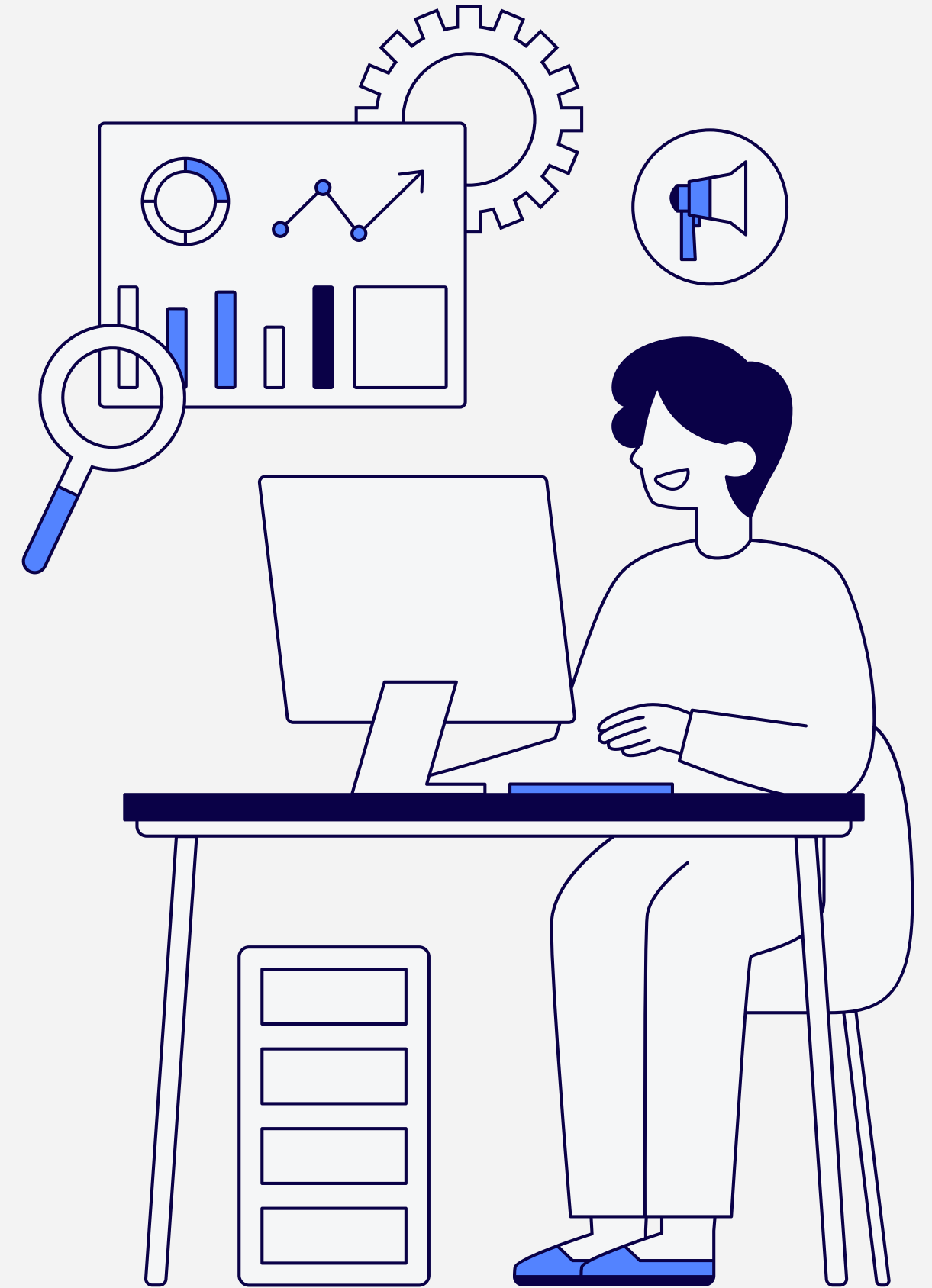


SECURE DATA ANAYZER

- **TEAM NAME:** HACKHIVE
- **TEAM MEMBERS:**
 - SWETHA T -BTECH IT 2nd yr
 - BRINDA P -BTECH IT 2nd yr



PROBLEM STATEMENT

Developers often write insecure code without realizing vulnerabilities like SQL Injection, XSS, or Hardcoded Credentials.

