NERDS: A Non-Invasive Environment for Remote Developer Studies

Joe Lewis
jlewis23@umd.edu
University of Maryland, College Park

Kelsey Fulton
Kelsey.fulton@mines.edu
Colorado School of Mines

17th Cyber Security Experimentation and Test Workshop
Tuesday, August 13th 2024
Philadelphia, PA, USA

Outline

- Problem motivation and prior work
- Overview of our system: NERDS
- Case study of using NERDS in two developer studies
- How you can use NERDS

Problem motivation and Prior Work

We study Developers

- Understanding developers is key to understanding software vulnerabilities
- Studying developers is challenging, as exact environments are hard to replicate

Studying developers remotely

Developer studies can be done remotely with a few key tradeoffs:

Advantages

- Larger recruitment pool
- Less time consuming
- Less expensive
- Data collection can be fully automated

Disadvantages

- Difficult to fully replicate environment
- Developing online platforms from scratch is expensive and time consuming
- No cognitive walkthroughs

Prior Work

- **Developer Observatory** (Stransky et al., 2017): platform for remote Python studies
  - Forms the basis for our work
  - Remote studies were easier and faster than in-person studies
  - Data collection limited by nature of remote study
- **OLab** (Huaman et al., 2022): Provides full remote desktop interface for remote developer studies

Limitations of Prior Work

- Developer Observatory
    - Limited scope – only works with Python studies
    - Limited to 20-50 concurrent users
    - Users complained about long-wait times to start study
- OLab
    - Heavier system than what we needed
    - Not open-sourced
    - Appears to only be tested with loads < 25 participants

System Overview

Requirements

- **Participant Experience**
- **Experimental Design**
- **Technical Requirements**

Requirements (cont.)

Participant Experience

- **Non-invasive**: system should place minimal requirements on the user’s system to participate
- **Flexibility**: Ability to leave and return the study at any point
- **Skip-revisit:** Skip and revisit tasks participants are struggling with

Requirements (cont.)

Experimental Design

- **Randomization**: randomized condition assignment and task order
- **Control/Real-world**: Researchers should be able to control environment while replicating real-world environments
- **Data collection**: Data should be gathered as much as possible without being over intrusive
- **Ethics**: Data should be stored securely with a pseudonymous identifier

Requirements (cont.)

Technical Requirements

- **Isolation**: full isolation between participants, including access and DoS
- **Scalability**: System should handle many concurrent participants at once
- **Adaptability**: System should be adaptable to other study designs

