HACKADEMIA

<discover>
<learn>
<defend>

The Problem

Cybersecurity exists at the intersection of many different computing fields: programming, operating systems, networking, etc. Complex cybersecurity concepts, like buffer overflows, are more difficult to learn because a wider breadth of knowledge, in all levels of program abstraction, is required.

Modules

Hackademia V1.0 focuses on three concepts:
> Stack Frame (x86-64)
> Buffer Overflow (IA32)
> Command Injection (IA32)

Challenges

> Initial Project Scope
> UI Design / Custom CSS
> Animation Feature
> Content Complexity
> Database Setup

Our Solution

Hackademia is an interactive cybersecurity educational tool used to animate stack-based vulnerabilities and exploits. This bridges the gap from high-level C code to low-level assembly- visualizing how a program is run within the CPU, and how attackers can take advantage of its flaws.

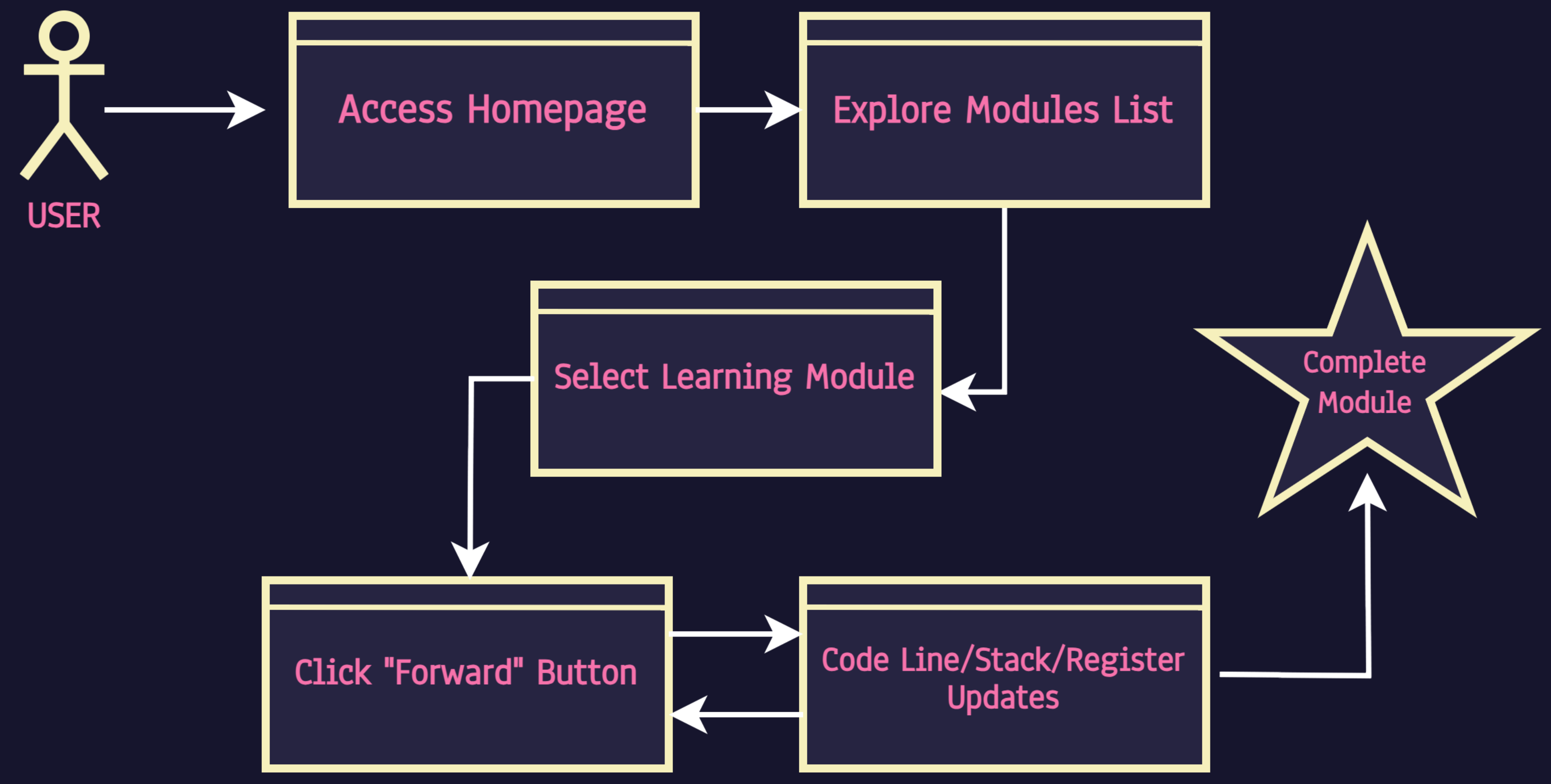
Tech Stack

REACT JS

Node.js

PostgreSQL

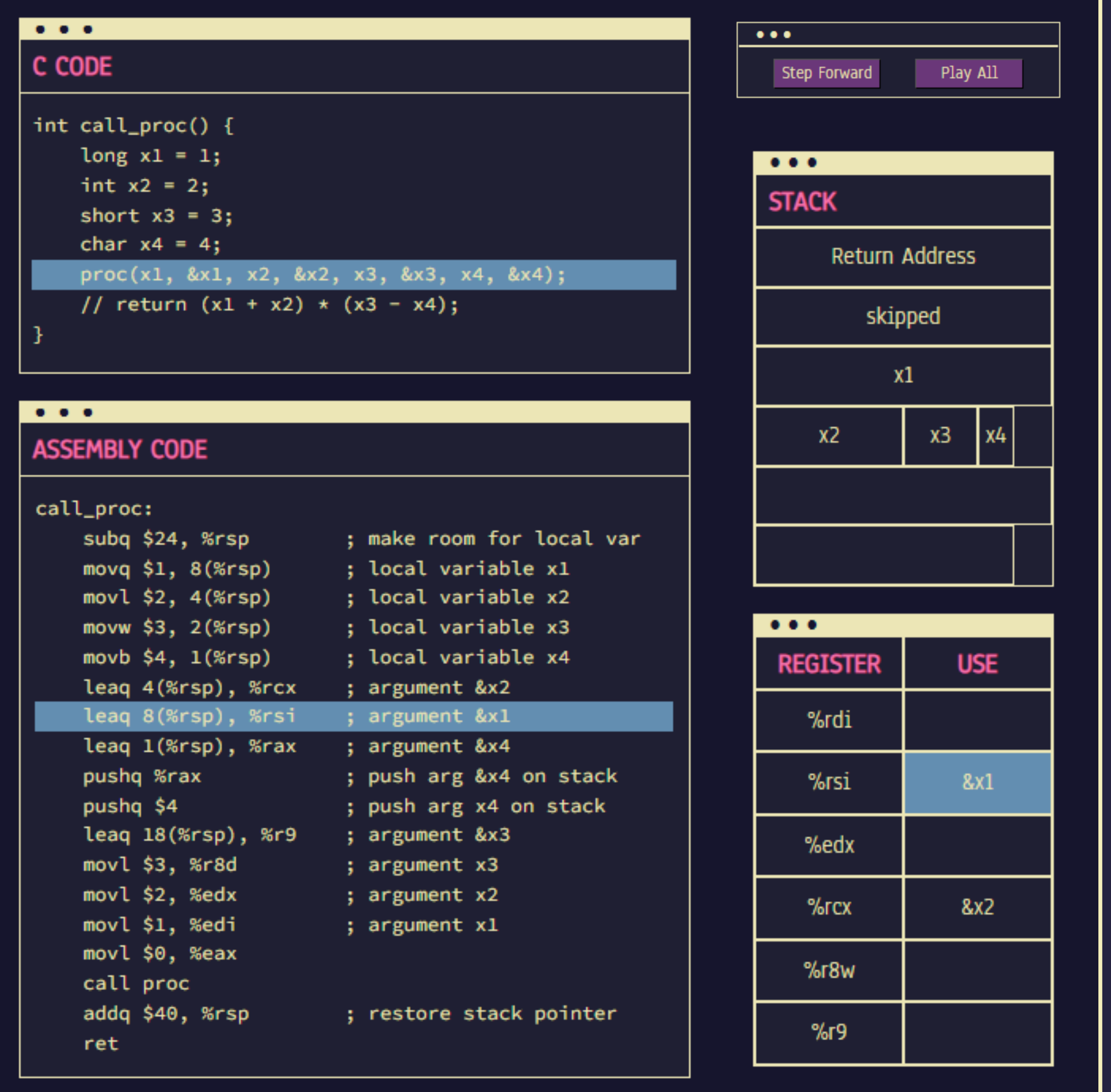
System Diagram



Animation Features

The animation for each module includes:
> C code
> x86-64 assembly
> Stack frame
> Registers
> Exploit demo (command injection module only)

In a debugging-style platform, students step through each assembly instruction of a program. The stack and registers are updated accordingly, giving direct insight as to what is happening under the hood.



C CODE

```
int call_proc() {
    long x1 = 1;
    int x2 = 2;
    short x3 = 3;
    char x4 = 4;
    proc(x1, &x1, x2, &x2, x3, &x3, x4, &x4);
    // return (x1 + x2) * (x3 - x4);
}