Traditional security tools **only detect issues but don't suggest fixes** or educate developers. This leads to security breaches, data leaks, and compliance failures.

Our Goal:

Create an **AI-powered security assistant** that detects, explains, and fixes vulnerabilities inside the developer's workflow.

PROPOSED SOLUTION

"AI-Powered Security Assistant for Developers"

- => AI-driven vulnerability detection → Scans code in real time.
- => Instant Fix Recommendations → AI suggests secure code alternatives.
- => Built-in OWASP Compliance → Maps issues to OWASP Top 10 vulnerabilities.
- => Developer-Friendly Integration → Works as a VS Code extension or web-based tool.

What Makes It Unique?

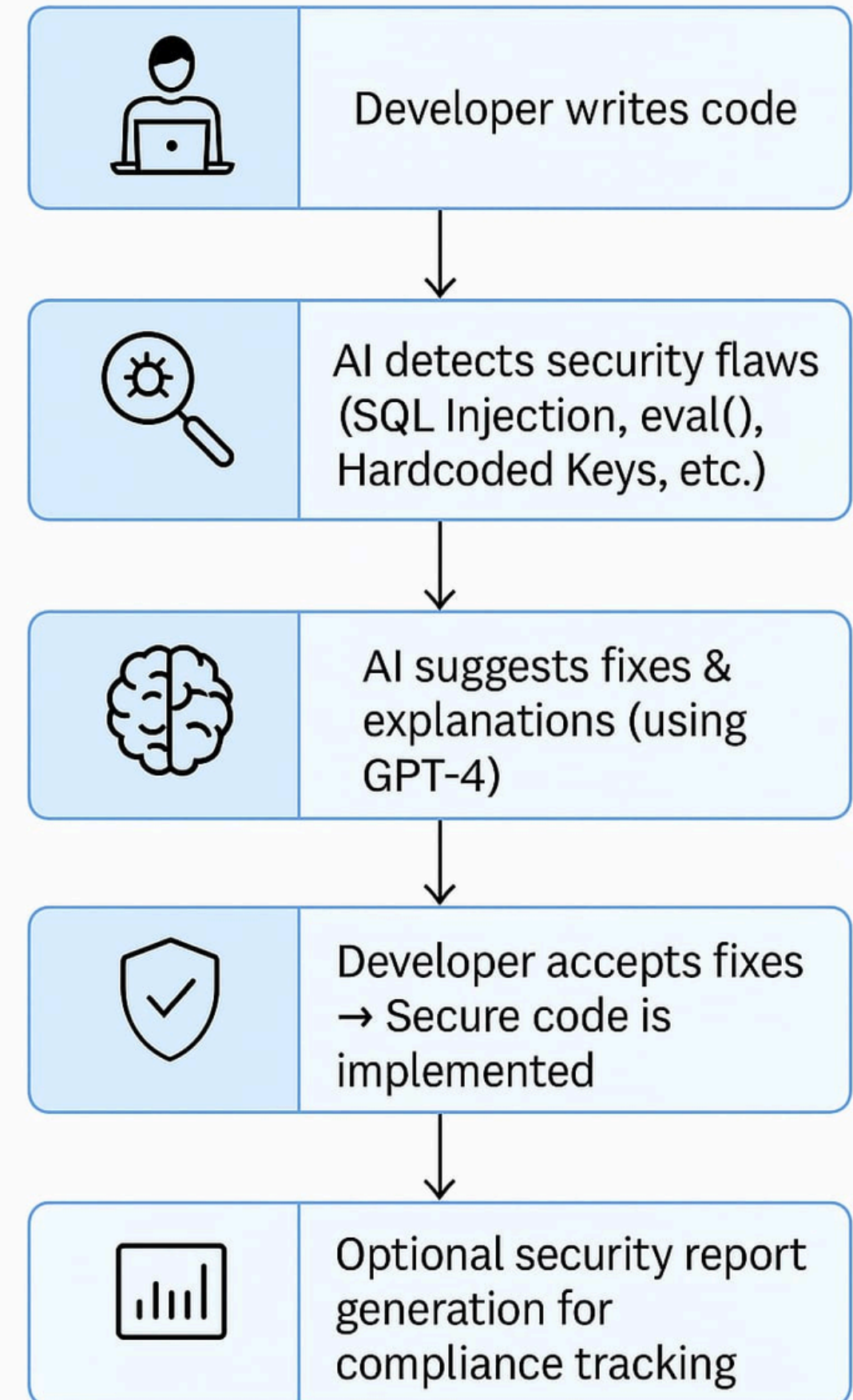
Unlike traditional security scanners, our solution provides context-aware fixes using GPT-4. Developers don't need deep security knowledge—AI educates and improves code on the go.