Our system: NERDS

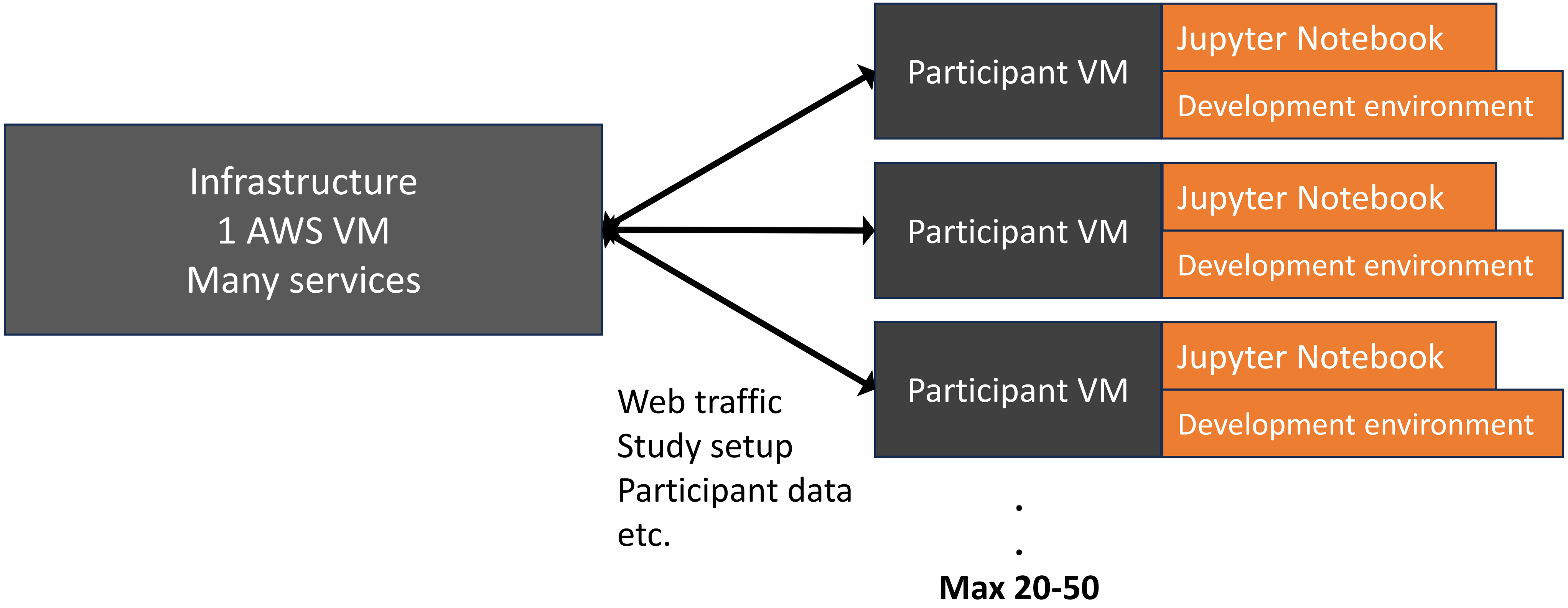
What is it?

An environment for hosting browser-based remote developer studies.

Heavily-modified version of Developer Observatory from Stransky et al.

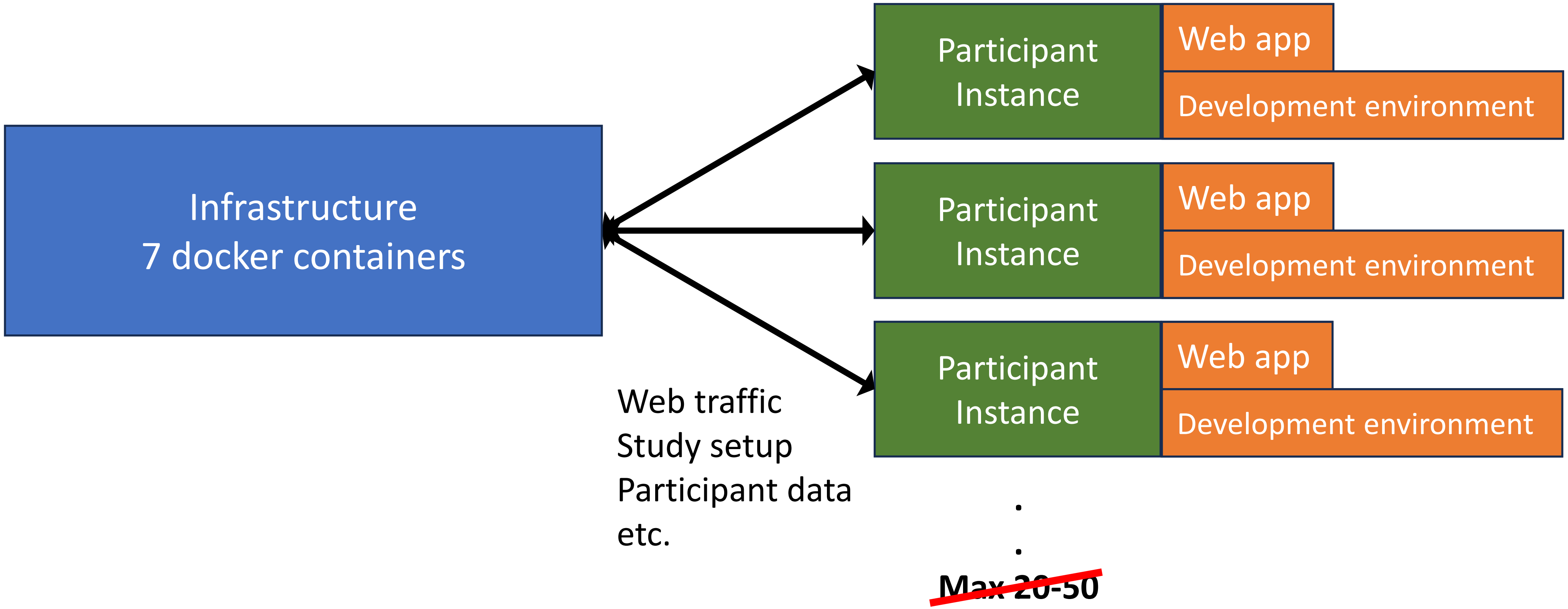
Technical Overview: Developer Observatory

