



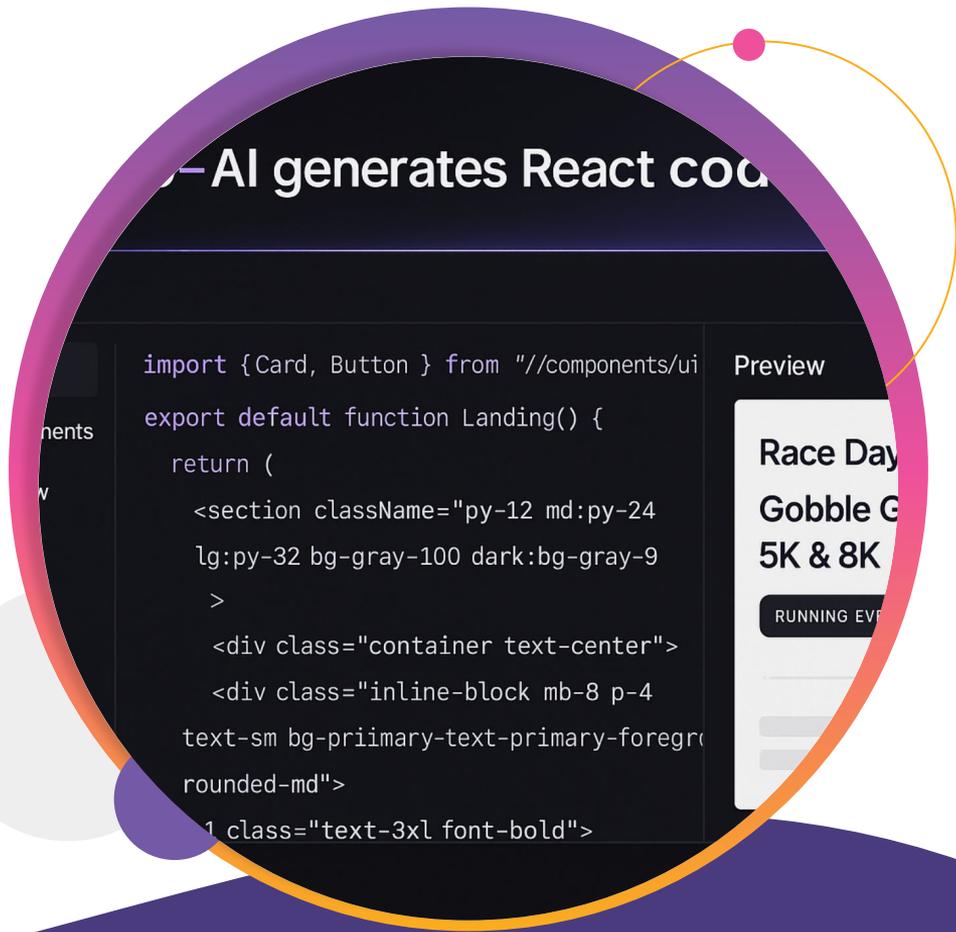
# Vibe Coding & Security

 RunSignup



# Agenda

- Quick recap on vibe coding
- When does security become a concern?
- Secure RunSignup Access
- Q/A





# What is Vibe Coding?

Coding by Intent, Not Syntax: A development style where the human provides high-level, natural language instructions ("vibes"), and an AI (LLM) writes the actual code.

## The Workflow

- Input: "Make the button blue and bouncy when clicked."
- Process: AI generates HTML/CSS/JS instantly.
- Loop: User tests it -> "The bounce is too slow" -> AI rewrites code -> User approves.

## Security Context

- The Black Box Risk: If you don't read the code, you don't see the vulnerabilities (e.g., hardcoded keys, insecure data handling).
- Blind Trust: Vibe coding prioritizes speed and functionality over security best practices.
- The "Vibe Check" Gap: It works ("good vibes"), but is it safe?



# Top 5 Risks in the Era of Vibe Coding

## 1. The "Hallucinated Package" Attack

The Vibe: "I need a library to parse this specific old file format."

The Risk: AI models sometimes invent package names that sound plausible but don't exist. Attackers can register these "hallucinated" names (Dependency Confusion) on public repositories (npm, PyPI) and inject malicious code that your vibe-coded app automatically downloads.

## 2. Hardcoded Secrets & API Keys

The Vibe: "Connect to this database and show me the user list."

The Risk: To get the code working instantly, AI often hardcodes credentials (AWS keys, DB passwords) directly into the source code rather than using environment variables. If the user doesn't read the code before pushing to GitHub, those secrets are public.

