example learning path with progress tracking for 1-week frontend plan.

By clicking on the **Add Content** option, LearnMate enables users to contribute useful learning materials to the platform by submitting their own recommended resources.

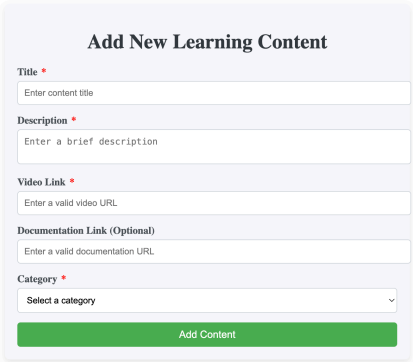


Figure 5.8: Form interface for submitting new learning content.

As shown above, users can add content by filling in a simple form that includes a title, brief description, a video URL, optional documentation link, and a category. This feature empowers learners to share helpful tutorials or documentation they come across during their learning journey, thereby enhancing community-driven resource building within the platform.

By clicking the **Ask LearnMate AI** button, users can receive AI-powered assistance to clarify doubts, understand concepts, or get learning suggestions in real time.

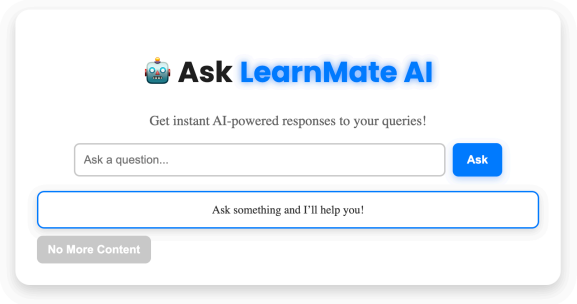


Figure 5.9: LearnMate AI input interface.

The AI assistant allows users to enter a question in natural language. The interface is designed to be simple and accessible to both beginners and experienced users.

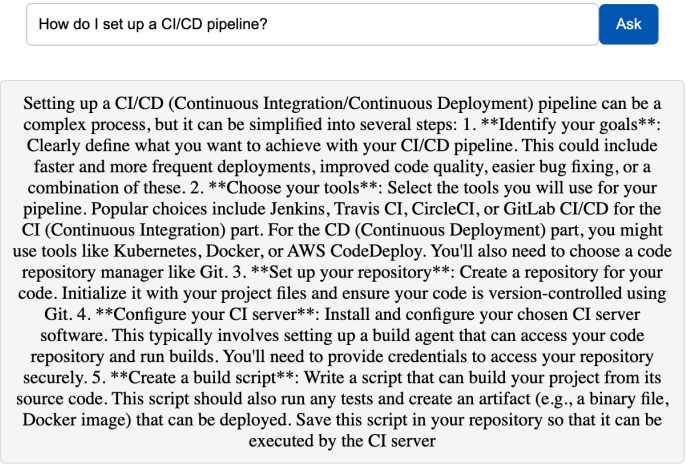


Figure 5.10: AI-generated response example for a CI/CD pipeline query.

When a user submits a question, LearnMate AI provides a detailed, context-aware expla-

nation. This can include step-by-step guidance, tool suggestions, or curated resource links depending on the query. The feature enhances the self-paced learning experience by reducing blockers and encouraging curiosity-driven exploration.

However, a current limitation of the AI assistant is that the responses are not always well-formatted or consistently structured. This can make longer answers—such as technical walkthroughs—less readable and may require users to interpret or reformat the information themselves. Improving the output formatting is a key area for future enhancement to increase usability and learning clarity.

Application Development Process

The LearnMate application was developed as a practical extension of the research findings and followed an iterative, user-centered design process. The development was carried out over multiple phases:

- **Design and Planning**: Based on the survey insights, initial user flows were designed to reflect the common needs and challenges faced by learners. Special attention was paid to supporting self-paced learning.
- **Frontend Development**: The frontend was built using **React.js**, chosen for its modularity and ability to create a responsive, interactive user experience. Tailwind CSS was used for lightweight, customizable styling. Components such as timelines, checklists, and resource cards were built to guide learners intuitively through their learning paths.
- **Backend and Data Storage**: The backend was powered by **Node.js** with an Express.js server, while learning paths, user progress, and feedback data were stored in

MongoDB, allowing for flexible document-based storage. RESTful APIs were designed to serve personalized content dynamically.

- **AI Integration**: The **LearnMate AI** feature was developed using the **StarCoder large language model**, an open-source code-focused model from Hugging Face and ServiceNow. StarCoder was integrated into the application to provide intelligent, context-aware responses to users' technical queries. Its strength in understanding programming languages and development-related queries made it especially suitable for LearnMate's use case. Prompt templates were carefully designed to generate helpful suggestions, debug guidance, and curated resource recommendations aligned with each learner's progress.

  While the AI assistant proved useful in providing real-time guidance, an identified limitation was the inconsistent formatting of its output. Some responses—particularly those involving step-by-step instructions—lacked clear structure, which may hinder readability. Improving the formatting and presentation of AI-generated content is a key area for future development, ensuring that explanations are more user-friendly and easier to follow.

- **Testing and Feedback**: To ensure usability and relevance, the application was tested with a small group of computer science students and recent graduates. Feedback was collected to refine the user interface, improve response relevance from the AI assistant, and ensure that the learning paths were logically structured and achievable.

- **Scalability Considerations**: As part of future planning, the architecture was designed with modularity in mind, allowing for future enhancements such as course progress syncing, user accounts, collaborative learning features, and analytics dashboards for educators.

Through LearnMate, this study transitions from pure research to real-world implementation, offering a tangible tool that embodies the principles of learning-driven software engineering. It serves as a living prototype that demonstrates how research-backed learning models can directly inform the development of intelligent, scalable, and learner-friendly educational technology.

5.4.1 LearnMate User Study: Evaluating Usability and Learning Impact

To further assess the practical impact of the LearnMate application, a structured user study was conducted with ten students from Virginia Tech and the University of Virginia. All participants were sophomores, with academic backgrounds in computer science, neuroscience, psychology, economics, and environmental sciences. This interdisciplinary sample was selected to evaluate the platform's usability, learning effectiveness, and overall learner experience across five dimensions: navigation, need for guidance, perceived learning gains, feature usefulness, and improvement suggestions. The feedback was collected through follow-up questions after participants explored curated learning paths and other key features across various software development domains.

Methodology

Participants were invited to use the LearnMate application in a self-paced exploratory session lasting approximately 8–10 minutes. During this time, they were encouraged to interact with the key features of the platform, including:

- Selecting a learning path using the "Learn" or "Schedule Your Learning Path" options.

- Browsing curated learning resources such as video tutorials and documentation.
- Viewing and understanding the structure of mini projects associated with learning topics.
- Exploring additional features like the Progress Tracker, LearnMate AI assistant, and Add Content section.

After completing their exploration, participants were asked a set of follow-up questions aimed at evaluating their experience. These questions focused on five areas: ease of navigation, need for guidance, perceived learning gains, feature usefulness, and suggestions for improvement. The responses were collected qualitatively to gain deeper insight into how users interacted with the platform and where improvements could be made.

The following sections present the findings from the follow-up discussions with participants. Their responses are grouped according to five key themes that align with the study's evaluation dimensions: ease of navigation, need for guidance, perceived learning gains, feature usefulness, and suggestions for improvement. Each theme summarizes user feedback and highlights patterns that emerged during the post-exploration conversations.

Ease of Navigation and User Experience

Most participants described the interface as intuitive and beginner-friendly. They appreciated the clean layout and clearly labeled features. One participant commented, *"Very easy, everything was clearly labeled."* Dropdown menus and sectioned content further enhanced the user experience. However, some users noted initial confusion regarding specific features, such as the "Add Content" button, suggesting the addition of a brief onboarding tutorial or tooltip hints to support new users.

Guidance and Feature Clarity

Seven out of ten participants required no additional help to navigate the platform. The flow was described as natural and well-structured. For users who did experience confusion, particularly around advanced features like the AI chatbot or content submission sections, lightweight support in the form of walkthroughs or contextual tooltips was recommended.

Learning Effectiveness

All ten participants expressed that a platform like LearnMate enhanced their understanding of technical concepts more effectively than traditional learning methods. The integrated format—combining curated videos, official documentation, and hands-on mini-projects—was particularly well-received, as it allowed users to learn by doing while also having structured guidance. Participant P1 shared, *"Definitely. It's better than just watching YouTube tutorials."* This sentiment was echoed by Participant P10, who noted, *"Yes, I'd recommend it to anyone starting out in dev."*

These responses suggest that the combination of practical exercises with high-quality learning materials contributes to greater conceptual clarity and retention, especially for beginners seeking a structured, applied learning experience.