- Infrastructure handles creating VMs and collecting data
- Participant VM spun up for each participant



Technical Overview: NERDS

- Major overhaul: change to Docker containers
- Improved **scalability** and **adaptability**, while maintaining other reqs.



Requirements - Review

Participant Experience	**Experimental**	**Technical**
• Non-invasive	• Randomization	• Isolation
• Flexibility	• Control	• Scalability
• Skip-revisit	• Real-world	• Adaptability
	• Data collection	
	• Ethics	

NERDS Features

- Totally browser-based system: only requirement is an up-to-date web browser (**non-invasive**)
- Participant instances are persistent, cookies used to return user to their instance (**flexibility**)
- Development environment is pre-installed and the same across all participants (**control**)
- System assigns conditions randomly to users and pre-loads instances with required files (**randomization**)
- System automatically collects data in *db* with a pseudonymous identifier (**ethics**)
- Containers are isolated from each other and the infrastructure (**isolation**)

Case studies

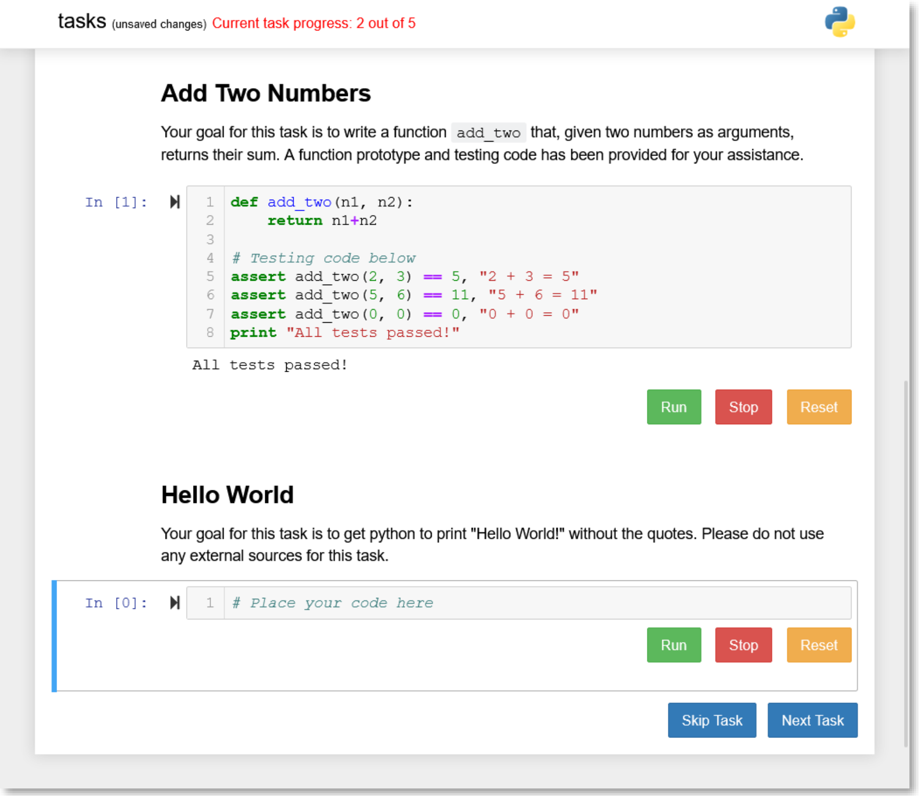
Case Study: Python study

Used in *Write, Read, or Fix? Exploring Alternative Methods for Secure Development Studies* at SOUPS 2024