FLOW DIAGRAM

"How Our AI Security Assistant Works"

- 🔧 Step 1: Developer writes code → AI scans for vulnerabilities.
- 🔍 Step 2: AI detects security flaws (SQL Injection, eval(), Hardcoded Keys, etc.).
- 🧠 Step 3: AI suggests fixes & explanations (using GPT-4).
- ✅ Step 4: Developer accepts fixes → Secure code is implemented.
- 📊 Step 5: Optional security report generation for compliance tracking.

How Our AI Security Assistant Works



TECH STACK AND IMPLEMENTATION

Frontend:

- HTML5, CSS3 – Basic form UI for uploading or entering code
- JavaScript (Fetch API) – For async code analysis requests
- Optional: Bootstrap (for UI styling)

Backend:

- Python 3.x
- Flask – Web framework to handle routes and requests
- Cohere Python SDK – For interacting with Cohere's AI model

AI/LLM:

- Cohere API – Using command-r-plus or command-r model via co.chat() API (not generate())

PROMPT TEMPLATE AND STRUCTURE



Template Structure:

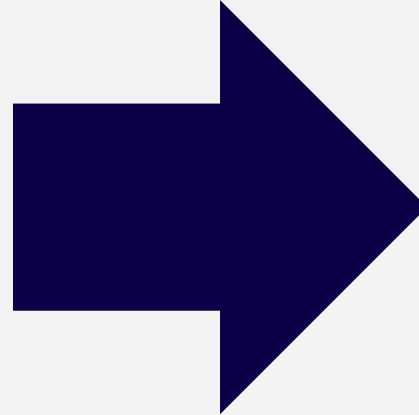
Analyze the following code and identify potential security vulnerabilities.
Provide a detailed explanation for each issue and suggest a secure alternative.
Ensure fixes follow OWASP security guidelines.



Why This Works?

Uses structured instructions to ensure AI provides detailed analysis.
Prevents generic responses by enforcing security compliance.
Optimized for accuracy with specific prompts tailored to different languages.


SAMPLE OUTPUT



Cohere Secure Code Analyzer

Paste your code here:

```
n=int(input())  
print(n)
```

Explanation level: Short 

Analyze

🔧 Suggested Fixes:

Here is a summary of the key issues identified in the code:

- ****Input Validation****: The code directly converts user input to an integer using `int(input())` without any validation. This can lead to potential code injection attacks if an invalid input is provided. It is good practice to validate user input to ensure it falls within the expected range or format.
- ****Lack of Input Sanitization****: The code does not sanitize the user input, which can potentially allow malicious input to exploit security vulnerabilities. It is important to sanitize input to prevent issues like SQL injection, cross-site scripting (XSS), or command injection.
- ****Lack of Error Handling****: The code lacks proper error handling. If the user provides non-integer input, the program may crash or produce unexpected results. It is recommended to use try-except blocks to handle potential exceptions and provide appropriate error messages.

CONCLUSION & IMPACT

"The Future of AI-Driven Security"

- ◆ Reduces security risks by providing instant fixes.
- ◆ Educates developers while they code.
- ◆ Seamless integration with developer tools (VS Code, Web).
- ◆ Future Scope: Expanding AI analysis to more languages (Java, C, JavaScript, etc.)

 This project ensures security is a priority from day one!