Most Valuable Features

Participants highlighted the following LearnMate features as the most useful:

- **Schedule Your Learning Path**: Personalizing timelines (e.g., 1 week, 1 month) helped students plan effectively.

- **Mini-Projects**: These offered a glimpse into how concepts could be applied in practice.
- **Side-by-Side Content**: Access to paired videos and documentation enhanced both visual and analytical learning.
- **Progress Tracker**: Helped users stay accountable and monitor completion.

As participant P3 described, *"Being able to choose a timeframe and topic made it really easy to fit learning into my schedule."* Others emphasized how the visual trackers motivated consistent progress.

Suggestions for Improvement

Participants offered thoughtful recommendations to enhance LearnMate:

- Introduce **quizzes or checkpoints** after lessons for reinforcement.
- Improve **AI chatbot responsiveness** and expand its query-handling scope.
- Add a **calendar view** for weekly learning plans.
- Allow users to **submit their own projects** and receive peer or mentor feedback.
- Enable **customizable learning preferences**, such as favoriting lessons or bookmarking.

These suggestions reflect a desire for increased interactivity, adaptive learning tools, and collaborative features, which can be addressed in future iterations of the tool.

Recommendation and Overall Value

Nine out of ten participants stated they would recommend LearnMate to their peers, especially beginner developers. The reasons included its structured approach, ease of use, and effectiveness in delivering hands-on learning.

Summary

The LearnMate user study reinforced the platform's value as an effective learning companion for learners across a range of technical domains. The positive feedback aligns with earlier survey findings, validating the platform's design goals and learner-centered philosophy. At the same time, participant suggestions offer a clear roadmap for future improvements, including enhanced interactivity and better-formatted AI responses. Overall, LearnMate demonstrates strong potential as a scalable, adaptable, and practical educational technology that supports personalized, self-directed learning in modern computing education and beyond.

Chapter 6

Limitations

Despite the valuable insights provided by this study, several limitations should be acknowledged. Addressing these limitations can help frame the findings within their proper context and guide future research efforts.

One of this study's main limitations is the relatively small sample size, particularly for the student survey. With 71 professional participants and 47 student respondents, the findings may not be fully generalizable to the broader population of full-stack developers. Future research should aim to include a larger and more diverse sample to enhance the generalizability of the results. The cross-sectional design of the survey captures a snapshot of the participants' experiences at a single point in time. This approach does not account for changes in learning behaviors, preferences, and challenges over time. Longitudinal studies that follow participants over an extended period could provide deeper insights into how learning processes and tool adoption evolve.

The survey was distributed primarily among full-stack developers working in various companies and Virginia Tech graduate and undergraduate students. This geographical focus may limit the applicability of the findings to other regions or educational institutions. Expanding the survey to include participants from different geographical areas and diverse educational backgrounds would enhance the comprehensiveness of the study. While the survey included open-ended questions to gather qualitative insights, the depth of these responses was limited. Conducting in-depth interviews or focus groups could provide richer, more nuanced data on

the learning experiences and challenges of full-stack developers. This qualitative approach would complement the quantitative findings and offer a more holistic understanding of the subject matter.

The technology landscape in software engineering evolves rapidly, and new tools and frameworks frequently emerge. The findings of this study may become outdated as new technologies gain prominence. Continuous research is necessary to keep pace with these changes and ensure that educational programs remain relevant and effective. By acknowledging these limitations, this study provides a clear context for interpreting the findings and highlights areas for future research.

Addressing these limitations in subsequent studies will help build a more robust understanding of the learning behaviors, preferences, and challenges faced by developers and contribute to the ongoing improvement of software engineering education.

Chapter 7

Future Work

Building on the insights and limitations of this study, several avenues for future research can be explored to further enhance our understanding of learning-driven software engineering.

Future research should aim to include a larger and more diverse sample of developers. By expanding the participant pool and including individuals from different geographical regions, educational backgrounds, and professional experiences, findings can be made more generalizable and reflective of the broader population of software developers.

Conducting in-depth qualitative research, such as interviews and focus groups, can provide richer, more nuanced data on the learning experiences and challenges developers face. Exploring personal narratives and detailed accounts of learning journeys can uncover insights not easily captured through surveys or short-term studies.

Future work should also investigate the learning behaviors and preferences of developers specializing in different domains, such as frontend, backend, mobile development, data science etc. Comparative studies across these roles can reveal distinct challenges and highlight the unique learning strategies required in each specialization. Additionally, evaluating the impact of learning resources such as formal education, coding bootcamps, mentorship programs, and peer-driven platforms can provide deeper insight into their long-term effectiveness. This knowledge can inform the design of targeted, high impact educational interventions.

In addition, future work should focus on extending and evaluating the LearnMate

platform itself. While initial feedback was positive, several areas for enhancement were identified during the user study. These include improving the formatting of AI-generated responses, integrating quiz-based checkpoints, enabling social learning features (such as peer feedback or project sharing), and supporting user accounts to enable personalized progress tracking. A long-term study measuring learning gains and retention among LearnMate users compared to those using traditional resources would provide valuable insight into its practical effectiveness. Moreover, integrating analytics dashboards for educators could help monitor learner engagement and tailor instructional content in real time.

Finally, future research should assess the long term outcomes of different learning approaches. Evaluating the impact of educational experiences on career advancement, job performance, and ongoing professional development will offer feedback for continuous improvement of software engineering education at large.

By pursuing these research directions, future studies can build on the foundation established by this work and contribute to the ongoing evolution of software engineering education. This continued exploration will help shape more effective, personalized, and adaptive learning pathways to meet the evolving needs of developers in today's fast paced technology landscape.

Chapter 8

Conclusions

This study examined how students and professionals learn new software development tools, with a particular focus on their behaviors, preferences, and challenges. Through survey data and a user study, it identified commonly used tools, the factors influencing tool adoption, and the types of learning resources developers rely on—such as video tutorials, official documentation, and peer support. The findings underscore the need for accessible, hands-on, and time-efficient learning resources, and they offer guidance for educators and curriculum designers seeking to align instructional content with real-world practices.

The development and evaluation of the LearnMate platform further illustrated how a personalized, interactive tool can support learners in navigating diverse technologies. Participants found the tool intuitive and helpful, though they also identified opportunities for improvement, such as better AI response formatting and more interactive learning features. Overall, this research contributes to a deeper understanding of how developers acquire technical skills and offers practical strategies to enhance education and training through adaptive, learner-centered tools.

Bibliography

[1] Emerging research in cloud distributed computing systems advances in systems analysis software engineering and high performance computing. 2019. URL https://api.semanticscholar.org/CorpusID:246472405.

[2] Andrew Begel and Beth Simon. Novice software developers, all over again. New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582160. doi: 10.1145/1404520.1404522. URL https://doi.org/10.1145/1404520.1404522.

[3] Edwin F. Boza, Xavier Andrade, Jorge Bermúdez Cedeño, Jorge R. Murillo, Harold Aragon, and Cristina L. Abad. Cloud computing based (serverless computing) using serverless architecture for dynamic web hosting and cost optimization. *2023 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6, 2023. URL https://api.semanticscholar.org/CorpusID:258870191.

[4] Jane Brindley, Walti Christine, and Lisa Marie Blaschke. Creating effective collaborative learning groups in an online environment. *International Review of Research in Open and Distance Learning*, 10, 06 2009. doi: 10.19173/irrodl.v10i3.675.

[5] Lan Cheng, Emerson Rex Murphy-Hill, Mark Canning, Ciera Nicole Christopher Jaspan, Collin Green, Andrea Marie Knight Dolan, Nan Zhang, and Elizabeth Kammer. What improves developer productivity at google? code quality. In *Foundations of Software Engineering: Industry Paper*, 2022.

[6] Felix Dobslaw, Kristian Angelin, Lena-Maria Öberg, and Awais Ahmad. The gap between higher education and the software industry – a case study on technology differences, 2023. URL https://arxiv.org/abs/2303.15597.

[7] Vahid Garousi, Gorkem Giray, Eray Tuzun, Cagatay Catal, and Michael Felderer. Closing the gap between software engineering education and industrial needs. *IEEE Software*, 37(2):68–77, 2020. doi: 10.1109/MS.2018.2880823.

[8] Ashokkumar Gurusamy and Ikram Ahamed Mohamed. The evolution of full stack development: Trends and technologies shaping the future. *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)*, 1(1):100–108, 2020.

