Impostor Syndrome in Final Year Computer Science Students: An Eye Tracking and Biometrics Study

Alyssia Chen, Carol Wong, Katy Tarrit, and Anthony Peruma(✉)

University of Hawai'i at Mānoa, Honolulu, Hawai'i, USA
{abc8,carolw8,katytm7,peruma}@hawaii.edu

Abstract. Imposter syndrome is a psychological phenomenon that affects individuals who doubt their skills and abilities, despite possessing the necessary competencies. This can lead to a lack of confidence and poor performance. While research has explored the impacts of imposter syndrome on students and professionals in various fields, there is limited knowledge on how it affects code comprehension in software engineering. In this exploratory study, we investigate the prevalence of imposter syndrome among final-year undergraduate computer science students and its effects on their code comprehension cognition using an eye tracker and heart rate monitor. Key findings demonstrate that students identifying as male exhibit lower imposter syndrome levels when analyzing code, and higher imposter syndrome is associated with increased time reviewing a code snippet and a lower likelihood of solving it correctly. This study provides initial data on this topic and establishes a foundation for further research to support student academic success and improve developer productivity and mental well-being.

Keywords: Cognitive Load and Performance · Eye Tracking · Biometrics · Heart Rate · Impostor Syndrome · Code Comprehension · Program Comprehension · Computer Science · Undergraduate Students

1 Introduction

Impostor syndrome, also known as impostor phenomenon, fraud syndrome, perceived fraudulence, or impostor experience, is a psychological phenomenon characterized by persistent self-doubt of intellect, skills, or accomplishments and feelings of fraudulence or inadequacy despite evidence of one's competence and accomplishments [5,6]. It is thought to affect high-achieving individuals across various domains, including higher education and high-skill professions [4,19]. Although there have been studies examining how women and minority groups cope with imposter syndrome [10], and how other variables are correlated such as self-esteem [11], and self-efficacy [14], there is limited research on the connection between impostor syndrome and undergraduate computer science students.

D. D. Schmorrow and C. M. Fidopiastis (Eds.): HCII 2024, LNAI 14694, pp. 22–41, 2024.
https://doi.org/10.1007/978-3-031-61569-6_2

As undergraduate students progress through their computer science degree, they not only acquire new skills but also refine existing ones, ultimately becoming proficient in designing and implementing software systems. By the time they reach the end of their academic journey, these students possess the qualifications necessary to begin their careers in the tech industry. It is natural to assume that at this stage, these students are confident in their skills and feel well-prepared for professional success. However, previous research indicates that students enrolled in STEM programs, including those related to computer science, often experience feelings of inadequacy, potentially impacting their confidence in their skills [15]. As such, this study aims to examine the prevalence and relationship of impostor syndrome with code comprehension performance in fourth-year computer science students through an augmented cognition approach.

1.1 Goal and Research Questions

Code comprehension is an essential activity in software development and maintenance. Source code conveys the system's behavior, through identifier names [13], which developers rely on to understand the code they are working on to fix defects or incorporate feature changes [21]. As computer science students prepare to enter the workforce, feelings of self-doubt and insecurity associated with impostor syndrome can undermine their ability to comprehend codebases, thereby negatively impacting their confidence and job performance. Therefore, this study aims to understand how impostor syndrome impacts code comprehension. We envision our study laying the foundation for future research in this area while also providing initial insights to improve computer science education and support students as they transition to professional roles. To this extent, we aim to answer the following research questions (RQs):

RQ 1: To what extent are final-year undergraduate computer science students confident in their program comprehension skills? This research question intends to examine the level of impostor syndrome exhibited by computer science students who are about to graduate. The research question assesses the demographic factors closely associated with impostor syndrome.

RQ 2: How does impostor syndrome affect cognitive processes involved in comprehending code? This research question aims to better understand cognitive processes involved in comprehending code, and how impostor syndrome can affect these processes. This is achieved by tracking eye movements, monitoring heart rates, and taking other measurements, such as the confidence of participants, time taken to complete a code question, and how often participants answered the coding question correctly.

1.2 Contribution

The main contributions from this work are as follows:

- We provide preliminary yet promising findings on the extent to which impostor syndrome affects the code comprehension cognition of experienced undergraduate students.
- This study establishes groundwork for further research exploring interventions against impostor syndrome.
- We make our dataset publicly accessible for replication/extension purposes.

1.3 Paper Structure

The rest of this paper is organized as follows: Section 2 presents a review of related work on imposter syndrome in software engineering. Section 3 describes the methodology used for this study. Section 4 provides answers to both research questions and reports on the results of the proposed experiments. Section 5 discusses the potential threats to the validity of the study. Finally, Sect. 6 concludes by summarizing the findings and suggesting potential directions for future work.

2 Related Work

In this section, we report on related work in this area. While there exist studies that examine imposter syndrome in students and industry professionals, we limit our focus to the literature that addresses imposter syndrome among students who are either taking a computing course or studying a computing subject/topic, as well as software engineering professionals.

In a survey with 200 undergraduate computer science students, Rosenstein et al. [15] show that 57% of the surveyed students exhibited frequent feelings of the Impostor Phenomenon. The authors also highlight that women experienced these feelings more frequently (71%) compared to men (52%). Heels and Devlin [8] examine the roles chosen by female students in a large software engineering team project and report that despite strong academic backgrounds, female students tend to opt for less technical roles than male students. The authors recommend exploring how unconscious bias or imposter syndrome affects female students. Interviews with 16 Digital Technologies teachers from primary and secondary schools by Varoy et al. [20] show that female students experience imposter syndrome more than male students and feel they are not making as much progress as male students, even when they are doing better work. The teachers note that male students tend to loudly proclaim their achievements, while female students work more quietly and cooperatively. This led the female students to underestimate their own abilities and progress. In a study on coding bootcamps, Thayer and Ko [17] found that imposter syndrome can act as an informal boundary to entering the software industry and persist even after gaining experience and employment. Additionally, participants in coding bootcamps can also experience imposter syndrome, similar to those in the software industry.