- Remote study in Python
- Task-based
- Read, write, and fix conditions with two different libraries – total of 6 conditions
- Only write and fix should be able to edit and run code

Participant Instance – Technical Detail

- Modified **participant instance** slightly from Stransky et al.
    - arbitrary number of tasks and sub-tasks
- 141 participants over 1 year period



Add Two Numbers

Your goal for this task is to write a function `add_two` that, given two numbers as arguments, returns their sum. A function prototype and testing code has been provided for your assistance.

In [1]:
```
def add_two(n1, n2):
    return n1+n2

# Testing code below
assert add_two(2, 3) == 5, "2 + 3 = 5"
assert add_two(5, 6) == 11, "5 + 6 = 11"
assert add_two(0, 0) == 0, "0 + 0 = 0"
print "All tests passed!"
```

```
All tests passed!
```

Hello World

Your goal for this task is to get python to print "Hello World!" without the quotes. Please do not use any external sources for this task.

In [0]:
```
# Place your code here
```

Case Study: Python study – Lessons Learned

Things that worked well

- Conducting a remote study was far easier than an in-person study
- Maintained validity
- Large sample size

Things that didn’t work well

- Some key insights were missed
- Participants can work around the restrictions of the system

Case Study: C study

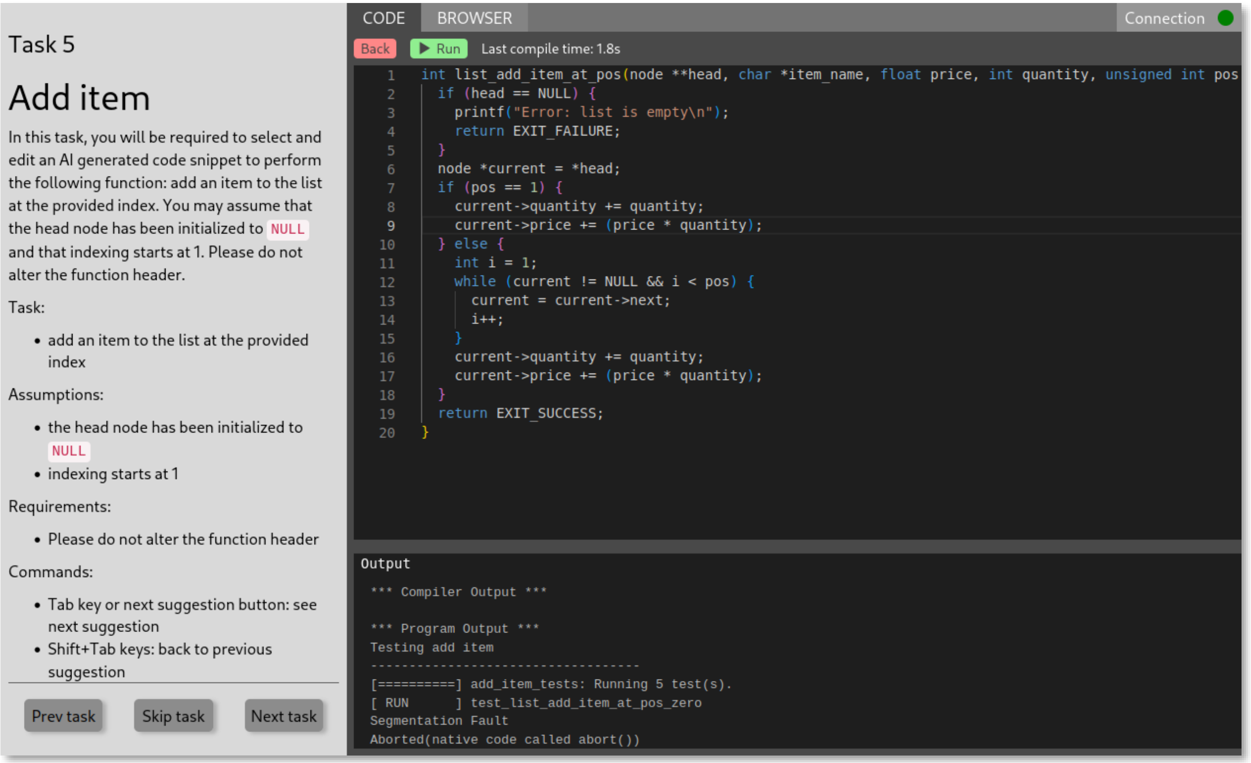
- Requirements – same as NERDS system plus...
  - Coding in C instead of Python
  - Collect participant search history
- Redesigned participant instance to support C language and a remote browser
- Currently on-going