[9] Vəhid Gəruslu, Görkem Giray, Eray Tüzün, Cagatay Catal, and Michael Felderer. Closing the gap between software engineering education and industrial needs. *IEEE Software*, 37:68 – 77, 03 2020. doi: 10.1109/MS.2018.2880823.

[10] Rajesh Kulkarni and Palacholla Padmanabham. Integration of artificial intelligence activities in software development processes and measuring effectiveness of integration. *IET Software*, 11, 09 2016. doi: 10.1049/iet-sen.2016.0095.

[11] Marjan Laal and Seyed Mohammad Ghodsi. Benefits of collaborative learning. *Procedia - Social and Behavioral Sciences*, 31:486–490, 2012. ISSN 1877-0428. doi: https://doi.org/10.1016/j.sbspro.2011.12.091. URL https://www.sciencedirect.com/science/article/pii/S1877042811030205. World Conference on Learning, Teaching Administration - 2011.

[12] Emerson Murphy-Hill, Da Young Lee, Gail C. Murphy, and Joanna McGrenere. How do users discover new tools in software development and beyond? *Computer Supported Cooperative Work (CSCW)*, 24(5):389–422, 2015. ISSN 1573-7551. doi: 10.1007/s10606-015-9230-9. URL https://doi.org/10.1007/s10606-015-9230-9.

[13] Veena N. Learning preferences among students. *IOSR Journal Of Humanities And Social Science*, 15:26–32, 01 2013. doi: 10.9790/0837-1562632.

[14] Damla Oguz and Kaya Oguz. Perspectives on the gap between the software industry and the software engineering education. *IEEE Access*, 7:117527–117543, 2019. doi: 10.1109/ACCESS.2019.2936660.

[15] Martin P. Robillard and Robert Deline. A field study of api learning obstacles. 16(6), 2011. ISSN 1382-3256. doi: 10.1007/s10664-010-9150-8. URL https://doi.org/10.1007/s10664-010-9150-8.

[16] Lucas Romao, Marcos Kalinowski, Clarissa Barbosa, Allysson Allex Araújo, Simone D. J. Barbosa, and Helio Lopes. Agile minds, innovative solutions, and industry-academia collaboration: Lean rd meets problem-based learning in software engineering education, 2024. URL https://arxiv.org/abs/2407.15982.

[17] Varun Shah. Towards efficient software engineering in the era of ai and ml: Best practices and challenges. 07 2019.

[18] Antero Taivalsaari, Tommi Mikkonen, Cesare Pautasso, and Kari Systä. Full stack is not what it used to be. In *International Conference on Web Engineering*, 2021. URL https://api.semanticscholar.org/CorpusID:234366295.

[19] Antero Taivalsaari, Tommi Mikkonen, Cesare Pautasso, and Kari Systä. Full stack is not what it used to be. In Marco Brambilla, Richard Chbeir, Flavius Frasincar, and Ioana Manolescu, editors, *Web Engineering*, pages 363–371, Cham, 2021. Springer International Publishing. ISBN 978-3-030-74296-6.

[20] John W Thomas. A review of research on project-based learning. 2000.

[21] Adam Trendowicz and Jürgen Münch. Chapter 6 factors influencing software development productivity—state-of-the-art and industrial experiences. volume 77 of *Advances in Computers*, pages 185–241. Elsevier, 2009. doi: https://doi.org/10.1016/

S0065-2458(09)01206-6. URL https://www.sciencedirect.com/science/article/pii/S0065245809012066.

[22] Kelley Walters and Joel Rodriguez. The importance of training and development in employee performance and evaluation. 04 2017.

[23] Iyad Zayour and Hassan Hajjdiab. How much integrated development environments (ides) improve productivity? *J. Softw.*, 8:2425–2431, 2013. URL https://api.semanticscholar.org/CorpusID:6630040.

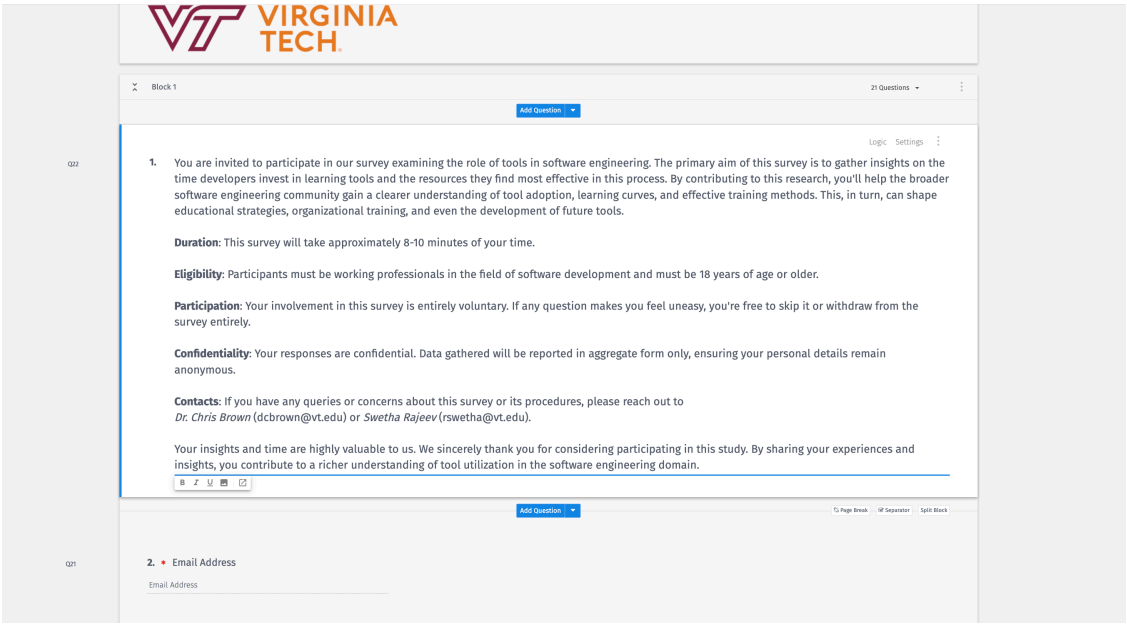
Appendices

Appendix A

Surveys

A.1 Survey for professionals

The following pages show the full set of survey questions presented to professional developers through the QuestionPro platform.



VIRGINIA TECH

Block 1 — 21 Questions

Add Question

Logic Settings

Q22

1. You are invited to participate in our survey examining the role of tools in software engineering. The primary aim of this survey is to gather insights on the time developers invest in learning tools and the resources they find most effective in this process. By contributing to this research, you'll help the broader software engineering community gain a clearer understanding of tool adoption, learning curves, and effective training methods. This, in turn, can shape educational strategies, organizational training, and even the development of future tools.

Duration: This survey will take approximately 8-10 minutes of your time.

Eligibility: Participants must be working professionals in the field of software development and must be 18 years of age or older.

Participation: Your involvement in this survey is entirely voluntary. If any question makes you feel uneasy, you're free to skip it or withdraw from the survey entirely.

Confidentiality: Your responses are confidential. Data gathered will be reported in aggregate form only, ensuring your personal details remain anonymous.

Contacts: If you have any queries or concerns about this survey or its procedures, please reach out to *Dr. Chris Brown* (dcbrown@vt.edu) or *Swetha Rajeev* (rswetha@vt.edu).

Your insights and time are highly valuable to us. We sincerely thank you for considering participating in this study. By sharing your experiences and insights, you contribute to a richer understanding of tool utilization in the software engineering domain.