## 3. Insecure Default Configurations

The Vibe: "Spin up a quick server to handle these requests."

The Risk: AI prioritizes functionality over hardening. It may generate a server that runs as root, has debugging modes enabled (exposing stack traces), or allows all Cross-Origin (CORS) requests (Access-Control-Allow-Origin: \*), leaving the door wide open for attacks.



# Top 5 Risks in the Era of Vibe Coding

## 4. Logic Flaws & Edge Case Blindness

The Vibe: "Make sure users can only see their own data."

The Risk: The AI might implement a simple check (e.g., client-side validation) that looks like it works but lacks robust server-side enforcement. Because the human isn't auditing the logic, subtle flaws (like Insecure Direct Object References - IDOR) slip through.

## 5. Training Data Regurgitation (Copy-Paste Vulnerabilities)

The Vibe: "Write a function to sanitize user input."

The Risk: The AI is trained on vast amounts of internet code—including old, insecure Stack Overflow answers. It might suggest outdated hashing algorithms (like MD5) or vulnerable regex patterns (ReDoS) because they were statistically common in its training data.



# When Does Security Become a Concern?



## The "Safe Zone" (Low Risk)

**What it looks like:** Local-only prototypes, static HTML/CSS, visual mockups, "Hello World" apps.

**The Vibe:** "I'm just playing with colors and layout."

**Verdict:** Minimal security concern. If it breaks, only your local machine sees it.



# When Does Security Become a Concern?



## The Tipping Point (The Danger Zone Begins)

**External Data Ingestion:** The moment your code accepts input from the outside world (e.g., a file upload, a user comment, or a URL).

**Risk:** Injection attacks (SQLi, XSS) become possible.

**API Integration:** Connecting to OpenAI, Stripe, or a weather service.

**Risk:** This requires keys. Where is the AI storing them? (Likely hardcoded).

**Database Connections:** Moving from "mock data" to a real SQL or NoSQL database.

**Risk:** The AI might write queries that allow unauthorized access to user records.



# When Does Security Become a Concern?



## The "Red Alert" Zone (High Risk)

**Authentication & Identity:** Asking the AI to "build a login page."

Risk: Rolling your own auth is notoriously dangerous; AI might use weak hashing or skip session validation.

**Deployment:** Moving code from localhost to a public URL (Vercel, Netlify, AWS).

Risk: Your "vibe" is now discoverable by bots and scanners looking for exposed endpoints.



# Best Practice: Principle of Least Privilege

## No Credentials

Getting publicly available information such as race information.

## API Keys

Getting information about races you have access to either as an event director or partner.

## OAuth 2.0

Building applications that allow others to authenticate using their RunSignup account and authorizing access.



# Work Without Credentials

<https://runsignup.com/API/races/GET>

The screenshot shows the RunSignup API Documentation page for the 'Get Races' endpoint. The page has a teal header with the title 'RunSignup API Documentation'. Below the header, there is a navigation bar with links for 'Products', 'Use Cases', 'AI', 'Pricing', 'Knowledge Base', and 'About', along with a 'Find a Race' button and a user profile icon. The main content area is titled 'RunSignup / API / Methods / Get Races'. On the left, there is a sidebar menu with options: 'API Overview', 'Getting Started', 'API Keys', 'OAuth2 Authentication', 'Documentation', 'API Methods' (highlighted in pink), 'Error Codes', and 'Code Examples'. The main content area is titled 'Get Races' and contains the following information:

**Get Races**  
Gets a list of races, with information similar to [Get Race](#), except that only the most current events are listed. *Event start and end times are in the timezone of the race. All other dates/times are in Eastern Time.* The races are separated into pages. The maximum number of races per page is 1,000.

**URL**  
<https://api.runsignup.com/rest/races>

**HTTP Method**  
GET

**Standard Parameters**

PARAMETER	HTTP METHOD	DEFAULT	DESCRIPTION	DATATYPE
api_key	GET		API Key	string
api_secret	GET		API Secret	string
tmp_key	GET		Temporary API Key from login API call. This should NOT be used in combination with API Key.	string
tmp_secret	GET		Temporary API Secret from login API call. This should NOT be used in combination with API Secret.	string

“

I'd like to build a race finding app that will display upcoming races within a certain distance of a zip code. Use the RunSignup API for getting races.

The documentation for the get races api is available here:  
<https://runsignup.com/API/races/GET>.