A survey by Ginter [7] of 104 software engineers shows that individuals exhibiting feelings of imposter syndrome usually had insecure attachment styles,

particularly anxiety and avoidance. The study also found that anxious attachment styles in individuals with imposter syndrome were linked to higher levels of depression and anxiety. In a study that surveyed 94 women employed at a global technology company, Trinkenreich et al. [18] note that imposter syndrome was a challenge women faced in software development teams and could lead to them leaving the organization. In an online survey of 134 Finnish women, Hyrynsalmi and Hyrynsalmi [9] reports that women are motivated to transition to the software industry but encounter challenges such as a lack of career counseling, uncertainty about the required education, and issues related to self-esteem and imposter syndrome.

3 Study Design

In this section, we provide details about the design of this study. At a high level, the study consisted of participants reviewing a set of code snippets and answering questions through online questionnaires. While reading code snippets, participants' eye movements and heart rates were monitored. We will elaborate on these activities below. The experiment took place in a private room without any windows, where only the participant and one investigator (i.e., an author) were present. A 24-inch monitor was used to display code snippets and questionnaires. The investigator used a secondary monitor, mouse, and keyboard to manage the experiment. After the calibration stage and before starting the experiment, each participant went through a trial run. This allowed to check if the eye tracker and biometric devices were functioning adequately but also to make sure participants understood and got familiar with the task. A couple trial code snippets were then displayed during this process and have not been included in the results. This study was approved by the Institutional Review Board of our institute and all participants provided consent prior to participating. The high-resolution images of the code snippets, survey questionnaire, and participant data are available at: [1].

3.1 Survey Design

As part of our study, we used three types of online questionnaires - a demographic pre-questionnaire, a code snippet questionnaire, and an impostor syndrome post-questionnaire. To construct and host these questionnaires, we leveraged Qualtrics. All participants answered survey questions using the same workstation that was connected to the eye tracker and biometrics device. Below, we have provided a brief description of each questionnaire.

Prior to code comprehension activities and questions, participants completed a pre-questionnaire capturing their demographics. All questions were required and are shown in Table 1.

During the coding analysis task, participants were presented with five individual code snippets, each accompanied by one single-choice question. The question

Table 1. Pre-Questionnaire capturing demographic details.

No.	Question	Type
1	What best describes your gender?	Single-Choice
2	What best describes your ethnicity?	Multi-Choice
3	What is your current academic year?	Single-Choice
4	What is your primary field of study/major?	Single-Choice
5	Are you a first generation college student?	Single-Choice
6	Do you have an immediate family member who has worked or is working in the software industry? (this includes internships)	Single-Choice
7	How many years of experience in the software industry do you have? (this includes internships)	Single-Choice
8	How many years of Java programming experience do you have?	Single-Choice
9	What is your preferred programming language?	Single-Choice
10	Do you have a software engineering-related job lined up post-graduation?	Single-Choice
11	How would you rate your overall experience with your programming education?	Single-Choice

required participants to either identify the correct output of the code or identify the number of the line where the code was causing a logical, runtime, or compile-time error. Participants only had five minutes to read each code snippet and answer the associated question. To reduce the possibility of participants guessing the answer, each question included an "I don't know" option. Section 4 provides screenshots of each code snippet, along with their associated question and multiple options for answer. Right after participants provided an answer to the question related to a specific code snippet, they were asked to provide feedback on the confidence-level for the answer they selected. After completing code snippets' comprehension activities, participants were asked to fill out a post-questionnaire to assess the extent of their imposter syndrome. For this study, Clance IP Scale questions [5] were customized to focus only on source code analysis and troubleshooting[1]. Those questions can be found in Table 2. All participants were required to answer all questions in this questionnaire.

3.2 Code Snippets

In this study, each participant was required to review five code snippets and answer their associated question. Code snippets were in the Java programming language as it is widely used in multiple courses at the University, and all students are familiar with this programming language. Code snippets covered concepts such as data structures, recursion, sorting, and string analysis, which are taught in undergraduate-level courses and are usually asked during job interviews in industry. Table 3 provides a summary of code snippets. The code snippets and their associated question are shown in Sect. 4.

[1] From The Impostor Phenomenon: When Success Makes You Feel Like A Fake (pp. 20–22), by P.R. Clance, 1985, Toronto: Bantam Books. Copyright 1985 by Pauline Rose Clance. Reprinted by permission. Do not reproduce without permission from Pauline Rose Clance, drpaulinerose@comcast.net.

Table 2. Post-Questionnaire containing the customized Clance IP Scale questions that focus on source code analysis and troubleshooting.

No.	Question	Type
1	I have often succeeded in programming tasks, even though I was afraid that I would not do well before I started working on it.	Single-Choice
2	I can give the impression that I'm more competent in my programming skills than I really am.	Single-Choice
3	I tend to avoid programming tasks and have a sense of dread when others assess my programming work.	Single-Choice
4	When people praise me for my code analysis and troubleshooting abilities, I'm afraid I won't be able to live up to their expectations of me in the future.	Single-Choice
5	I sometimes think my success in code analysis and troubleshooting is due to external factors (i.e., environmental or people) rather than due to my skills.	Single-Choice
6	I'm afraid people important to me may find out that I'm not as capable at programming as they think I am.	Single-Choice
7	I tend to remember the incidents in which I have not done my best in programming more than those times I have done my best.	Single-Choice
8	I rarely do a programming task as well as I'd like to do it.	Single-Choice
9	Sometimes I feel or believe that my success in analyzing and troubleshooting code has been the result of some kind of accident.	Single-Choice
10	It's hard for me to accept compliments or praise about my programming skills or accomplishments	Single-Choice
11	At times, I feel my success in code analysis and troubleshooting has been due to some kind of luck.	Single-Choice
12	I'm disappointed at times in my code analysis and troubleshooting accomplishments and think I should have accomplished much more.	Single-Choice
13	Sometimes I'm afraid others will discover how much code analysis and troubleshooting knowledge or ability I really lack.	Single-Choice
14	I'm often afraid that I may fail at a new code analysis and troubleshooting assignment or undertaking, even though I generally do well at what I attempt.	Single-Choice
15	When I've succeeded at programming and received recognition for my accomplishments, I have doubts that I can keep repeating that success.	Single-Choice
16	If I receive a great deal of praise and recognition for my code analysis and troubleshooting accomplishments, I tend to discount the importance of what I've done.	Single-Choice
17	I often compare my code analysis and troubleshooting abilities to those around me and think they may be more intelligent than I am.	Single-Choice
18	I often worry about not succeeding with a programming task, even though others around me have considerable confidence that I will do well.	Single-Choice
19	If I'm going to gain recognition for my code analysis and troubleshooting skills, I hesitate to tell others until it is an accomplished fact.	Single-Choice
20	I feel bad and discouraged if I'm not"the best" or at least "very special" in code analysis and troubleshooting situations that involve achievement.	Single-Choice

Table 3. Description of the five code snippets utilized in this study.