```

ASSEMBLY CODE

```
call_proc:
    subq $24, %rsp           ; make room for local var
    movq $1, 8(%rsp)         ; local variable x1
    movl $2, 4(%rsp)         ; local variable x2
    movw $3, 2(%rsp)         ; local variable x3
    movb $4, 1(%rsp)         ; local variable x4
    leaq 4(%rsp), %rcx       ; argument &x2
    leaq 8(%rsp), %rsi       ; argument &x1
    leaq 1(%rsp), %rax       ; argument &x4
    pushq %rax               ; push arg &x4 on stack
    pushq $4                 ; push arg x4 on stack
    leaq 18(%rsp), %r9       ; argument &x3
    movl $3, %r8d            ; argument x3
    movl $2, %edx            ; argument x2
    movl $1, %edi            ; argument x1
    movl $0, %eax
    call proc
    addq $40, %rsp           ; restore stack pointer
    ret
```

STACK

Return Address			
skipped			
x1			
x2	x3	x4	

REGISTER	USE
%rdi	
%rsi	&x1
%edx	
%rcx	&x2
%r8w	
%r9	

Next Phase: Hackademia V2.0

The next version of Hackademia will allow students to input C code and run the animation using the program provided. This allows students to see more clearly how their program is run and if it is vulnerable to buffer overflow attacks.

The steps to required to design and implement V2.0 are:
> Integrate user input capabilities
> Embed in-application x86-64 compilation
> Update animation components in real-time given user input

Molly Vongsakhamphouy
Computer Science
Backend Developer

Emily Bever
Computer Science
Full-stack Developer

Robin Heft
Computer Science
Cyber Content Developer

Ryan Connolly
Computer Science
Frontend Developer

Advisor:
Giovani Abuaitah