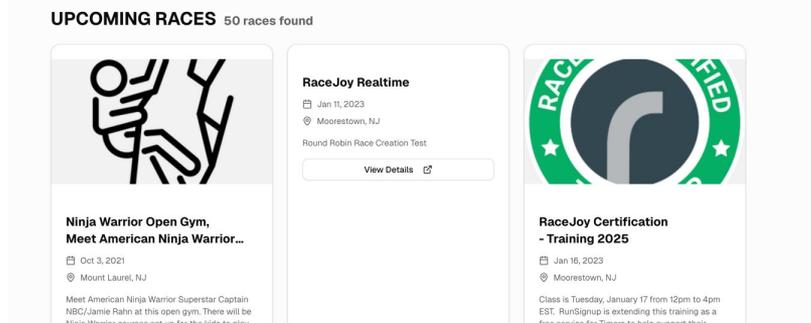
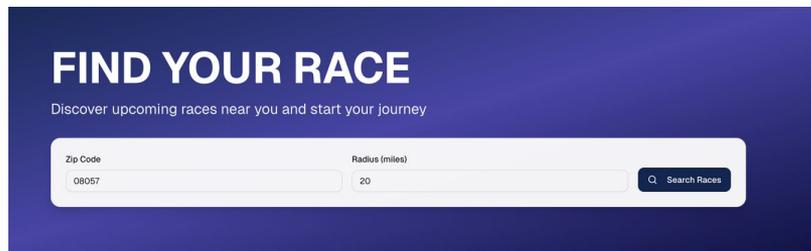
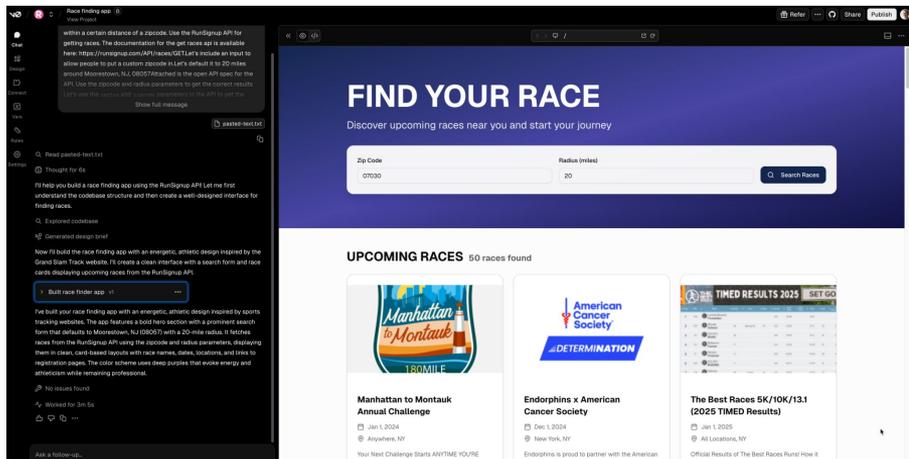
Include an input to allow people to put a custom zip code in.

Set the default to 20 miles around Moorestown, NJ, 08057

Attached is the open API spec for the API. Use the `radius` and `zipcode` parameters in the API to get the correct results

”

# Work Without Credentials



<https://v0-race-finding-app.vercel.app/>



# When Credentials Are Required

[https://test.runsignup.com/API/race/:race\\_id/participants/GET](https://test.runsignup.com/API/race/:race_id/participants/GET)



I have a race on test.runsignup.com with ID 7257. I'd like to create an app that lists the participants of the race for event ID 18220.

I'd like to use the API with this documentation::

[https://test.runsignup.com/API/race/:race\\_id/participants/GET](https://test.runsignup.com/API/race/:race_id/participants/GET).

The API endpoint is

[https://api.test.runsignup.com/rest/race/:race\\_id/participants](https://api.test.runsignup.com/rest/race/:race_id/participants)

Attached is the OpenAPI Specification

It will require using the `rsu\_api\_key` get parameter and the `X-RSU-API-SECRET` http header for authentication.