Requirements - Review

Participant Experience	**Experimental**	**Technical**
• Non-invasive	• Randomization	• Isolation
• Flexibility	• Control	• Scalability
• Skip-revisit	• Real-world	• Adaptability
	• Data collection	
	• Ethics	

Participant Instance – Technical Detail

- New C development environment
  - Based on Visual studio code (**real-world**)
- New WebAssembly C runtime
  - No code runs on participant instance (**isolation**, **scalability**)
- New built-in browser
  - Runs in container, displayed through noVNC (**non-invasive, flexibility**)



Task 5

Add item

In this task, you will be required to select and edit an AI generated code snippet to perform the following function: add an item to the list at the provided index. You may assume that the head node has been initialized to `NULL` and that indexing starts at 1. Please do not alter the function header.

Task:

- add an item to the list at the provided index

Assumptions:

- the head node has been initialized to `NULL`
- indexing starts at 1

Requirements:

- Please do not alter the function header

Commands:

- Tab key or next suggestion button: see next suggestion
- Shift+Tab keys: back to previous suggestion

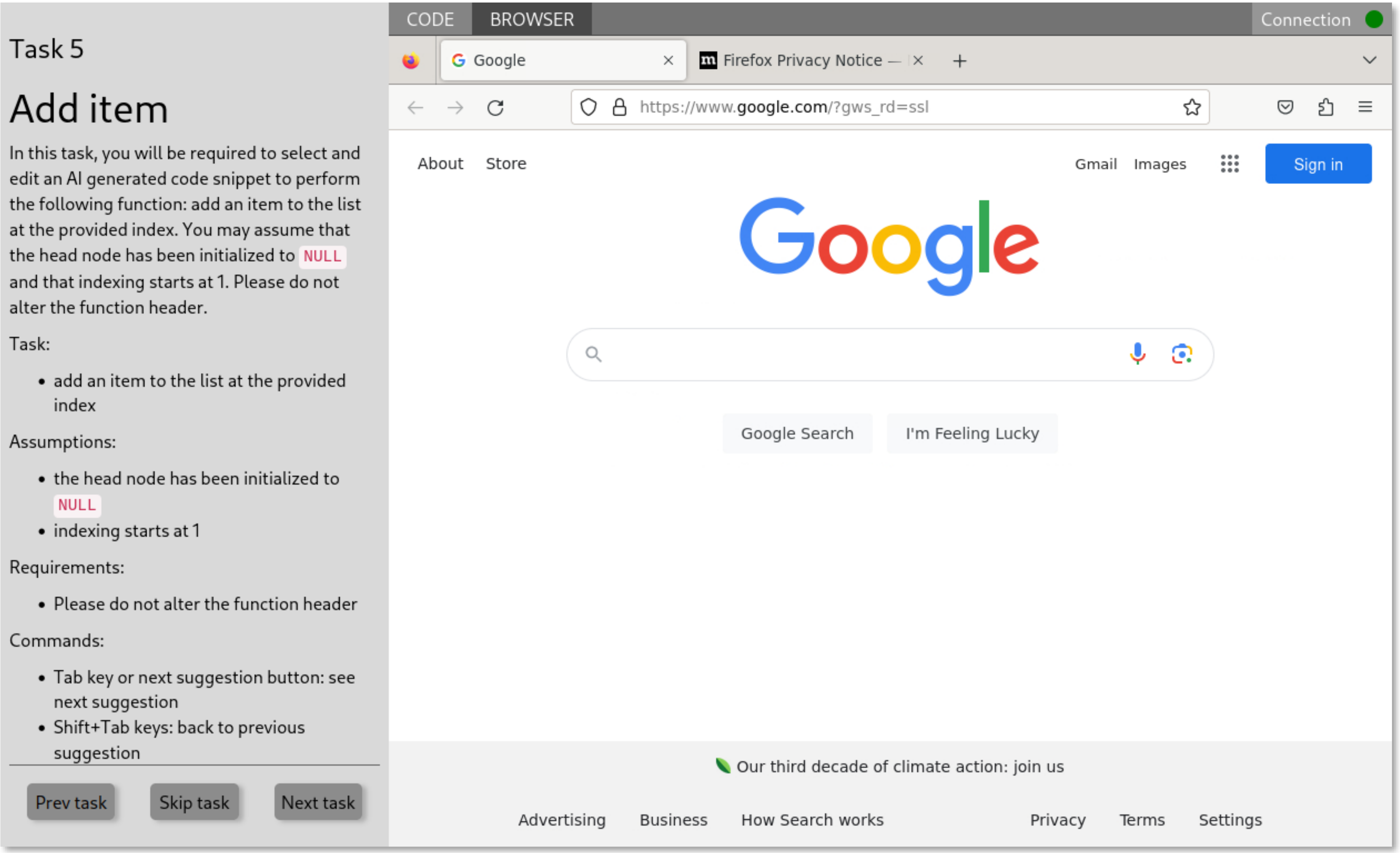
```c
int list_add_item_at_pos(node **head, char *item_name, float price, int quantity, unsigned int pos
  if (head == NULL) {
    printf("Error: list is empty\n");
    return EXIT_FAILURE;
  }
  node *current = *head;
  if (pos == 1) {
    current->quantity += quantity;
    current->price += (price * quantity);
  } else {
    int i = 1;
    while (current != NULL && i < pos) {
      current = current->next;
      i++;
    }
    current->quantity += quantity;
    current->price += (price * quantity);
  }
  return EXIT_SUCCESS;
}
```

Output

```
 *** Compiler Output ***

 *** Program Output ***
 Testing add item