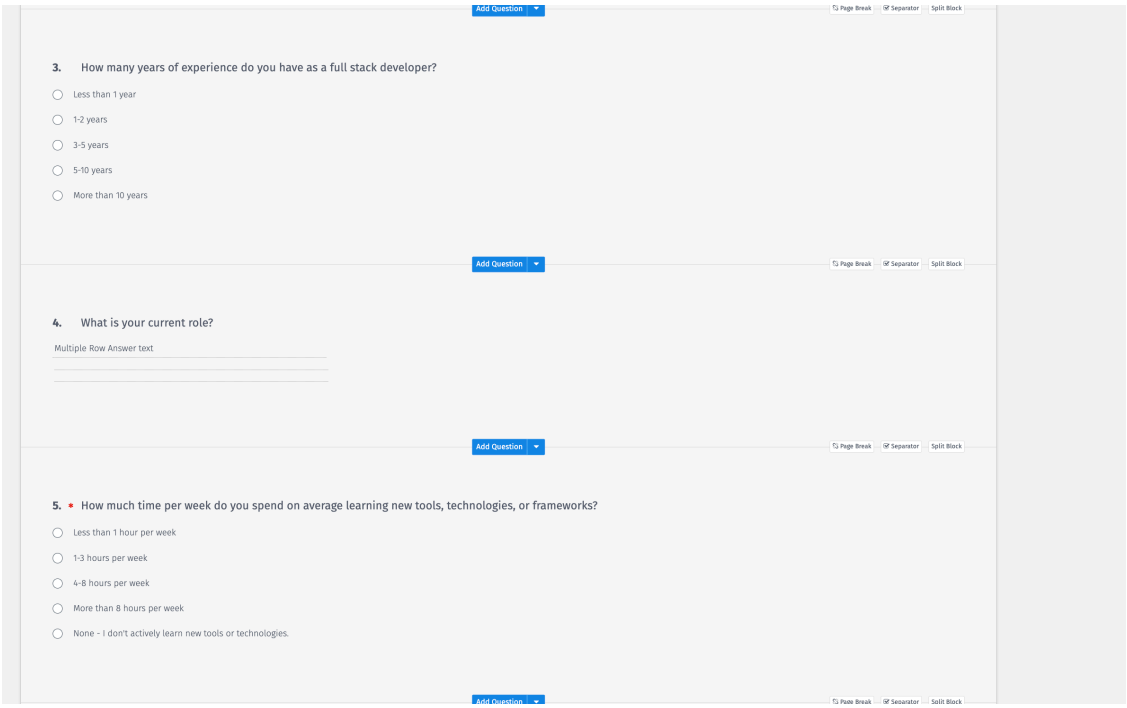
Add Question — Page Break — IF Separator — Split Block

Q21

2. * Email Address

Email Address

Figure A.1: Professional Survey – Page 1



Add Question | Page Break | Separator | Split Block

3. How many years of experience do you have as a full stack developer?

- Less than 1 year
- 1-2 years
- 3-5 years
- 5-10 years
- More than 10 years

Add Question | Page Break | Separator | Split Block

4. What is your current role?

Multiple Row Answer text

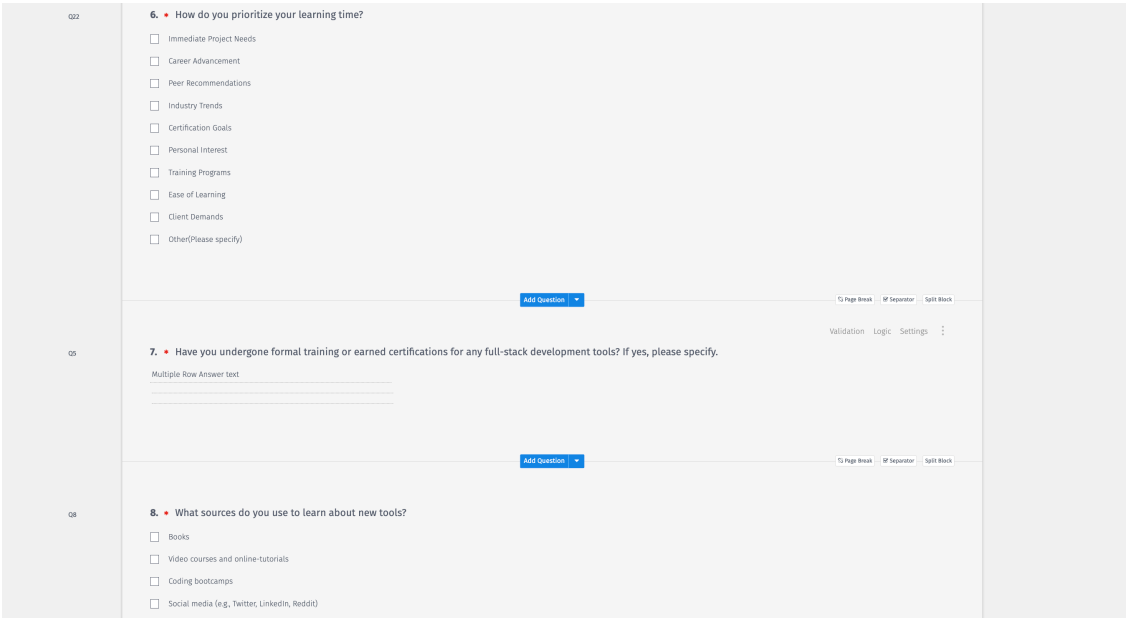
Add Question | Page Break | Separator | Split Block

5. • How much time per week do you spend on average learning new tools, technologies, or frameworks?

- Less than 1 hour per week
- 1-3 hours per week
- 4-8 hours per week
- More than 8 hours per week
- None - I don't actively learn new tools or technologies.

Add Question | Page Break | Separator | Split Block

Figure A.2: Professional Survey – Page 2



Q22

6. • How do you prioritize your learning time?

- Immediate Project Needs
- Career Advancement
- Peer Recommendations
- Industry Trends
- Certification Goals
- Personal Interest
- Training Programs
- Ease of Learning
- Client Demands
- Other(Please specify)

Add Question | Page Break | Separator | Split Block

Validation Logic Settings

Q5

7. • Have you undergone formal training or earned certifications for any full-stack development tools? If yes, please specify.

Multiple Row Answer text

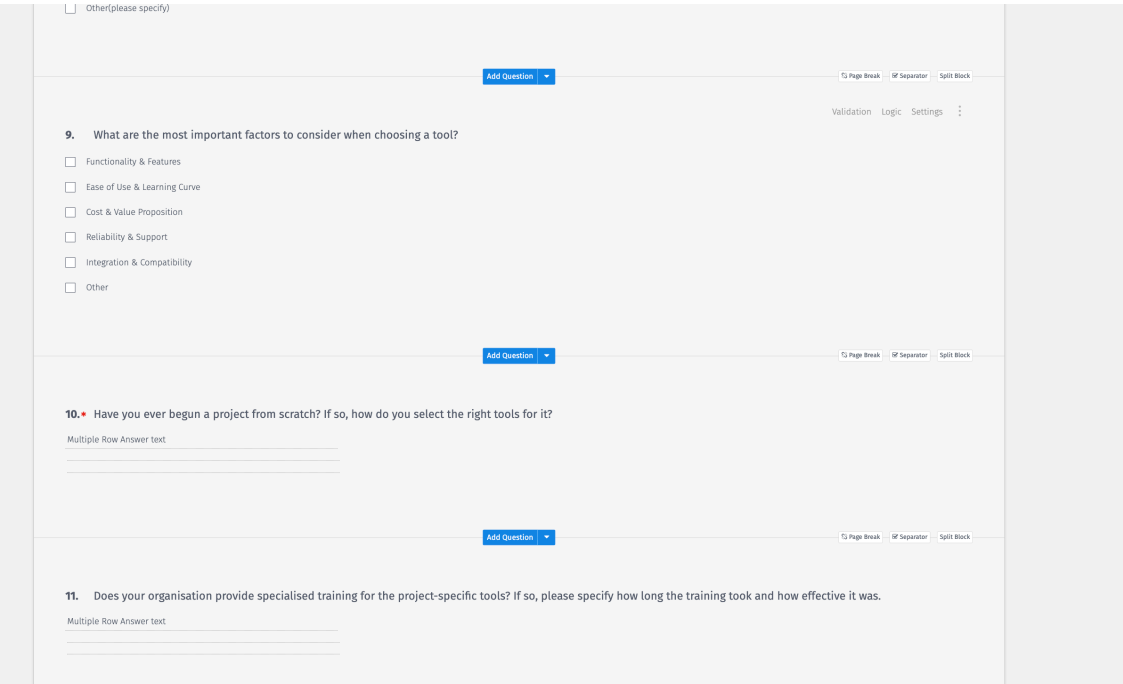
Add Question | Page Break | Separator | Split Block

Q8

8. • What sources do you use to learn about new tools?

- Books
- Video courses and online-tutorials
- Coding bootcamps
- Social media (e.g., Twitter, LinkedIn, Reddit)

Figure A.3: Professional Survey – Page 3



Other(please specify)

Add Question | Page Break | Separator | Split Block

Validation Logic Settings

9. What are the most important factors to consider when choosing a tool?

- Functionality & Features
- Ease of Use & Learning Curve
- Cost & Value Proposition
- Reliability & Support
- Integration & Compatibility
- Other

Add Question | Page Break | Separator | Split Block

10.* Have you ever begun a project from scratch? If so, how do you select the right tools for it?

