

# "I Feel Like I'm Teaching in a Gladiator Ring": Barriers and Benefits of Live Coding

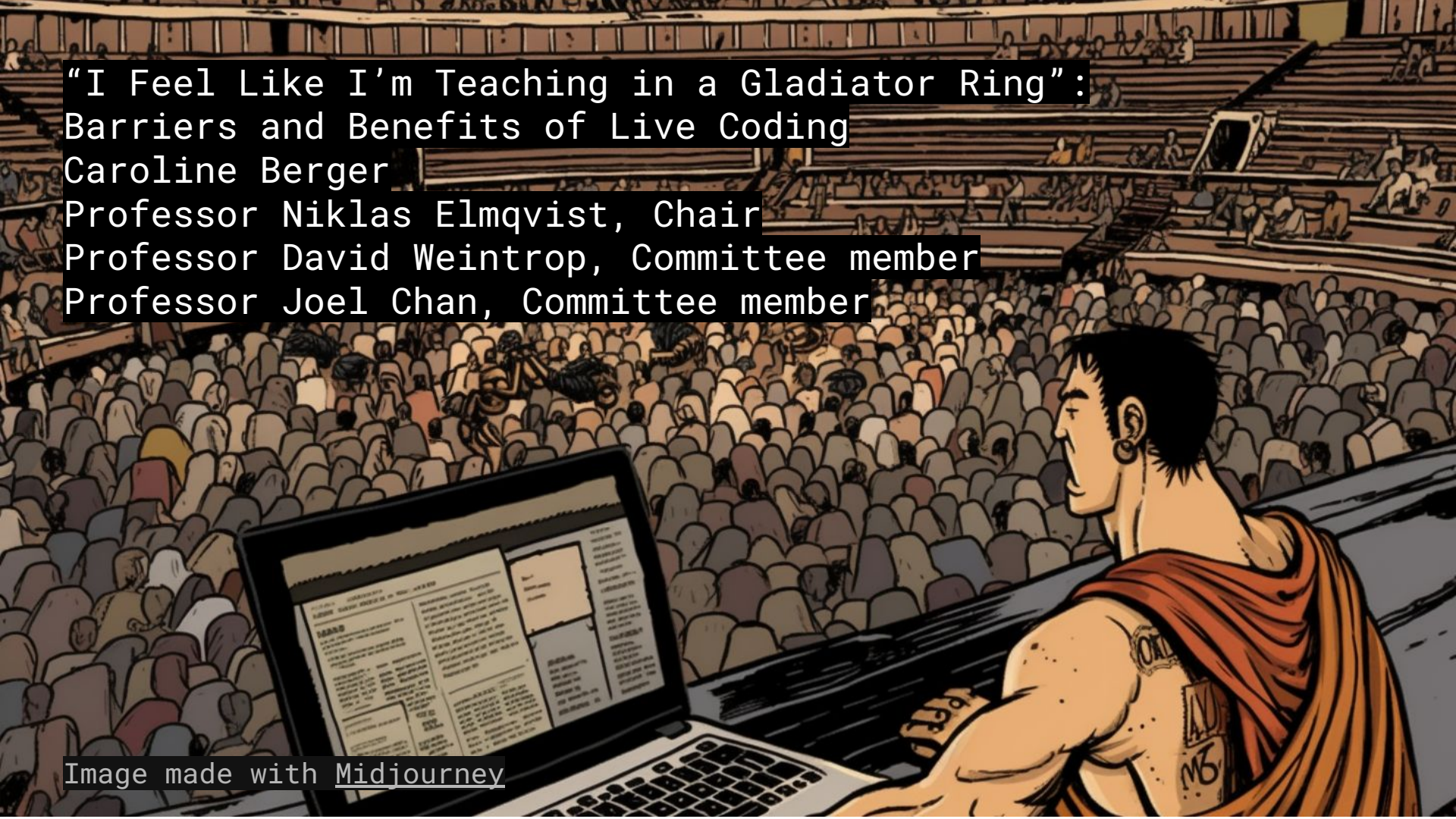
Caroline Berger

Professor Niklas Elmqvist, Chair

Professor David Weintrop, Committee member

Professor Joel Chan, Committee member

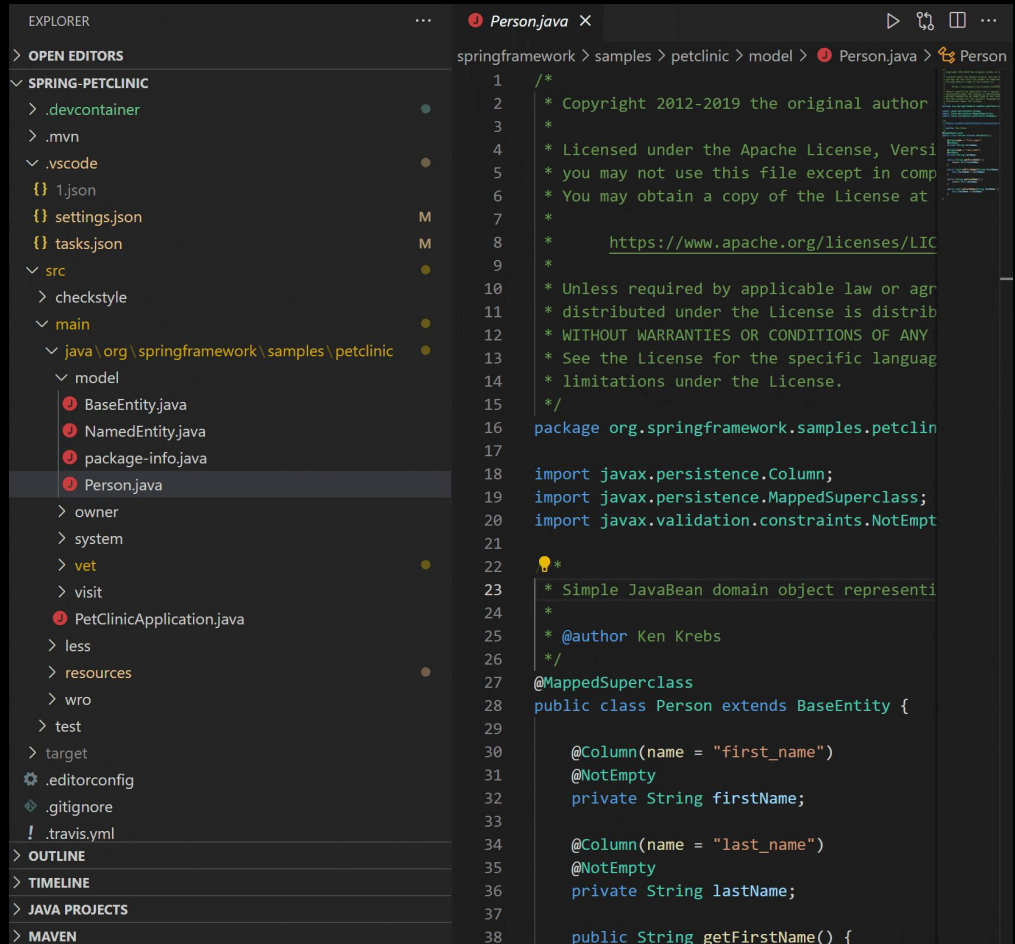
Image made with Midjourney



*“And then I’m teaching [...] in a classroom that feels like a **gladiatorial ring**. 200 seats in a wall up in front of me. And I have to lean back to see the top. And really the only constraint in that classroom is that it’s **terrifying**. It is the most terrifying experience I’ve ever had.” – Participant 08 (Computer Science instructor)*

Live coding is  
“the process of  
writing code live  
on a computer in  
front of students  
during class”