Id	Description	Task
1	This code snippet involves array size manipulation within a loop that will cause a runtime issue. Unlike the error in the other code snippets, participants were not explicitly informed that an error exists in the code.	Determine the output
2	This is an implementation of the bubble sort algorithm. However, there is an error in the condition used in the loop, which will lead to the list being sorted incorrectly. The method's name is called 'bubbleSort', so the participant is aware of the intended behavior of the code.	Identify the number of the line where the error occurs
3	This is a recursive function that prints a sequence of digits. There are no errors in the code.	Determine the output
4	This code snippet accepts a provided string and prints the length of the last word in the provided string. The names of the identifiers in the code do not indicate the code's behavior. There are no errors in this code.	Determine the output
5	This code checks if a given string is a palindrome. The implementation contains an error in the calculation that results in a runtime error.	Identify the number of the line where the error occurs

3.3 Eye Tracker and Biometric Device

Eye movements and physiological samples such as heart rate were recorded respectively using the Gazepoint GP3 HD Eye Tracker and the Biometrics device. These devices are research-grade and are commonly used for academic as well as medical research [2,3]. The eye tracker has a sampling rate of 150 Hz to reduce the chance of loss of tracking and a 0.5–1.0°C of visual angle accuracy. During the experiment, participants sat on a chair facing a computer monitor where code snippets and surveys were presented to them. Prior to the experiment, the camera was calibrated using the Gazepoint Control software. Participants were asked to make saccades to nine targets presented on the screen for calibration purposes. Those nine targets were presented at the top, at the middle and at the bottom of the screen going from the left through the middle to the right of the screen. The heart rate was measured using a finger sensor module. The Gazepoint Analysis software was utilized to manage the experiment, including starting/stopping the eye tracker and the creation of areas of interest (i.e., AOIs).

3.4 Data Preprocessing

Once data collection was completed, data for analysis was prepared using Python. The temporal gaze point data (a single gaze point represents where a user looked and when in milliseconds) from Gazepoint was first cleaned by removing durations of time where: (1) the code was not on the screen due to latency from Qualtrics when displaying the survey question, (2) the time between code snippets, and (3) the participant was looking away from the screen and was not focused on the code snippet. The times for (1) and (3) were determined manually by two investigators watching the video recordings with precision up to three decimal places. The cutoff time for (2) was automated by trimming the data up to when the participant submitted their answer for a code snippet question.

The following aggregate measures were calculated for each participant: average heart rate, total time spent on a snippet, and most looked at AOI (specifically, only AOIs containing code were analyzed for this study; answer choices below each code snippet were not considered). Duration spent in each AOI was determined by summing the differences in time between the first instance when a gaze point was identified to be within an AOI and the first instance when the participant moved to a different AOI. All durations for the AOIs were then normalized by the number of words, excluding non-alphanumeric characters (as described in Sharafi et al.[16]).

Other calculations for the data included determining levels of imposter syndrome (detailed in RQ1 of Sect. 4) and determining the correctness of answers (a binary true/false if the answer was correct) for each code snippet which were determined by two investigators. Additionally, recoding of ordinal variables to be numerical were coded starting from the number zero and non-codeable responses were designated as NaN.

3.5 Participants

In this study, fifteen final-year undergraduate students enrolled in the Computer Science program at the Information and Computer Sciences Department of the University of Hawai‘i at Mānoa were recruited. These participants were not given any monetary compensation or extra credit for participating in the study; participation was voluntary.

Out of the fifteen participants who took the survey, nine of them identified as male, five as female, and one as non-binary/third gender. Moreover among the participant pool, ten identified themselves as Asian only, one as White/Caucasian, one as Black/African American, and the remaining three participants identified with two or more ethnicities.

In terms of Java programming experience, eight participants indicated they had one to two years of experience, four participants mentioned they had three to five years of experience, and three participants reported they had less than one year of experience. Additionally, six participants had between one to two years of industry experience, five participants had less than a year of industry experience, two participants had three to five years of industry experience, and the remaining two participants had no prior industry experience. Table 4 provides a breakdown of participants' demographics.

3.6 Pilot Study

We followed the best practice of conducting a pilot study to identify any flaws in our methodology. For this purpose, we recruited four students to participate in a pilot study, who were not included in the actual study. All data collected during the pilot study were discarded after its completion. During the pilot run, we identified certain survey questions that needed to be reworded to improve clarity as well as code snippets that required consistent formatting. We also made improvements to the positioning of the eye tracker and the participant's

Table 4. Demographic details of the participants in the study.

Participant ID	Gender	Ethnicity	Years of Exp. in Software Industry	Years of Exp. in Java
1	Male	White or Caucasian	1–2 years	1–2 years
2	Female	Asian	1–2 years	1–2 years
3	Female	Asian, White or Caucasian	1–2 years	3–5 years
4	Male	Asian	None	3–5 years
5	Male	Asian	>1 year	3–5 years
6	Female	Asian	1–2 years	1–2 years
7	Female	Asian	3–5 years	3–5 years
8	Non-binary / third gender	Asian	>1 year	>1 year
9	Male	Black or African American	None	1–2 years
10	Male	Asian	1–2 years	>1 year
11	Female	Asian, Native Hawaiian	1–2 years	1–2 years
12	Male	Asian	>1 year	1–2 years
13	Male	Asian	3–5 years	1–2 years
14	Male	Asian	>1 year	>1 year
15	Male	American Indian or Alaska Native, Asian, White or Caucasian	>1 year	1–2 years

chair. Furthermore, the pilot study helped us construct a script/instructions to follow when conducting the actual study.

4 Results

In this section, we report the study's results by answering our RQs.