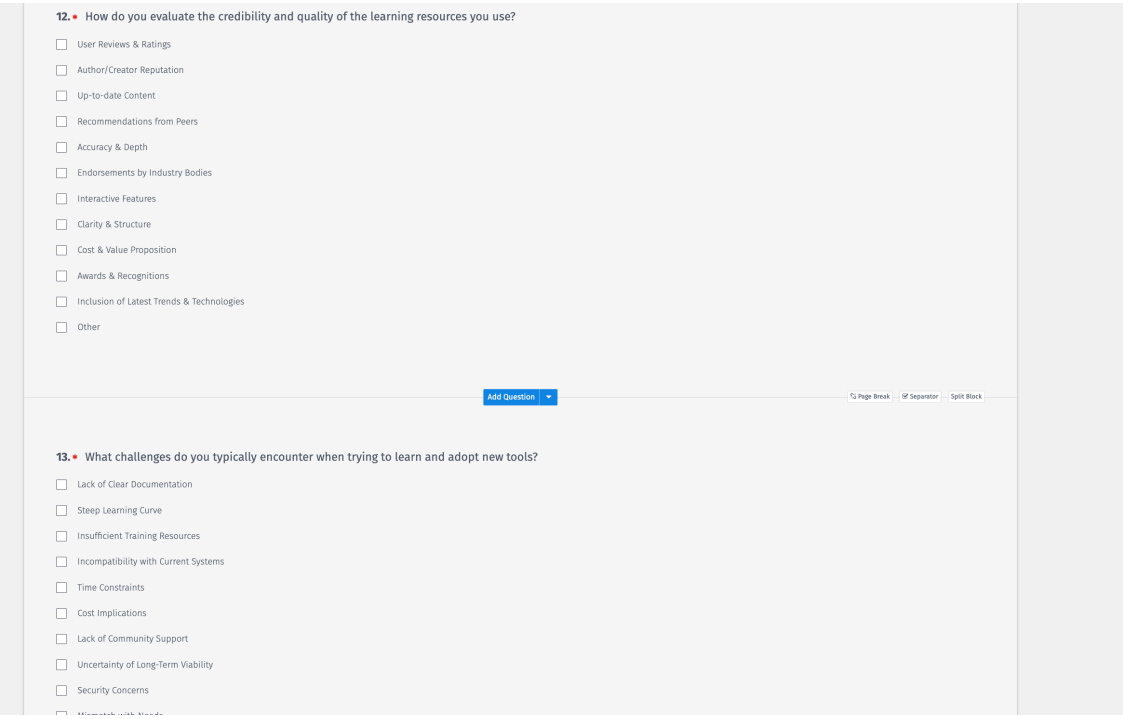
Multiple Row Answer text

Add Question | Page Break | Separator | Split Block

11. Does your organisation provide specialised training for the project-specific tools? If so, please specify how long the training took and how effective it was.

Multiple Row Answer text

Figure A.4: Professional Survey – Page 4



12.* How do you evaluate the credibility and quality of the learning resources you use?

- User Reviews & Ratings
- Author/Creator Reputation
- Up-to-date Content
- Recommendations from Peers
- Accuracy & Depth
- Endorsements by Industry Bodies
- Interactive Features
- Clarity & Structure
- Cost & Value Proposition
- Awards & Recognitions
- Inclusion of Latest Trends & Technologies
- Other

Add Question | Page Break | Separator | Split Block

13.* What challenges do you typically encounter when trying to learn and adopt new tools?

- Lack of Clear Documentation
- Steep Learning Curve
- Insufficient Training Resources
- Incompatibility with Current Systems
- Time Constraints
- Cost Implications
- Lack of Community Support
- Uncertainty of Long-Term Viability
- Security Concerns