Selvaraj et al. Live coding:  
A review of the  
literature. ITiCSE '21.



```
EXPLORER
> OPEN EDITORS
✓ SPRING-PETCLINIC
  > .devcontainer
  > .mvn
  ✓ .vscode
    {} 1.json
    {} settings.json
    {} tasks.json
  ✓ src
    > checkstyle
    ✓ main
      ✓ java \org \springframework \samples \petclinic
        ✓ model
          ① BaseEntity.java
          ① NamedEntity.java
          ① package-info.java
          ① Person.java
        > owner
        > system
        > vet
        > visit
        ① PetClinicApplication.java
        > less
        > resources
        > wro
        > test
        > target
        ⚙ .editorconfig
        ⚙ .gitignore
        ! .travis.yml
  > OUTLINE
  > TIMELINE
  > JAVA PROJECTS
  > MAVEN

Person.java
springframework > samples > petclinic > model > ① Person.java
1  /*
2  * Copyright 2012-2019 the original author
3  *
4  * Licensed under the Apache License, Versi
5  * you may not use this file except in comp
6  * You may obtain a copy of the License at
7  *
8  * https://www.apache.org/licenses/LIC
9  *
10 * Unless required by applicable law or agr
11 * distributed under the License is distrib
12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY
13 * See the License for the specific languag
14 * limitations under the License.
15 */
16 package org.springframework.samples.petclin
17
18 import javax.persistence.Column;
19 import javax.persistence.MappedSuperclass;
20 import javax.validation.constraints.NotEmpt
21
22 ① *
23 * Simple JavaBean domain object representi
24 *
25 * @author Ken Krebs
26 */
27 @MappedSuperclass
28 public class Person extends BaseEntity {
29
30     @Column(name = "first_name")
31     @NotEmpty
32     private String firstName;
33
34     @Column(name = "last_name")
35     @NotEmpty
36     private String lastName;
37
38     public String getFirstName() {
```



Image source [Wikipedia](#)

# COGNITIVE APPRENTICESHIP

## THEORETICAL FRAMEWORK

# COGNITIVE APPRENTICESHIP


- Modeling: “teacher performs a task so students can observe”;
- Coaching: “teacher observes and facilitates while students perform a task”;
- Scaffolding: “teacher provides supports to help the student perform a task”;
- Articulation: “teacher encourages students to verbalize their knowledge and thinking”;
- Reflection: “teacher enables students to compare their performance with others”; and
- Exploration: “teacher invites students to pose and solve their own problems”.

Collins et al. Cognitive Apprenticeship  
American Educator. 1991.

**THEORETICAL FRAMEWORK**

A

instructor's editor



print("hello world")

instructor's output


hello world

B

Refresh code

What strategies is the instructor using?

Questions from class

Why do you put ' instead of "?  6


Ask question

Ask

C

D

editor



print("hello world")

output

hello world

E

What strategies am I using?

Articulation: B & E

PROTOTYPE

**A**

Class id: SillyPanda64

editor

```
print("hello world")
```

output

```
hello world
```

**B**Students coding  
along with you

75

**C**

Active Students

	Words per minute	Lines of code	Total runs	
Abe	0	1	1	<button>Open code</button>
Marcy	0	0	0	<button>Open code</button>
Luke	0	0	0	<button>Open code</button>
Stacey	2	1	1	<button>Open code</button>
Abe	5	0	0	<button>Open code</button>
Marcy	0	0	0	<button>Open code</button>
Luke	2	0	0	<button>Open code</button>
Stacey	0	1	1	<button>Open code</button>

**D**Code it yourselfBreakoutPoll

Modeling: A  
Coaching: C  
Exploration: D

**PROTOTYPE**



Abe

editor

print("hello world")

output

hello world

Message student

Send emoji

Sally

editor

print("world")

output

world

Message student

Send emoji

John

editor

print("my name is")