 ----------------------------------
 [==========] add_item_tests: Running 5 test(s).
 [ RUN      ] test_list_add_item_at_pos_zero
 Segmentation Fault
 Aborted(native code called abort())
```

Participant Interface – Remote Browser



Task 5

Add item

In this task, you will be required to select and edit an AI generated code snippet to perform the following function: add an item to the list at the provided index. You may assume that the head node has been initialized to `NULL` and that indexing starts at 1. Please do not alter the function header.

Task:

- add an item to the list at the provided index

Assumptions:

- the head node has been initialized to `NULL`
- indexing starts at 1

Requirements:

- Please do not alter the function header

Commands:

- Tab key or next suggestion button: see next suggestion
- Shift+Tab keys: back to previous suggestion

Case Study: C Study – Lessons Learned

Things that worked well

- Switching participant instances was easy – though development cost was high

Things that didn’t work well

- Embedded browser may not be convenient to use
- Be careful with production systems ☺

How YOU can use NERDS

- All code is released open source on https://joelewiss.github.io/nerds
- Both participant instances are released for Python and C
- Documentation on website
- Developing custom instances
- We would **love** to hear from you if you’re interested in using our system

Key Takeaways

- NERDS is a **flexible, open source** system for hosting remote developer studies
- Tested in production with multiple types of studies
- Provides observable advantages to in-person studies **with key tradeoffs**
- Available for download and use

https://joelewiss.github.io/nerds

Joe Lewis
jlewis23@umd.edu
University of Maryland, College Park
College Park, MD, USA

Kelsey Fulton
kelsey.fulton@mines.edu
Colorado School of Mines
Golden, CO, USA