Figure A.5: Professional Survey – Page 5

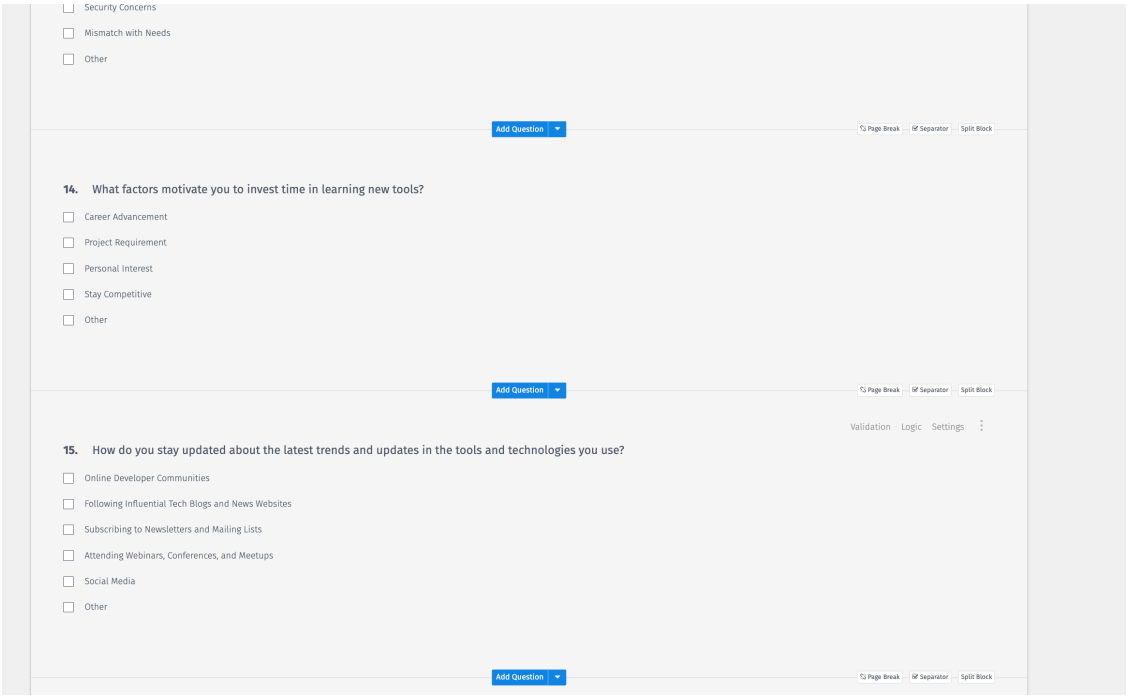


Figure A.6: Professional Survey – Page 6

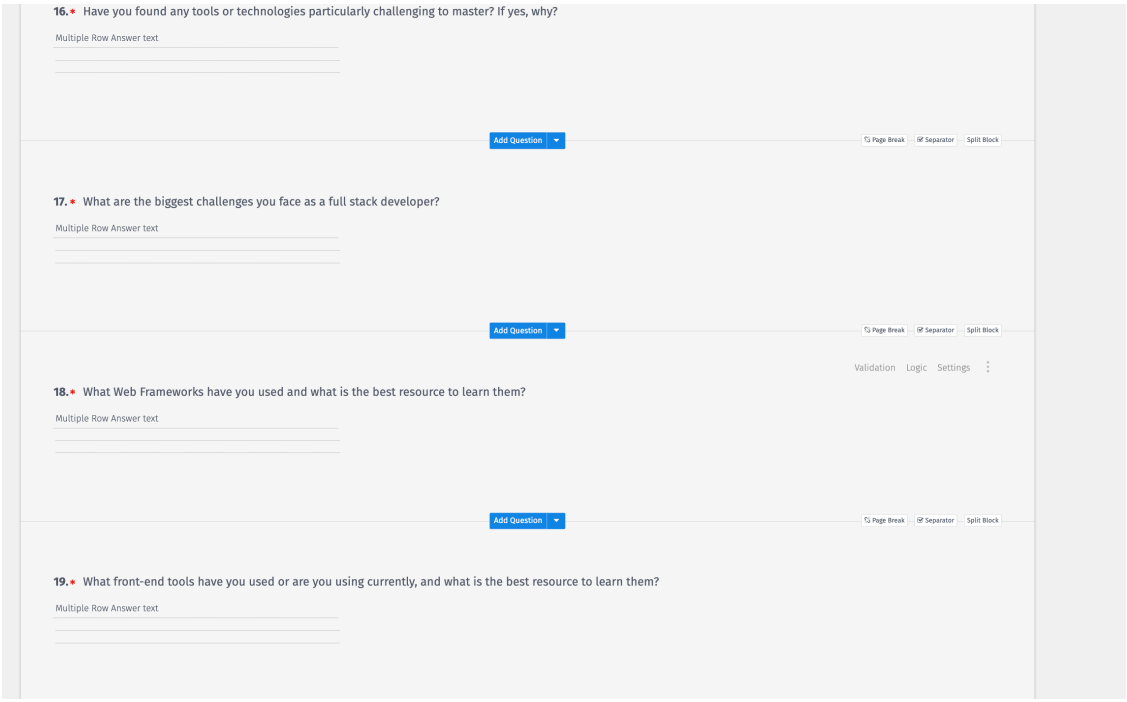


Figure A.7: Professional Survey – Page 7

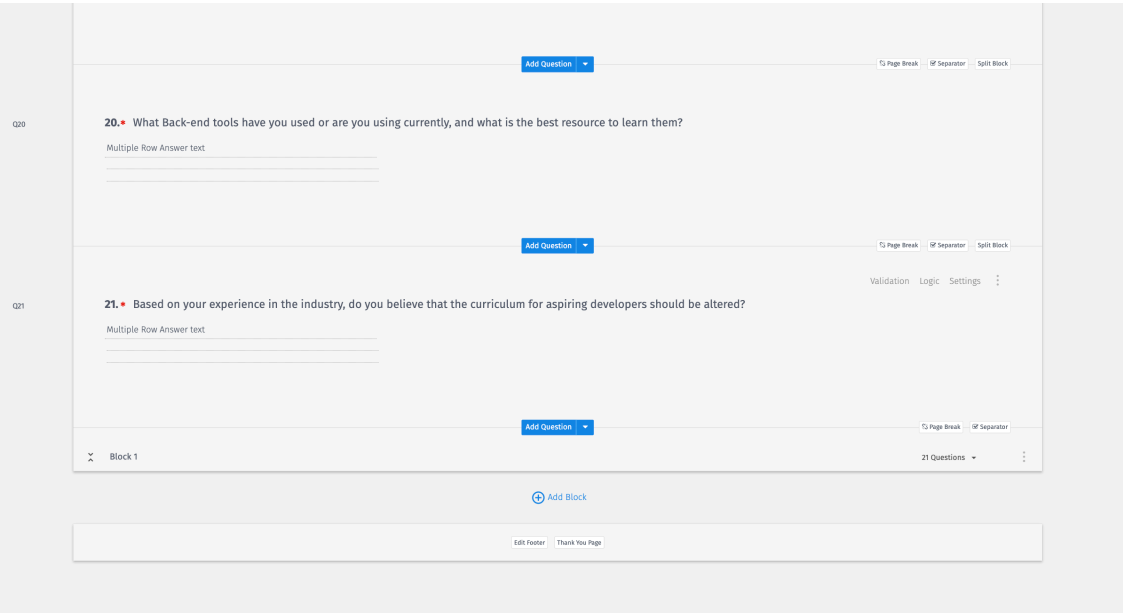
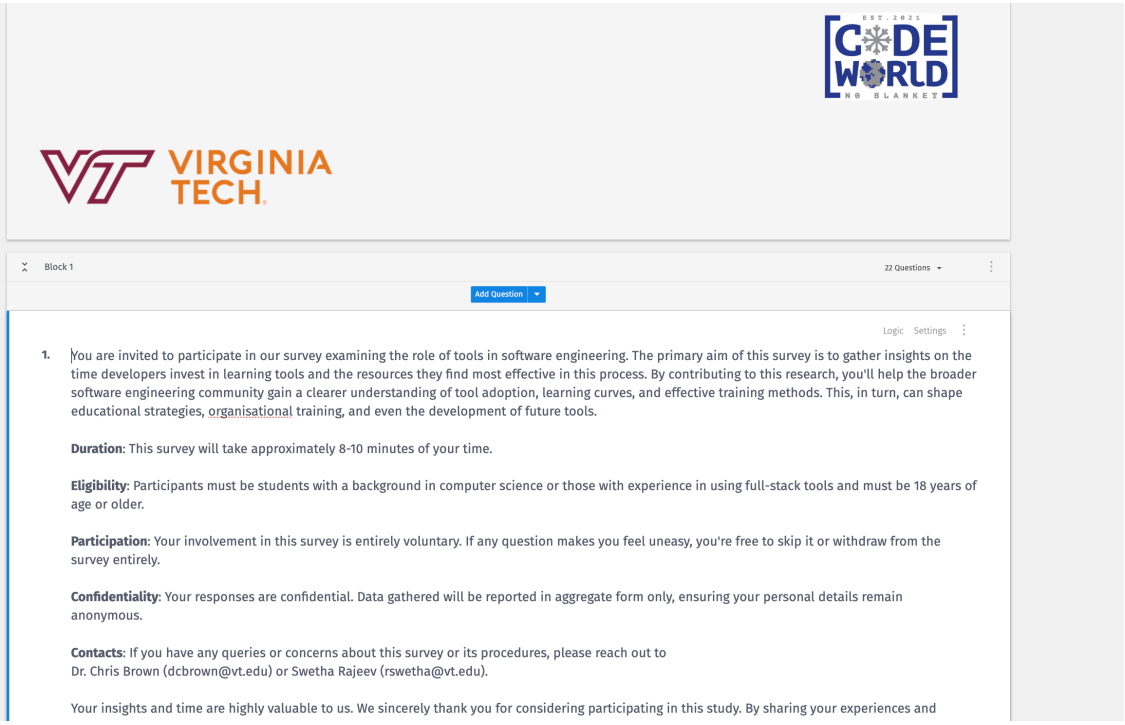


Figure A.8: Professional Survey – Page 8

A.2 Survey for students

The following pages contain the full set of survey questions presented to students. These screenshots were taken directly from the QuestionPro platform.



Block 1 22 Questions

Add Question

Logic Settings

1. You are invited to participate in our survey examining the role of tools in software engineering. The primary aim of this survey is to gather insights on the time developers invest in learning tools and the resources they find most effective in this process. By contributing to this research, you'll help the broader software engineering community gain a clearer understanding of tool adoption, learning curves, and effective training methods. This, in turn, can shape educational strategies, organisational training, and even the development of future tools.

Duration: This survey will take approximately 8-10 minutes of your time.

Eligibility: Participants must be students with a background in computer science or those with experience in using full-stack tools and must be 18 years of age or older.

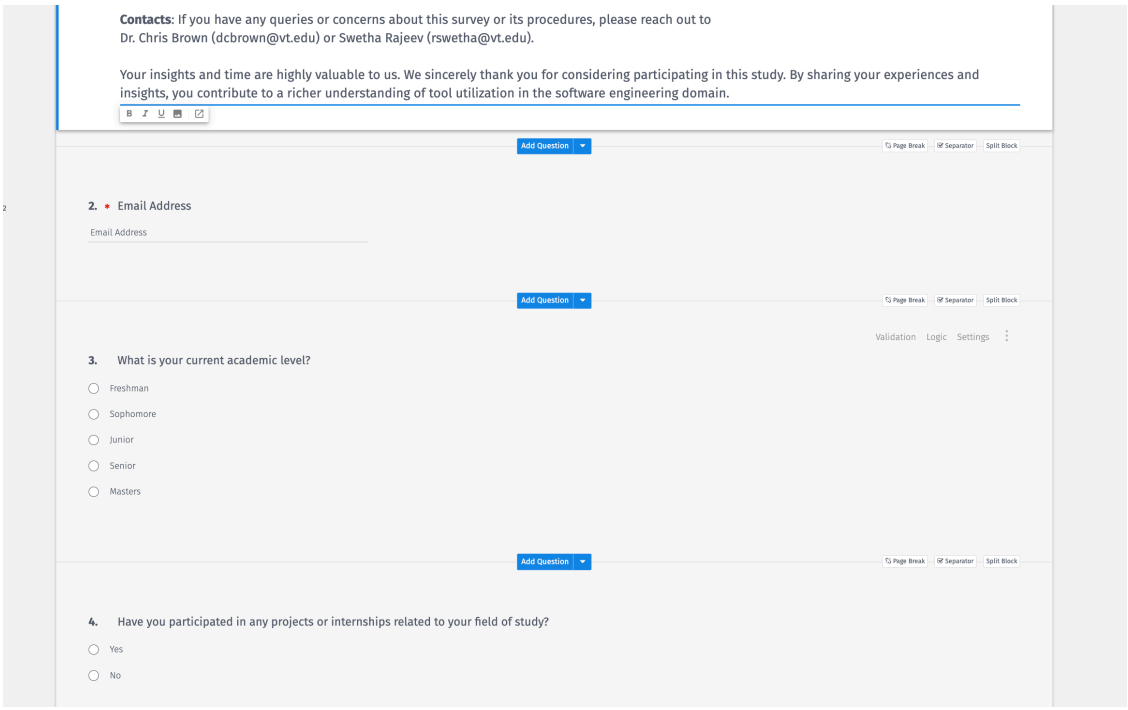
Participation: Your involvement in this survey is entirely voluntary. If any question makes you feel uneasy, you're free to skip it or withdraw from the survey entirely.

Confidentiality: Your responses are confidential. Data gathered will be reported in aggregate form only, ensuring your personal details remain anonymous.