**Please handle the API key and Secret securely following best practices from a security perspective.**



# When Credentials Are Required

The screenshot displays a development environment with several components:

- Chat Window (Top Left):** A message from the AI assistant: "I have a race on test.runsignup.com with ID 7257. I'd like to create an app that lists the participants of the race for event ID 18220. I'd like to use the API with this documentation: https://test.runsignup.com/API/race/:race\_id/participants/GET. The endpoint is https://api.test.runsignup.com/rest/race/:race\_id/participants It will require using the rsu\_api\_key get parameter and the X-RSU-API-SECRET http header for authentication, please". A "Show full message" button is visible below the text.
- Code Editor (Center):** Shows the same AI message as the chat window. Below the message, a "Show full message" button is present.
- Terminal (Bottom):** Contains the following text:
  - Search: Read pasted-text.txt
  - Thought for 8s
  - Thought: "I'll help you create an app that lists race participants from the RunSignup API. Let me first understand the codebase structure, then build a secure implementation."
  - Analyzed codebase
  - Now I'll create the race participants app with secure API handling. I'll build a route handler that keeps the API credentials on the server, and a page to display the participants.
  - Created race participants app
  - Created race participants app
  - I've created a secure race participants app with the following features:
  - Security Best Practices:
  - Ask v0 to build...
  - Buttons: +, v0 Agent, Stop
- Environment Variables Dialog (Right):** Titled "Add Environment Variables". It contains:
  - Environment: All Environments
  - RSU\_API\_KEY: Enter a key
  - RSU\_API\_SECRET: Enter a key
  - Submit button
- Project Page (Background):** A browser window showing "Race Participants" with a red error message.
- Working Diagnostic (Bottom Right):** A small window titled "Working" with "Running Diagnostics..." and a close button.



# When Credentials Are Required

## Add Environment Variables ✕

This generation requires environment variables. Enter them below.