4.1 RQ 1: To what extent are final-year undergraduate computer science students confident in their program comprehension skills?

Based on the Clance Imposter Phenomenon (IP) Scale [5], participants were sorted into four Imposter Phenomenon Characteristics (IPC) categories: Few IPC for those who demonstrated few imposter syndrome characteristics, Moderate IPC for those with moderate characteristics, Frequent IPC, and Intense IPC, with each corresponding to the severity level of IP. These levels correspond to how frequently and seriously IP interferes with a person's life.

Table 5. Range of values associated with each IPC level

IPC Level	Scores
Few	> 40
Moderate	41–60
Frequent	61–80
Intense	< 80

Once participants filled out the post-questionnaire based on a version of the Clance IP Scale adapted to the task, their score was calculated as follows: Each question had a range of multiple-choice answers, which were "not at all true," "rarely," "sometimes," "often," and "very true." Each answer mapped to a numerical value from 1 to 5, respectively. These values are

then tallied to create an aggregate score for each participant and used to classify each participant into one of the four IPC groups. The range of values associated with each IPC category can be found in Table 5.

Of the fifteen participants, three had Few IPC, five had Moderate IPC, five had Frequent IPC, and two had Intense IPC. Of the three participants with Few IPCs, all three identified themselves as male. In the Moderate and Frequent IPC groups, each comprising five individuals, two participants identified themselves as female and three as male. Amongst the two participants with Intense IPC, one identified as female and the other one as non-binary/third gender.

A t-test to compare the IPC scores between males and females (the participant who responded "non-binary" was omitted) found that the nine males (M = 50.56, SD = 12.32) compared to the five females (M = 68.80, SD = 12.28) scored significantly lower on the modified Clance IP scale, $t(12) = -2.658$, $p = .021$. Therefore, females were associated with higher levels of the imposter phenomenon, particularly in relation to source code analysis and troubleshooting.

One-way ANOVA tests were additionally conducted to compare how IPC scores were related to software industry experience and Java experience. Findings show that there is no significant difference in IPC scores so there is no relationship between these variables ($p > .05$ for both).

Summary for RQ1. We found that female students had higher characteristics of imposter syndrome. The number of years of experience in the software industry or the tested programming language did not seem to correlate to the level of characteristics of imposter syndrome.

4.2 RQ 2: How Does Imposter Syndrome Affect the Cognitive Processes Involved in Comprehending Code?

To answer this research question, we examine the eye tracking and biometrics results of participants comprehending each code snippet in the study. Below, for each code snippet, we discuss the average heart rate, Areas of Interest (AOI) metrics, time spent by participants on the snippet, the percentage of correct answers, and the extent to which participants are confident with their answers. Additionally, Fig. 1 displays all five code snippets that were included in our study, along with their corresponding question.

AOIs are regions of a stimulus from which quantitative metrics can be derived [16]. Each AOI contains its own set of metrics, e.g., the average time participants took to look at a specific AOI. Each line of code, excluding those solely comprising of curly braces, was designated an AOI. Each stimulus had an average of 13.4 AOIs. As mentioned in Sect. 3, each AOI was normalized by the number of words it contained on its line in order to allow fair comparison between AOIs (words were characterized as sequences of alphanumeric characters that are separated by spaces after excluding non-alphanumeric characters).

The AOIs were then grouped into one of ten different categories, depending on the functionality of the line of code. These categories are as follows: else statement, for loop, if statement, import statement, method call, method declaration,

1. What is the correct output:

```
1   import java.util.ArrayList;
2   public static void main(String[] args) {
3           List<Integer> list = new ArrayList<>();
4           list.add(1);
5           list.add(2);
6           list.add(3);
7           for (Integer item : list) {
8               if (item.equals(1)) {
9                   list.remove(item);
10              }
11          }
12      for (int i = 0; i < list.size(); i++) {
13              int curr = list.get(i);
14              System.out.print(curr);
15      }
16  }
```

○ Output:_____ ○ Null

○ Compile/Runtime error ○ I don't know

2. If there's an error, which line number contains it?

```
1   static void bubbleSort(int[] arr) {
2       int n = arr.length;
3       int temp = 0;
4       for(int i = 0; i < n; i++){
5           for(int j = 1; j < i; j++){
6               if(arr[j-1] > arr[j]){
7                   temp = arr[j-1];
8                   arr[j-1] = arr[j];
9                   arr[j] = temp;
10              }
11          }
12      }
13  }
```

○ Line Number: _____ ○ I don't know

○ There is no error

3. What is the correct output:

```
1   public static void emptyVase(int flowersInVase) {
2       if (flowersInVase > 0) {
3           emptyVase(flowersInVase - 1);
4           System.out.print(flowersInVase);
5       }
6   }
7
8   public static void main (String[] argos){
9       emptyVase(7);
10  }
```

○ Output:_____ ○ Null

○ Compile/Runtime error ○ I don't know

4. What is the correct output:

```
1   public int function(String input) {
2       int length = 0;
3       for (int i = input.length() - 1; i >= 0; i--) {
4           if (s.charAt(i) != ' ') {
5               length++;
6           } else {
7               if (length > 0)
8                   return length;
9           }
10      }
11      return length;
12  }
13
14  public static void main(String[] args) {
15      System.out.println(function("Hello World"));
16  }
```

○ Output:_____ ○ Null

○ Compile/Runtime error ○ I don't know

5. If there's an error, which line number contains it?

```
1   public boolean palindromeChecker (String str) {
2       String reversedStr = "";
3       int length = str.length();
4       for (int i = (length - 1); i >=0; --i) {
5           reversedStr = reversedStr - str.charAt(i);
6       }
7       str = str.toLowerCase();
8       reversedStr = reversedStr.toLowerCase();
9       if (str.equals(reversed_str)) {
10          return true;
11      } else {
12          return false;
13      }
14  }
```

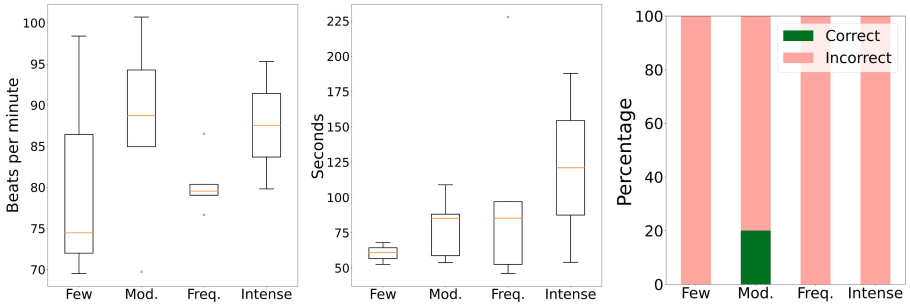
○ Line Number: _____ ○ I don't know

○ There is no error

Fig. 1. The five code snippets that were part of our study along with their associated question and answer choices.

recursive method call, return statement, variable assignment, and variable declaration. An example of a method declaration is "public static void main(String[] args) {", while an example of a method call is "list.add(1)."