output

my name is

Message student

Send emoji

Alyssa

editor

print(

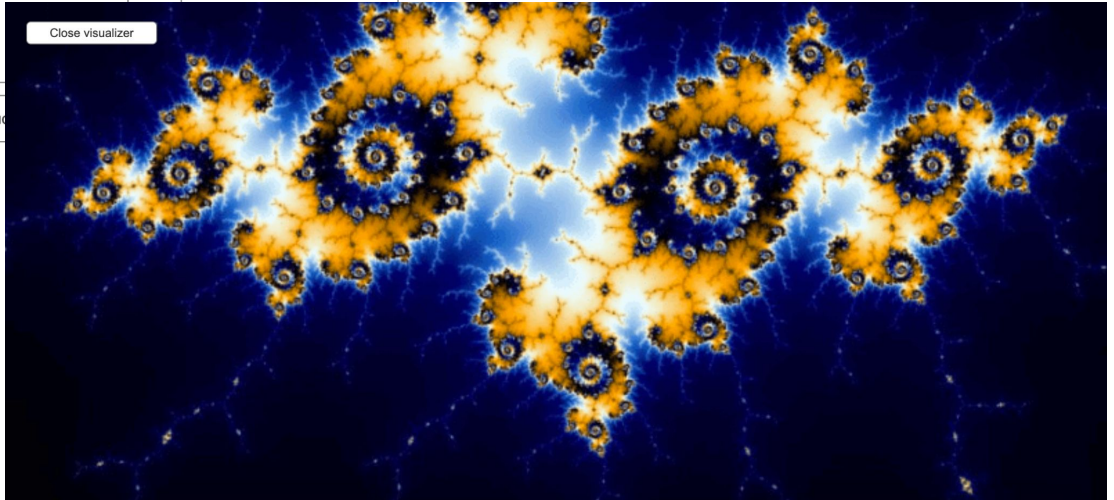
output

Message student

Send emoji

Next page >

Close gallery



Coaching  
Exploration

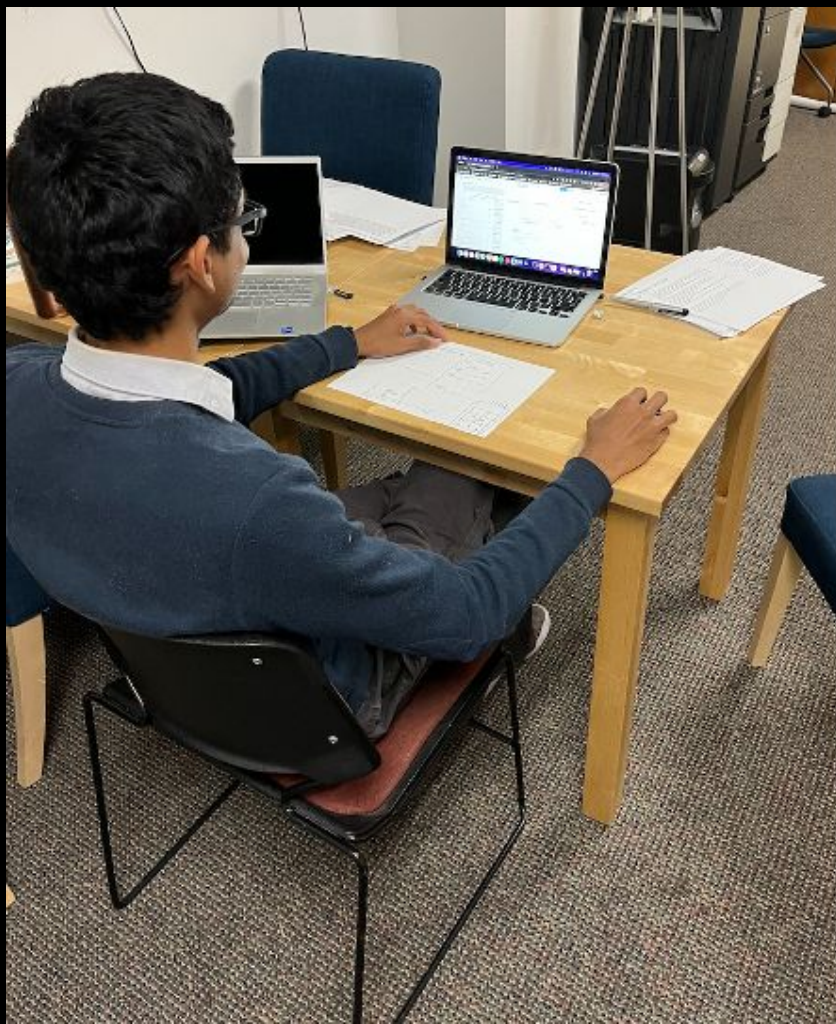
PROTOTYPE



# 1 MIN PAUSE / DANCE PARTY



GIF source [Ukiyo](#)



# Interview & Prototype Feedback

2 instructors  
7 teaching assistants  
6 students

What makes live coding hard?

In an ideal world, how could tools support live coding?