Contacts: If you have any queries or concerns about this survey or its procedures, please reach out to
Dr. Chris Brown (dcbrown@vt.edu) or Swetha Rajeev (rswetha@vt.edu).

Your insights and time are highly valuable to us. We sincerely thank you for considering participating in this study. By sharing your experiences and

Figure A.9: Student Survey – Page 1



Contacts: If you have any queries or concerns about this survey or its procedures, please reach out to
Dr. Chris Brown (dcbrown@vt.edu) or Swetha Rajeev (rswetha@vt.edu).

Your insights and time are highly valuable to us. We sincerely thank you for considering participating in this study. By sharing your experiences and insights, you contribute to a richer understanding of tool utilization in the software engineering domain.

Add Question Page Break Separator Split Block

2. * Email Address

Email Address

Add Question Page Break Separator Split Block

Validation Logic Settings

3. What is your current academic level?

- Freshman
- Sophomore
- Junior
- Senior
- Masters

Add Question Page Break Separator Split Block

4. Have you participated in any projects or internships related to your field of study?

- Yes
- No

Figure A.10: Student Survey – Page 2

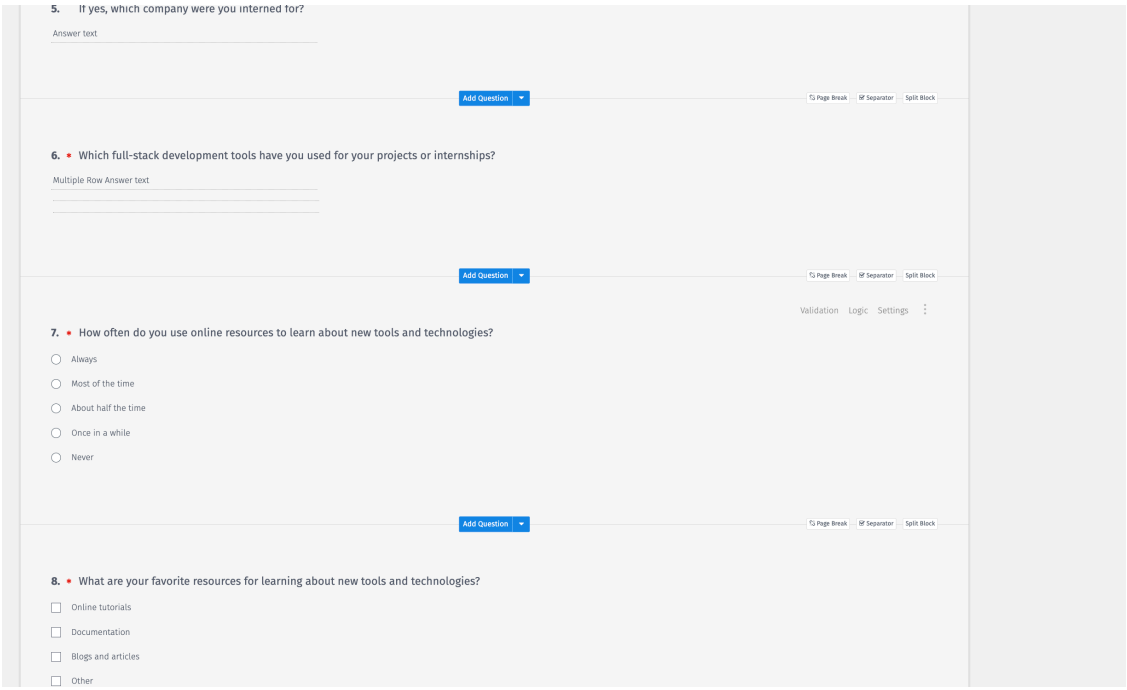


Figure A.11: Student Survey – Page 3

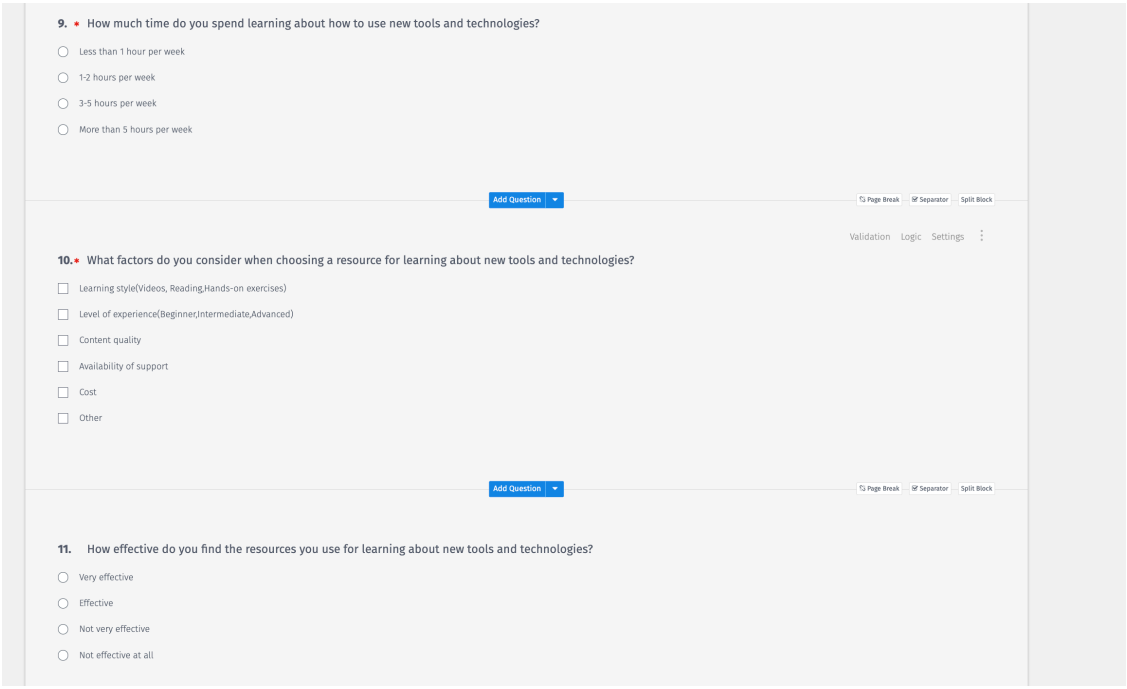
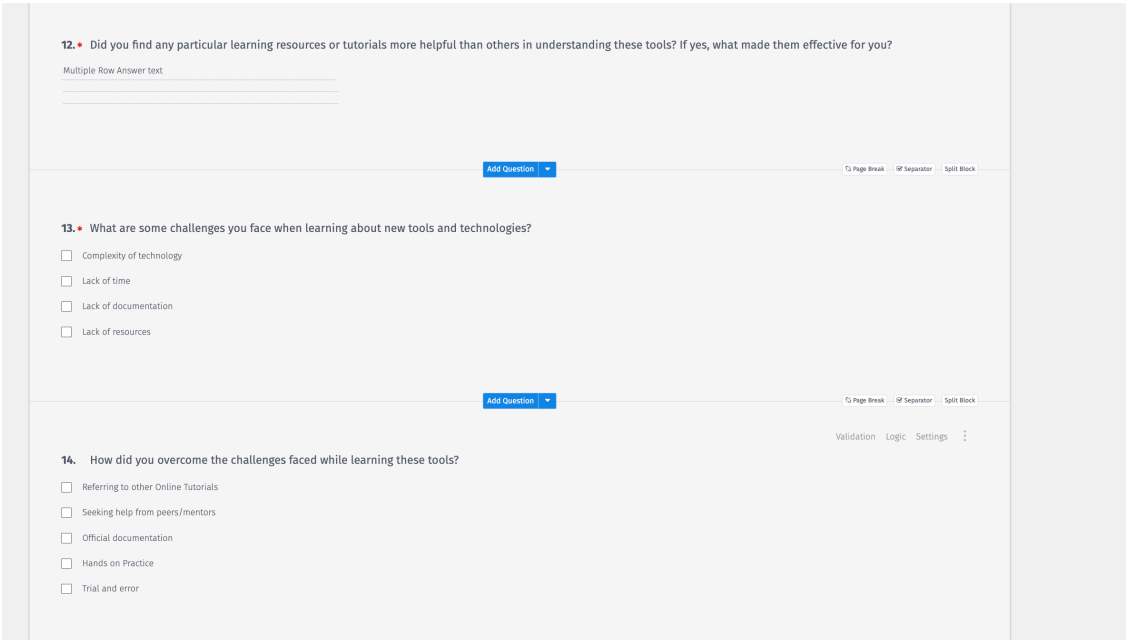


Figure A.12: Student Survey – Page 4



12.* Did you find any particular learning resources or tutorials more helpful than others in understanding these tools? If yes, what made them effective for you?