Code Snippet 1 - Array Size Manipulation



(a) Average Heart Rate (b) Time Spent on Snippet (c) % Correctly Answered

Fig. 2. Biometrics and Other Measures Averaged per IPC Group for Snippet 1
Average heart rate and time spent was further averaged within each IPC grouping (Few, Moderate, Frequent, and Intense IPC). The percentage of participants who scored correctly in each IPC grouping is displayed in the last figure.

It was found that for Code Snippet 1, only one participant from the Moderate IPC group answered the problem correctly (see Fig. 2c). The lowest median for the average heart rate occurred in the Few IPC group, whereas the highest median average heart rate occurred in the Moderate IPC group (see Fig. 2a). The amount of time participants took before submitting an answer was the lowest for those in the Few IPC group and the highest in the Intense IPC group (see Fig. 2b).

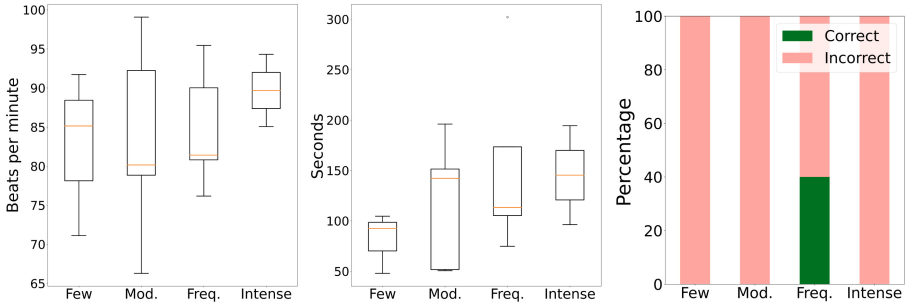
The lines of code that participants spent the longest duration on were import statements for Few IPC, Moderate IPC, and Frequent IPC (for each row in Table 6, the percentage was highest for these three IPC groups). Those in the Intense IPC spent the longest duration on if statements (23.87%). For all but those in Frequent IPC, the least amount of time was spent on method declarations.

Table 6. Duration spent on a Code Category in Snippet 1

	For Loop	If Stmt	Import Stmt	Method Call	Method Decl	Var Decl	Multiple
Few IPC	16.18% (0.46)	18.28% (0.52)	21.93% (0.62)	21.29% (0.6)	10.71% (0.3)	11.14% (0.32)	0.47% (0.01)
Moderate IPC	17.17% (0.8)	12.7% (0.59)	25.17% (1.17)	18.25% (0.85)	11.62% (0.54)	14.51% (0.67)	0.58% (0.03)
Frequent IPC	17.76% (0.91)	10.93% (0.56)	30.31% (1.56)	16.01% (0.82)	12.66% (0.65)	12.07% (0.62)	0.24% (0.01)
Intense IPC	20.63% (1.18)	23.87% (1.36)	14.71% (0.84)	17.65% (1.01)	9.67% (0.55)	11.26% (0.64)	2.21% (0.13)

Row percentages for each IPC are shown in each cell with red highlights denoting largest percentage and green highlights denoting smallest (ignoring the Multiple category). The duration normalized by number of words are in parenthesis.

Code Snippet 2 - Bubble Sort



(a) Average Heart Rate (b) Time Spent on Snippet (c) % Correctly Answered

Fig. 3. Biometrics and Other Measures Averaged per IPC Group for Snippet 2 Average heart rate and time spent was further averaged within each IPC grouping (Few, Moderate, Frequent, and Intense IPC). The percentage of participants who scored correctly in each IPC grouping is displayed in the last figure.

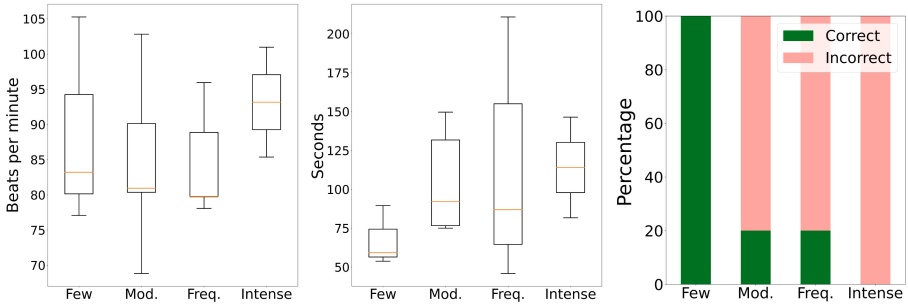
As shown in Fig. 3, only two participants from the Frequent IPC group answered the problem correctly. The average heart rate was the lowest in the Moderate IPC group and Highest in the intense IPC group. Participants in the Few IPC group spent the least amount of time and those in the Intense IPC group spent the most.

Referring to Table 7, those from the Few IPC, Moderate IPC, and Frequent IPC spent the least amount of time on the if statements, while those in the Intense IPC group spent the least amount of time on variable assignments. Besides the Moderate IPC group, every group spent the most time on method declarations.