**METHOD**

## *Teaching environment constraints*



Image source [Temple University](#)

**FINDINGS**



# Balancing act while trying to teach

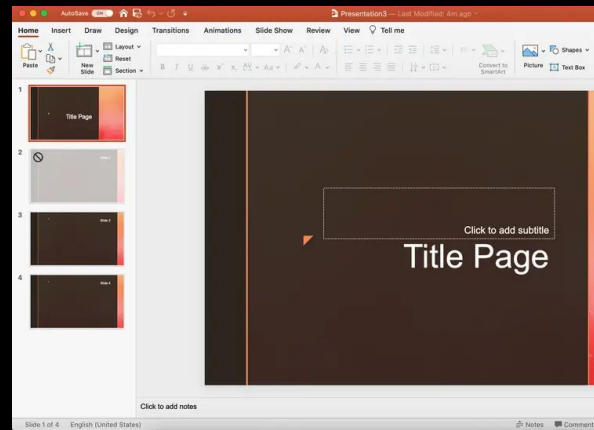
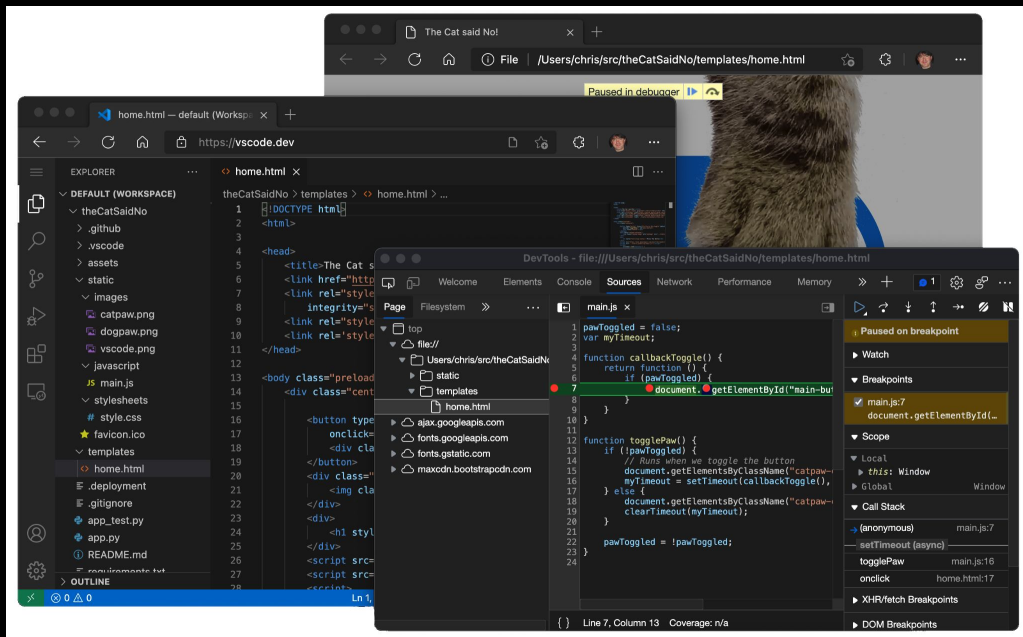
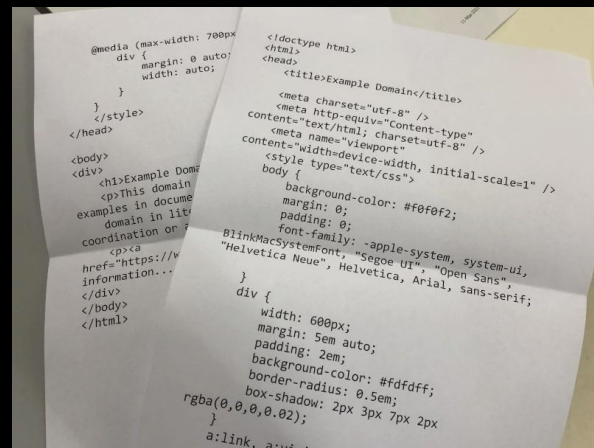


Image sources Visual Studio Code,  
Microsoft, Kevin van der Vleuten



# FINDINGS

## Live coding can be scary

If there's a spelling mistake [when white boarding], or if [a student coding on a whiteboard in front of the class] miss[es] a comma or something, no one cares... You do that on a computer, then **it'll scream at you**, and then there will be the red squiggly. - Participant 02 (Computer Science TA)

```
int main()  
{  
    cout << "Hello, World!" << endl  
    return 0;  
}
```

```
def fizbuzz (int num)  
    if (num % 2 == 0)  
        printf("fiz")  
    else  
        printf("buzz")
```