Multiple Row Answer text

Add Question | Page Break | Separator | Split Block

13.* What are some challenges you face when learning about new tools and technologies?

- Complexity of technology
- Lack of time
- Lack of documentation
- Lack of resources

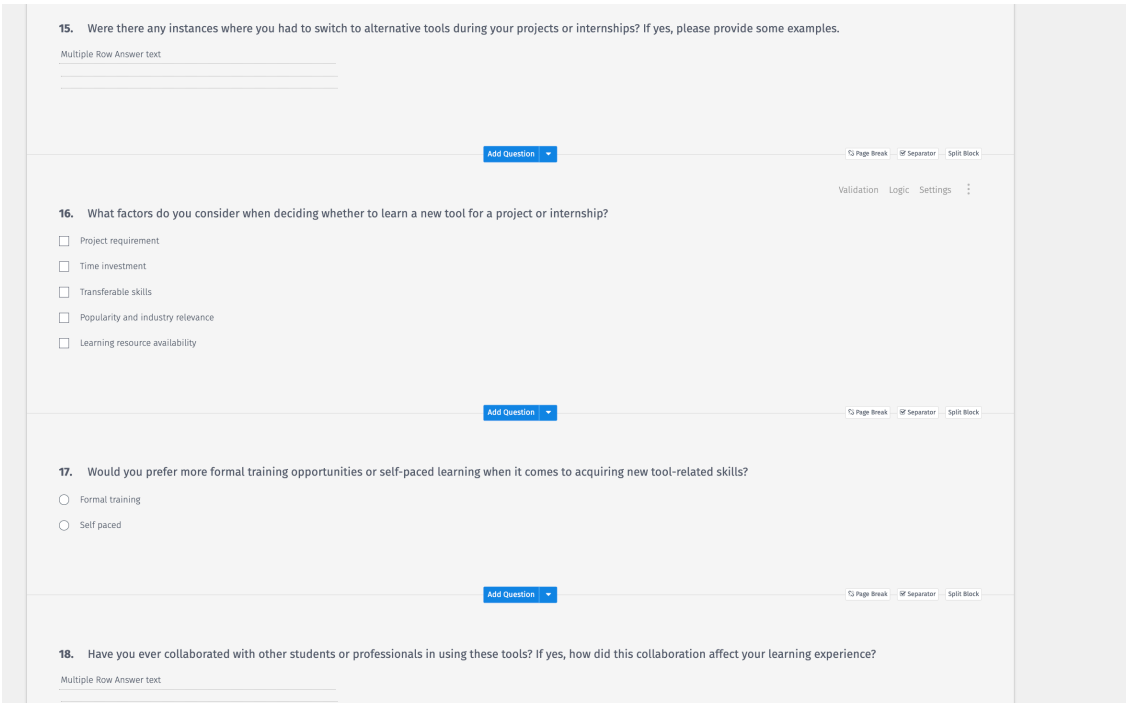
Add Question | Page Break | Separator | Split Block

Validation Logic Settings

14. How did you overcome the challenges faced while learning these tools?

- Referring to other Online Tutorials
- Seeking help from peers/mentors
- Official documentation
- Hands on Practice
- Trial and error

Figure A.13: Student Survey – Page 5



15. Were there any instances where you had to switch to alternative tools during your projects or internships? If yes, please provide some examples.

Multiple Row Answer text

Add Question | Page Break | Separator | Split Block

Validation Logic Settings

16. What factors do you consider when deciding whether to learn a new tool for a project or internship?

- Project requirement
- Time investment
- Transferable skills
- Popularity and industry relevance
- Learning resource availability

Add Question | Page Break | Separator | Split Block

17. Would you prefer more formal training opportunities or self-paced learning when it comes to acquiring new tool-related skills?

- Formal training
- Self paced

Add Question | Page Break | Separator | Split Block

18. Have you ever collaborated with other students or professionals in using these tools? If yes, how did this collaboration affect your learning experience?

Multiple Row Answer text

Figure A.14: Student Survey – Page 6

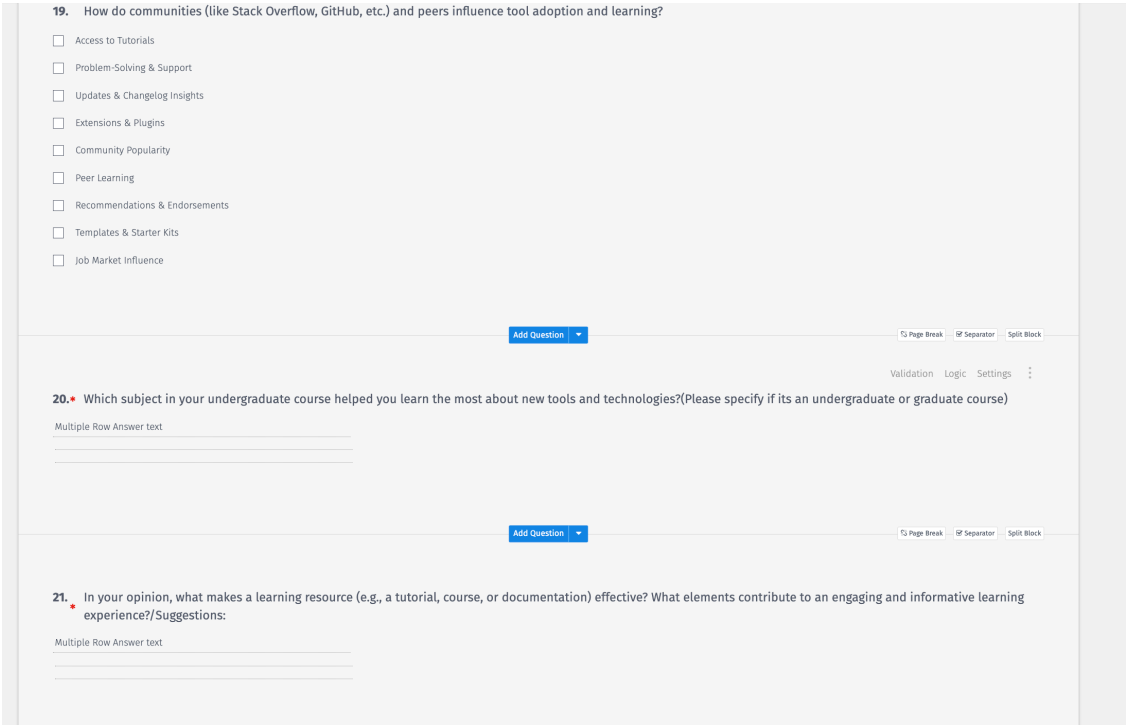


Figure A.15: Student Survey – Page 7

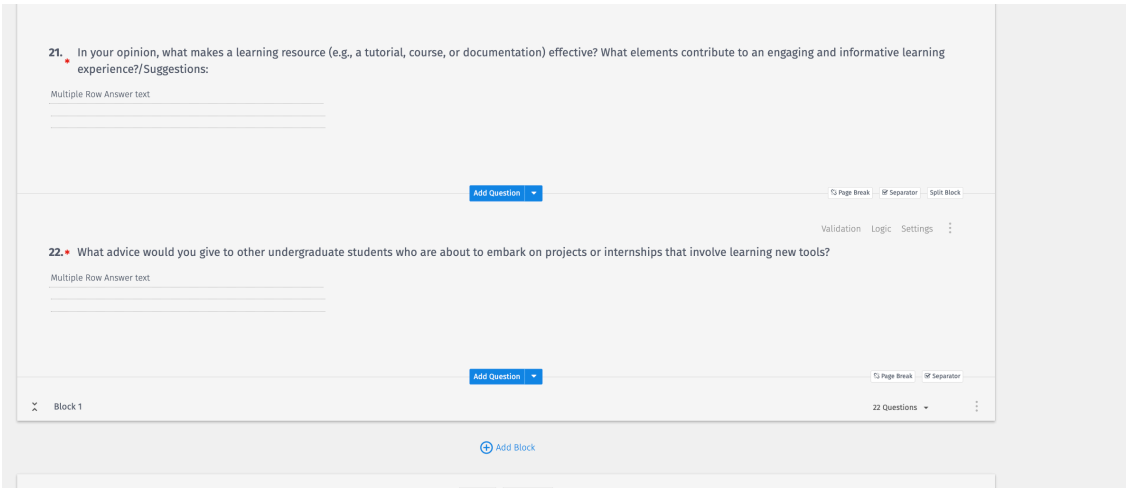


Figure A.16: Student Survey – Page 8