Table 7. Duration spent on a Code Category in Snippet 2

	For Loop	If Stmt	Method Decl	Var Asgmt	Var Decl
Few IPC	14.09% (0.66)	8.33% (0.39)	39.24% (1.84)	11.82% (0.55)	26.52% (1.24)
Moderate IPC	16.45% (1.01)	13.05% (0.8)	24.17% (1.48)	25.87% (1.59)	20.46% (1.26)
Frequent IPC	19.54% (1.69)	10.92% (0.95)	36.96% (3.2)	12.79% (1.1)	19.89% (1.72)
Intense IPC	17.95% (1.01)	13.02% (0.73)	36.51% (2.06)	12.7% (0.72)	19.74% (1.11)

Code Snippet 3 - Recursion



(a) Average Heart Rate (b) Time Spent on Snippet (c) % Correctly Answered

Fig. 4. Biometrics and Other Measures Averaged per IPC Group for Snippet 3
Average heart rate and time spent was further averaged within each IPC grouping (Few, Moderate, Frequent, and Intense IPC). The percentage of participants who scored correctly in each IPC grouping is displayed in the last figure.

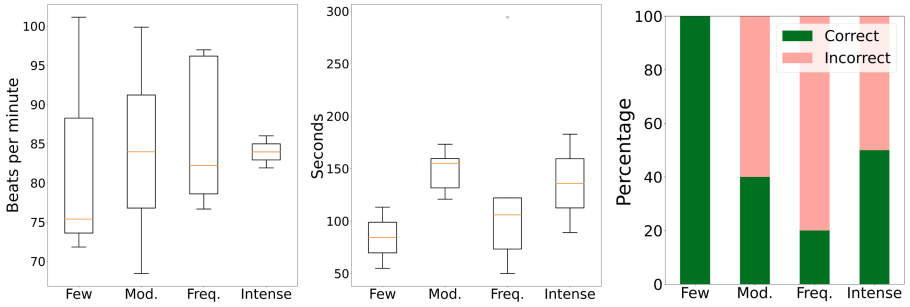
For Code Snippet 3, all of the participants in the Few IPC group answered the problem correctly, whereas no participants from the Intense IPC group answered correctly (Fig. 4). One participant from the Moderate and Frequent IPC groups answered correctly. The lowest median average heart rate was from the Frequent IPC group, and the highest median was from the Intense IPC group. Participants from the Intense IPC group spent the longest amount of time on the snippet compared to those in the Few IPC group, who spent the shortest.

This snippet had less consistent results across the IPC groupings as can be seen in Table 8. However, it can seen that those in the Few IPC group spent the most time on if-statements and the least amount of time on recursive calls, whereas those in the Intense IPC group spent most of their time on method declarations and the least amount of time on if-statements.

Table 8. Duration spent on a Code Category in Snippet 3

	If Stmt	**Method Call**	**Method Decl**	**Recursive Call**
Few IPC	31.25% (1.08)	22.73% (0.79)	30.88% (1.07)	15.14% (0.52)
Moderate IPC	28.7% (2.39)	15.21% (1.27)	14.06% (1.17)	42.03% (3.5)
Frequent IPC	24.38% (2.13)	33.73% (2.95)	21.83% (1.91)	20.06% (1.75)
Intense IPC	8.06% (0.6)	34.43% (2.58)	34.54% (2.59)	22.96% (1.72)

Code Snippet 4 - String Analysis



(a) Average Heart Rate (b) Time Spent on Snippet (c) % Correctly Answered

Fig. 5. Biometrics and Other Measures Averaged per IPC Group for Snippet 4
Average heart rate and time spent was further averaged within each IPC grouping (Few, Moderate, Frequent, and Intense IPC). The percentage of participants who scored correctly in each IPC grouping is displayed in the last figure.

Referring to Fig. 5, seven participants answered this question correctly, including all three from Few IPC, three between Moderate and Frequent IPC, and one from Intense IPC. The median average heart rate was similar between Moderate, Frequent, and Intense IPC, but those in the Few IPC group had the lowest average heart rate. For time spent on the snippet, participants with Intense IPC had the longest average duration; participants with Few IPC had the shortest average duration.

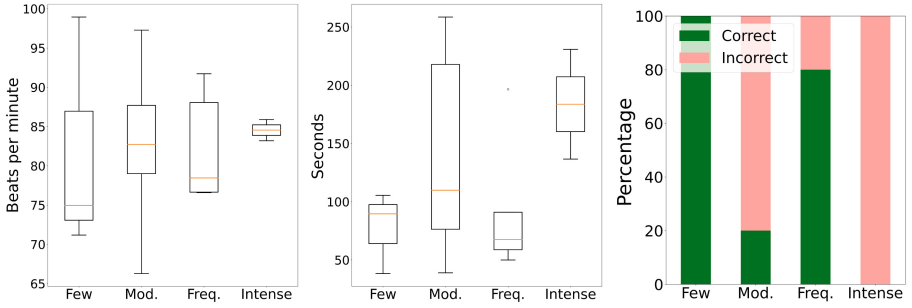
As seen in Table 9, participants with Few, Moderate, and Frequent IPC spent the longest duration on variable assignments. For code categories with the shortest duration, there was no consistency among the four IPC groups.

Table 9. Duration spent on a Code Category in Snippet 4

	Else Stmt	For Loop	If Stmt	Method Call	Method Decl	Return Stmt	Var Asgmt	Var Decl	Multiple
Few IPC	3.16% (0.26)	7.95% (0.65)	8.74% (0.72)	9.61% (0.79)	19.49% (1.6)	17.82% (1.46)	23.97% (1.97)	7.7% (0.63)	1.56% (0.13)
Moderate IPC	11.28% (1.83)	10.19% (1.66)	9.96% (1.62)	11.15% (1.81)	11.59% (1.88)	9.78% (1.59)	20.56% (3.34)	9.7% (1.58)	5.76% (0.94)
Frequent IPC	14.72% (2.29)	4.6% (0.72)	9.42% (1.47)	9% (1.4)	12.65% (1.97)	10.71% (1.67)	25.78% (4.02)	10.58% (1.65)	2.54% (0.4)
Intense IPC	15.12% (1.52)	7.69% (0.78)	5.32% (0.54)	7.72% (0.78)	24.46% (2.47)	8.23% (0.83)	12.54% (1.26)	15.35% (1.55)	3.57% (0.36)

Code Snippet 5 - Palindrome

For Code Snippet 5, eight participants answered correctly, again including all three from Few IPC, one from Moderate IPC, and four from Frequent IPC (See Fig. 6). No participant from the intense IPC group answered correctly. Those in the few IPC group had the lowest average heart rate, whereas those in the



(a) Average Heart Rate (b) Time Spent on Snippet (c) % Correctly Answered

Fig. 6. Biometrics and Other Measures Averaged per IPC Group for Snippet 5

Average heart rate and time spent was further averaged within each IPC grouping (Few, Moderate, Frequent, and Intense IPC). The percentage of participants who scored correctly in each IPC grouping is displayed in the last figure.

intense IPC had the highest. Similar to the other snippets, participants with Few IPC spent the least amount of time on the snippet, whereas those with Intense IPC spent the most amount of time.