Image sources  
David Neely, Weber  
State University

**FINDINGS**

## Treasure hunt for errors



Image source  
Jitendra Zaa

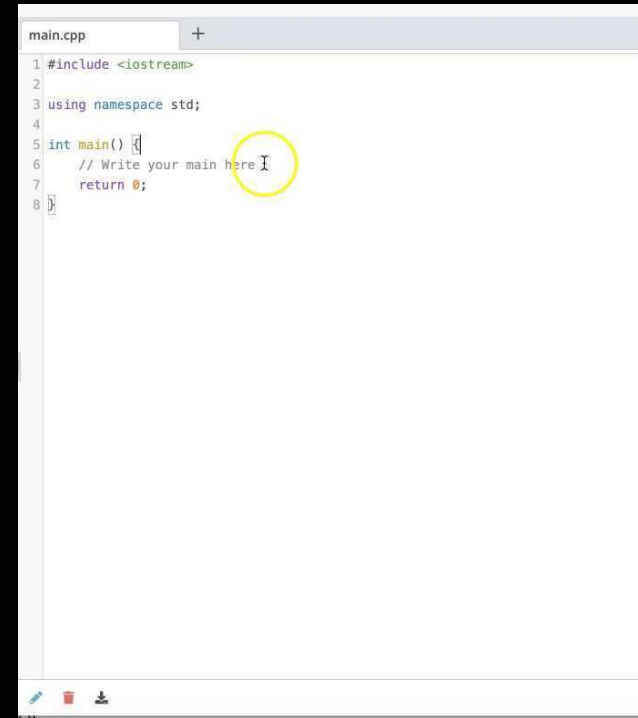
Oh, the students love to find the problems as I code them. I kind of built this open atmosphere with a lot of debugging being a big focus of the course and encourage the students as I'm typing and we're seeing what's going on if they see something that's wrong, a lot of times they'll say aloud. Otherwise, it's the process... I built in errors in the code to see a lot of the problems that they may stumble across as they code. - Participant 11 (Computer Science Instructor)

**FINDINGS**



## Scaffolds: use wisely

It's usually better when they start from zero with some example. I've seen some professors or where if they're finishing some project and half of it is already done. And then they start from there, the professor already has the half of it in their mind, but this might be the first time ever a student is seeing it. And yeah, I've had that experience and being completely lost. And then you have to go back afterwards and see the half that they started with, understand that first, and then mentally replay the lecture and then it makes sense but it's a little bit of work. - Participant 02 (Computer Science Teaching Assistant)



```
main.cpp +
1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     // Write your main here I
7     return 0;
8 }
```

Image source [LBD Community College](#)