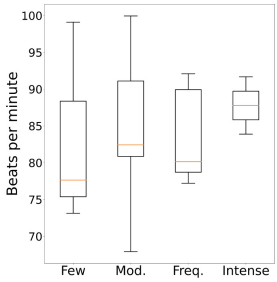
Referring to Table 10 all four IPC groupings spent the most time looking at method declarations. Those with Few and Moderate IPC spent the least amount of time on return statements, whereas those with Frequent and Intense IPC spent the least amount of time on if statements.

Table 10. Duration spent on a Code Category in Snippet 5

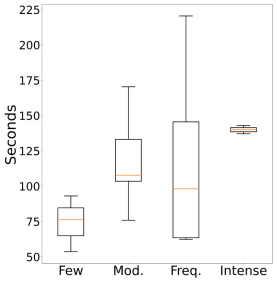
	Else Stmt	For Loop	If Stmt	Method Decl	Return Stmt	Var Asgmt	Var Decl
Few IPC	5.5% (0.33)	12.38% (0.75)	6.02% (0.37)	44.68% (2.72)	5.4% (0.33)	7.73% (0.47)	18.28% (1.11)
Moderate IPC	14.16% (1.81)	10.32% (1.32)	11.6% (1.48)	25.57% (3.26)	9.03% (1.15)	14.81% (1.89)	14.51% (1.85)
Frequent IPC	10.19% (0.74)	6.34% (0.46)	6.34% (0.46)	37.46% (2.74)	7.27% (0.53)	8.99% (0.66)	23.41% (1.71)
Intense IPC	20.88% (2.89)	3.94% (0.54)	1.67% (0.23)	40.36% (5.58)	9.97% (1.38)	7.14% (0.99)	16.04% (2.22)

Overall Analysis. When averaging time across all 5 code snippets and for each imposter syndrome level, there appears to be a trend where the median of the time spent increases as participants score higher on Clance IP Scale (Fig. 7b). When averaging how many times participants in each IPC group answered questions correctly across all code snippets, there was a downward trend (Fig. 7c). To measure the extent of the relationship between these variables, we conducted

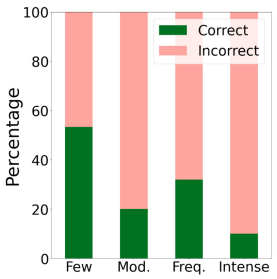
a Pearson correlation test. The Pearson correlation test yielded statistically significant (i.e., p-value < 0.05) correlation coefficients of 0.52, equating to a moderate positive correlation between IPC score and time spent on a snippet and -0.52, equating to a moderate negative correlation between IPC score and average correctness. While in Fig. 7a, there appears to be a positive trend between average heart rate and IPC, it was not statistically significant. Similarly, while there seems to be a pattern of those with high IPC being less confident in their answers in 7d, no significant results were found.



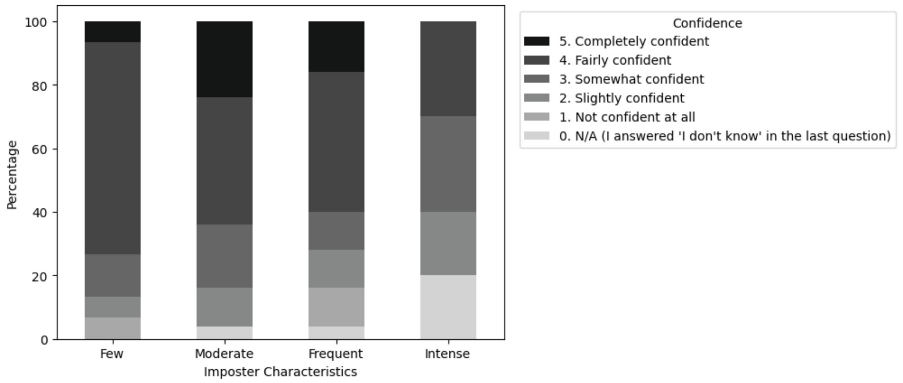
(a) *AvgHeartRate*



(b) *AvgTimeSpent*



(c) *CorrectlyAnswered*



(d) *ConfidenceinAllSnippets*

Fig. 7. Combined Results

To gain a deeper insight into overall how participants of varying IPC scores read code, a Pearson Correlation 2-tailed test was conducted to compare average normalized durations for each AOI code category (e.g., else-statement, for-loop). The test yielded a statistically significant (i.e., p-value < 0.05) correlation coefficient: 0.56 for the IPC score and Method Calls, equating to a positive moderate correlation, and 0.52 for the IPC score and Method Declarations. This could indicate that those with higher IPC scores tend to spend more time looking at code associated with these two categories.

Summary for RQ2. Students with higher characteristics of imposter syndrome were associated with spending longer on snippets and were less likely to answer the question related to each code snippet correctly. Across all code snippets, those with higher characteristics of imposter syndrome were more likely to look at the Method Call and Method Declaration code categories.

5 Threats to Validity

Although our study sample size is small and limited to a single institute, it remains valuable as an initial exploratory study providing a foundation for further research. Furthermore, research shows that program comprehension studies using an eye tracker have an average of 18.08 participants with a standard deviation of 9.90 [12]. Likewise, due to the small sample size, the small gender and race/ethnicity distribution in our participants' pool may not be representative of the general population. The number and type of code snippets used in this study in order to evaluate comprehension threaten the study's validity, as they may not represent real-world codebases that students will encounter in their careers. However, these code snippets represent basic programming concepts that students learn when preparing for their degree and are commonly asked in entry-level software engineering industry interviews.