*Computers optional*

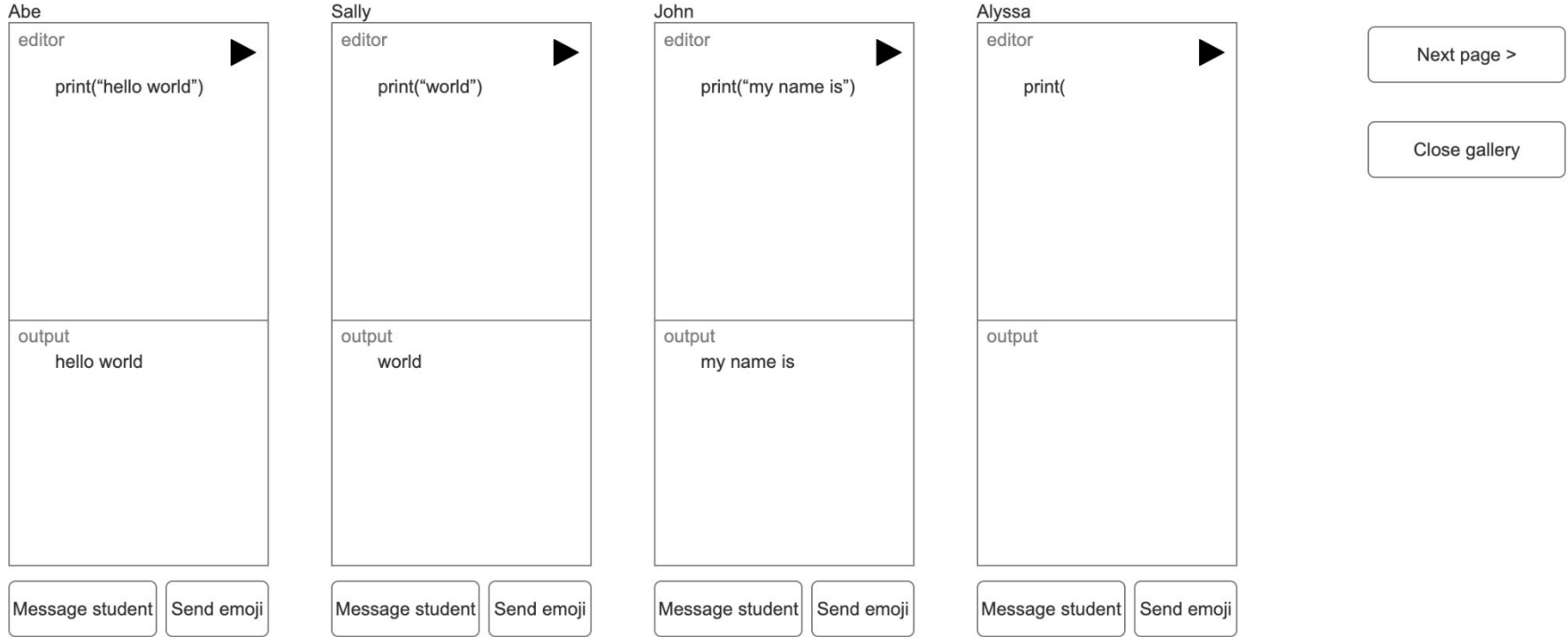


Image sources [ZDNET](#), [UCSB](#)



**DESIGN GUIDELINES**

## Peeking into student editors



# DESIGN GUIDELINES

DESIGN GUIDELINE	COGNITIVE APPRENTICESHIP STAGE	TYPE OF LIVE CODING
Personal computers optional	Modeling	Instructor-led
Directing attention	Modeling	Instructor-led
Many keyboards, one digital space	Coaching, Scaffolding	Student-instructor collaborative
Errors as signals of student progress	Coaching, Scaffolding	Student-led
Peeking into student editors	Coaching, Scaffolding	Student-led

**DESIGN GUIDELINES**

*Teaching environment*

*Support from other teachers*



*Mindset shift*

Caroline Berger  
cberger2@umd.edu

**IMPLICATIONS**

Backup slides



# Related work

- Improv helps instructors to flip between their editor and slides (Chen & Guo, Improv: Teaching programming at scale via live coding, 2019)
- VizProg displays students' progress towards a solution (Zhang et al., Vizprog: Identifying misunderstandings by visualizing students' coding progress, 2023)
- Overcode analyzes student submissions (Glassman et al., Overcode: Visualizing variation in student solutions to programming problems at scale, 2015)
- Codeopticon has a gallery view for tutoring purposes (Guo, Codeopticon: Real-time, one-to-many human tutoring for computer programming, 2015)

# Future work

- Validate proposed design guidelines
- Research live coding tools in informal learning environments like hobbyist communities
- Explore analog systems of engagement like clickers

# Limitations

- Work might not translate to other education settings and geographic contexts
- Data completeness - Half of P04's interview was lost due to technical issues