The validity of our study may also be threatened by the limitations of the Gazepoint GP3 HD Eye Tracker and Biometrics Kit. However, these are research-grade devices that have been previously utilized in scientific publications [2,3]. Overall, while our study has some inherent limitations as an initial investigation, it establishes a valuable foundation for further research on this topic. Finally, as the experiment was conducted in a controlled lab environment, it may not fully reflect real-world software development situations and, therefore, could impact the validity of our results. However, the controlled setting helped us focus on the effects of imposter syndrome on code comprehension. Moreover, we followed established guidelines to set up our eye-tracking experiment [16].

6 Conclusion and Future Work

Imposter syndrome is a psychological barrier that can negatively affect the performance of students and professionals. While research on imposter syndrome in software engineering does exist, little is known about how it affects code comprehension cognition. In this exploratory study, we examine the level of imposter syndrome final-year undergraduate computer science students exhibit when comprehending code. We further measure the cognitive impact using an eye tracker and heart rate monitor. Our findings show that: (1) students identifying as males show lower levels of imposter syndrome, (2) higher levels of imposter syndrome are associated with increased duration of time spent on a snippet and lower chances of solving the problem correctly, and (3) those with higher imposter syndrome levels are more likely to look at method declarations and method calls.

While limited in scope, our study establishes a foundation for further research on imposter syndrome and its effects on core software engineering competencies.

Our future work involves working with industry professionals in a similar study to understand how imposter syndrome affects computer science professionals in their day-to-day work, including its impact on mental health.

Acknowledgments. We thank students who took the time to participate in our study.

Data Availability Statement. The high-resolution images of the code snippets, survey questionnaires, and participant data are available at: [1].

References

1. https://zenodo.org/doi/10.5281/zenodo.10656486
2. Eye-tracking technology for academic research | gazepoint. https://www.gazept.com/academic-research/?v=7516fd43adaa
3. Publications & citations | gazepoint. https://www.gazept.com/meet-the-team/publications/?v=7516fd43adaa
4. Chakraverty, D.: Faculty experiences of the impostor phenomenon in STEM fields. CBE Life Sci. Educ. **21**(4), ar84 (2022)
5. Clance, P.R.: Clance impostor phenomenon scale. Pers. Individ. Diff. (1985)
6. Clance, Pauline Rose Imes, S.A.: The imposter phenomenon in high achieving women: dynamics and therapeutic intervention. Psychother. Theor. Res. Pract. **15**(3) (1978). https://doi.org/10.1037/h0086006, https://doi.org/10.1037/h0086006
7. Ginter, A.N.: Imposter phenomenon, insecure attachment style, and mental health in software engineers: an examination of the moderating effect of self esteem. Ph.D. thesis, Alliant International University (2023)
8. Heels, L., Devlin, M.: Investigating the role choice of female students in a software engineering team project. In: Proceedings of the 3rd Conference on Computing Education Practice. CEP 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3294016.3294028
9. Hyrynsalmi, S., Hyrynsalmi, S.: What motivates adult age women to make a career change to the software industry? In: 2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), pp. 1–8 (2019). https://doi.org/10.1109/ICE.2019.8792630
10. Maji, S.: "They Overestimate Me All the Time:" exploring imposter phenomenon among indian female software engineers. metamorphosis. J. Manag. Res. **20**(2), 55–64 (2021). https://doi.org/10.1177/09726225211033699, http://journals.sagepub.com/doi/10.1177/09726225211033699
11. Naser, M.J., Hasan, N.E., Zainaldeen, M.H., Zaidi, A., Mohamed, Y.M.A.M.H., Fredericks, S.: Impostor phenomenon and its relationship to self-esteem among students at an international medical college in the middle east: a cross sectional study. Front. Med. **9**, 850434 (2022). https://doi.org/10.3389/fmed.2022.850434, https://www.frontiersin.org/articles/10.3389/fmed.2022.850434/full
12. Obaidellah, U., Al Haek, M., Cheng, P.C.H.: A survey on the usage of eye-tracking in computer programming. ACM Comput. Surv. **51**(1) (2018). https://doi.org/10.1145/3145904

13. Peruma, A., Mkaouer, M.W., Decker, M.J., Newman, C.D.: An empirical investigation of how and why developers rename identifiers. In: Proceedings of the 2nd International Workshop on Refactoring, pp. 26–33. IWoR 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3242163.3242169
14. Pákozdy, C., Askew, J., Dyer, J., Gately, P., Martin, L., Mavor, K.I., Brown, G.R.: The imposter phenomenon and its relationship with self-efficacy, perfectionism and happiness in university students. Curr. Psychol. (2023). https://doi.org/10.1007/s12144-023-04672-4
15. Rosenstein, A., Raghu, A., Porter, L.: Identifying the prevalence of the impostor phenomenon among computer science students. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, pp. 30–36. SIGCSE 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3328778.3366815
16. Sharafi, Z., Sharif, B., Guéhéneuc, Y.G., Begel, A., Bednarik, R., Crosby, M.: A practical guide on conducting eye tracking studies in software engineering. Empir. Softw. Eng. **25**, 3128–3174 (2020)
17. Thayer, K., Ko, A.J.: Barriers faced by coding Bootcamp students. In: Proceedings of the 2017 ACM Conference on International Computing Education Research, pp. 245–253. ICER 2017, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3105726.3106176
18. Trinkenreich, B., Britto, R., Gerosa, M.A., Steinmacher, I.: An empirical investigation on the challenges faced by women in the software industry: a case study. In: Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society, pp. 24–35. ICSE-SEIS '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3510458.3513018
19. Trinkenreich, B., Wiese, I., Sarma, A., Gerosa, M., Steinmacher, I.: Women's participation in open source software: A survey of the literature. ACM Trans. Softw. Eng. Methodol. **31**(4) (2022). https://doi.org/10.1145/3510460
20. Varoy, E., Luxton-Reilly, A., Lee, K., Giacaman, N.: Understanding the gender gap in digital technologies education. In: Proceedings of the 25th Australasian Computing Education Conference, pp. 69–76. ACE 2023, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576123.3576131
21. Von Mayrhauser, A., Vans, A.M.: Program comprehension during software maintenance and evolution. Computer **28**(8), 44–55 (1995). https://doi.org/10.1109/2.402076