# Participants

2 instructors  
7 teaching  
assistants  
6 students

13 men  
1 woman  
1 non-binary person

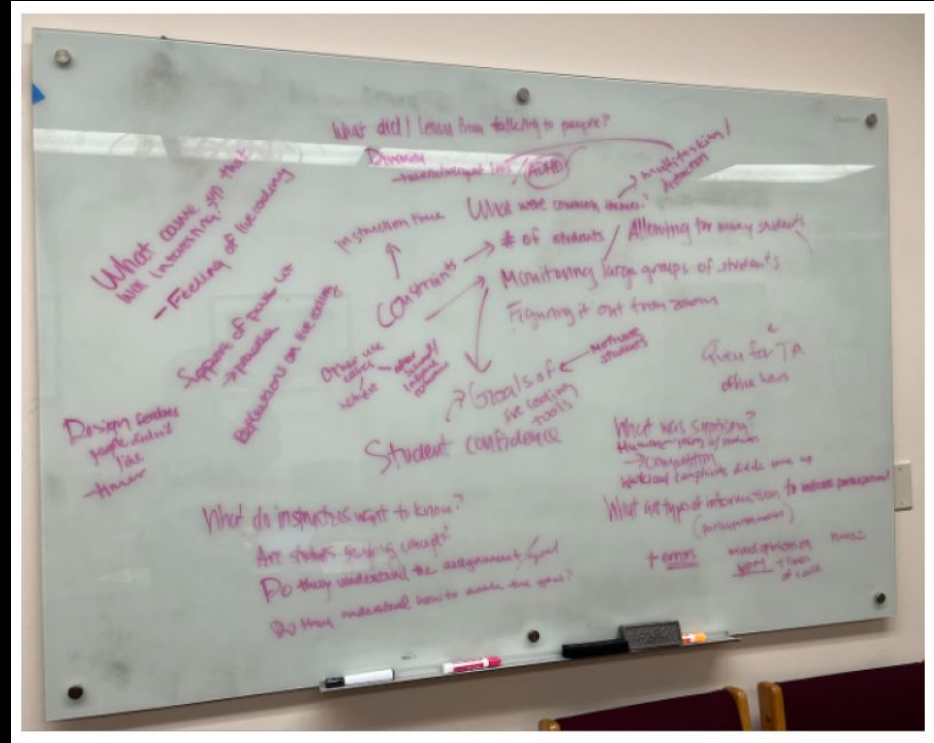
#	ROLE	SETTING	AGE	GENDER	EDUCATION
P01	Teaching Assistant	Online	26-35	Man	B.Sc. CS; M.Sc. CS; Ph.D. Info. Sci. student
P02	Teaching Assistant	In-Person	18-25	Non-Binary	B.Sc. CS student
P03	Student	In-Person	18-25	Man	B.Sc. CS, Math student
P04	Student	In-Person	18-25	Man	B.A. Psychology; M.Sc. HCI student
P05	Student	In-Person	26-35	Man	B.Eng. CS, Eng.; M.Sc. HCI student
P06	Teaching Assistant	In-Person	18-25	Man	B.Sc. CS student
P07	Teaching Assistant	In-Person	26-35	Man	BSc. CS; MSc. CS; Ph.D. HCI student
P08	Instructor	In-Person	36-45	Man	M.Sc. Info. Sys.
P09	Student	In-Person	18-25	Man	B.Sc. CS student
P10	Student	In-Person	18-25	Man	B.Sc. CS; M.Sc. HCI student
P11	Instructor	In-Person	45+	Man	B.Sc. Chemistry; Ph.D Info. Sci. student
P12	Teaching Assistant	In-Person	18-25	Man	B.Sc. CS student
P13	Student	In-Person	18-25	Man	B.Sc. CS, Robotics student
P14	Teaching Assistant	Online	18-25	Woman	B.Sc. Math, CS; Ph.D. CS, CS Ed. student
P15	Teaching Assistant	In-Person	26-35	Man	B.A. Design; Ph.D. Info. Sci. student

## Reflexive thematic analysis

### Significant statements

## Reflexive thematic analysis

### Significant statements



# Coding Process

CODE	NOTE	QUOTE
Description	Live coding as a performance	<i>"It's like live performance. It's really hard to practice it enough that you know that it's going to work, but also have that kind of ability to take student suggestions and potentially go in a direction that you haven't tested and might not work out." (P08)</i>
Barrier	Fear of messing up	<i>"Part of it is the pressure of just being in front of an audience. And you sort of, I mean, naturally you don't want to mess up. And so thinking of that gives you some sort of, I guess, anxiety, but I guess for me over time at first I was definitely like nervous since it was my first time doing anything like that. But I think in my experience, I got less nervous and much more comfortable. But yeah, I think the main thing is definitely just the anxiety of messing up so badly for students." (P06)</i>
Benefit	On-the-fly nature of live coding	<i>"I'm doing some example, then it's easier to change stuff on the fly and then surprise students." (P02)</i>
Design opportunity	Gallery camera view	<i>"Okay, these students got it these students didn't. I would love to have a second screen that had, you know, the small kind of security camera view where I had every student desktop and be able to see that they're all on their own." (P11)</i>