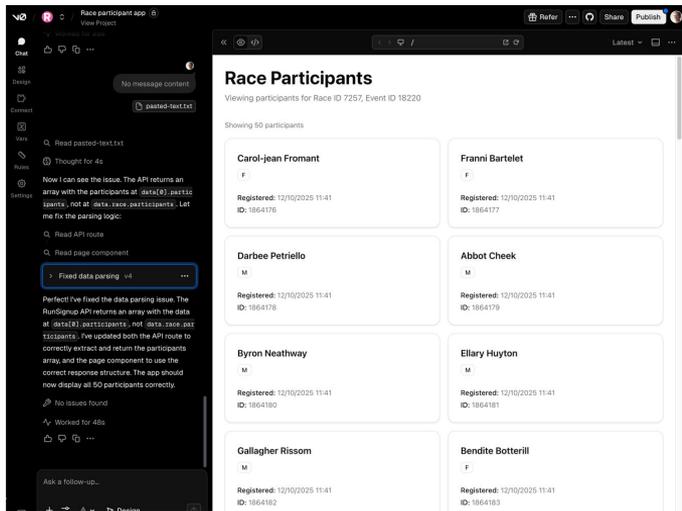
Environment ⓘ

 All Environments ▾

RSU\_API\_KEY

RSU\_API\_SECRET

# When Credentials Are Required



<https://v0-race-participant-app.vercel.app/>

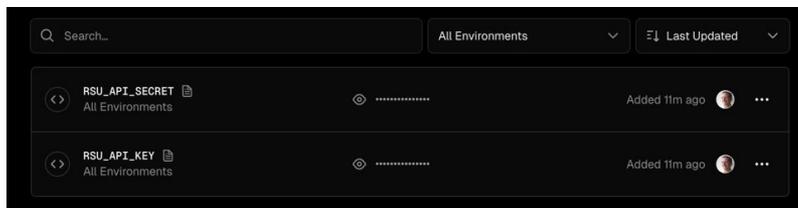
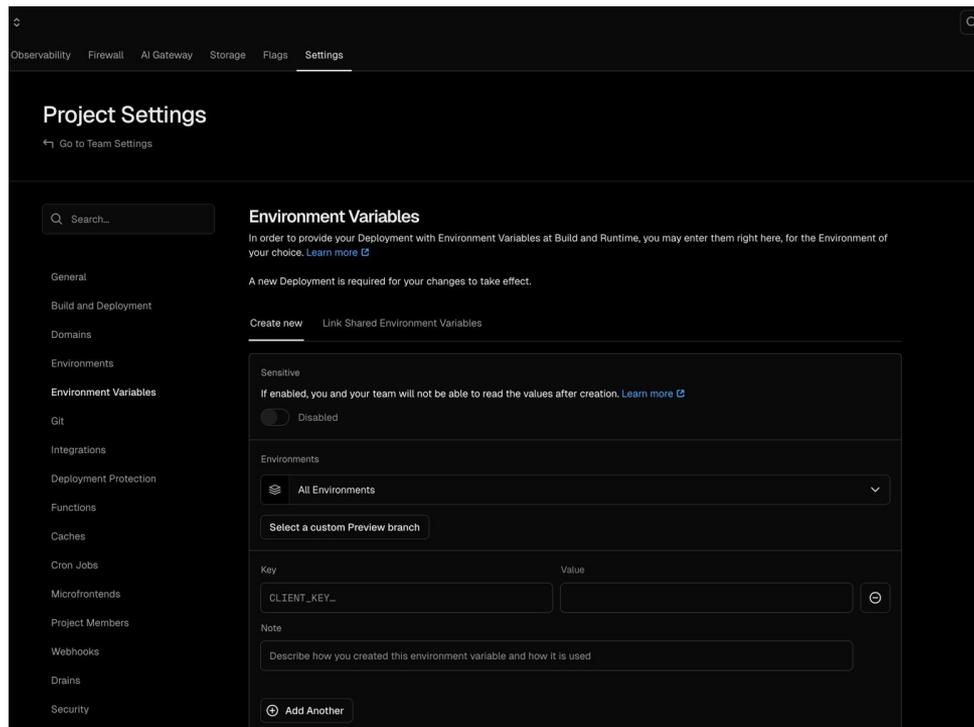
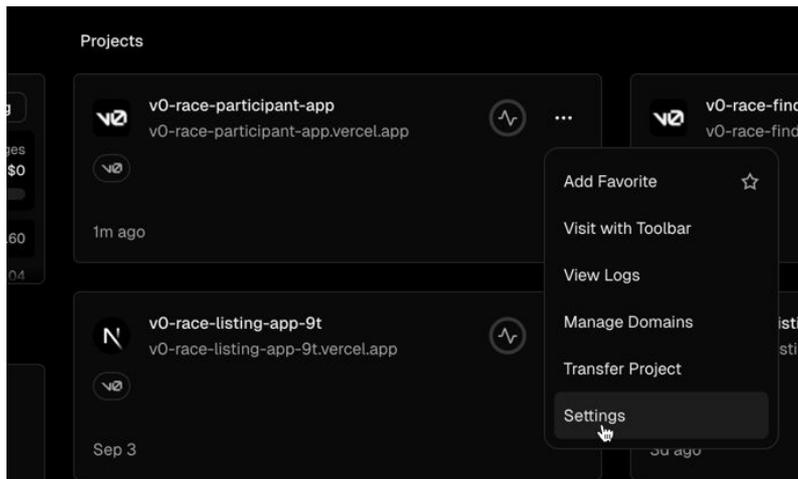
## Race Participants

Viewing participants for Race ID 7257, Event ID 18220

Showing 50 participants

<b>Carol-jean Fromant</b> F Registered: 12/10/2025 11:41 ID: 1864176	<b>Franni Bartelet</b> F Registered: 12/10/2025 11:41 ID: 1864177	<b>Darbee Petriello</b> M Registered: 12/10/2025 11:41 ID: 1864178
<b>Abbot Cheek</b> M Registered: 12/10/2025 11:41 ID: 1864179	<b>Byron Neathway</b> M Registered: 12/10/2025 11:41 ID: 1864180	<b>Ellary Huyton</b> M Registered: 12/10/2025 11:41 ID: 1864181
<b>Gallagher Rissom</b> M Registered: 12/10/2025 11:41 ID: 1864182	<b>Bendite Botterill</b> F Registered: 12/10/2025 11:41 ID: 1864183	<b>Shannan Priditt</b> M Registered: 12/10/2025 11:41 ID: 1864184
<b>Keri Metson</b> F Registered: 12/10/2025 11:41 ID: 1864185	<b>Nester Hanford</b> M Registered: 12/10/2025 11:41 ID: 1864186	<b>Glenna Oxherd</b> Registered: 12/10/2025 11:41 ID: 1864187

# Setting Environment Variables in Vercel



<https://v0-race-participant-app.vercel.app/>



# Setting Environment Variables in Vercel

The screenshot shows the Vercel environment variables management interface. At the top, there is a search bar with the placeholder text "Search...". To the right of the search bar are two dropdown menus: "All Environments" and "Last Updated". Below these are two rows of environment variables, each with a code icon, the variable name, the scope, a visibility icon, a masked value, the creation time, a user profile picture, and a menu icon.

Code Icon	Variable Name	Scope	Visibility Icon	Value	Added	User	More
<>	RSU_API_SECRET	All Environments	👁	.....	Added 11m ago		⋮
<>	RSU_API_KEY	All Environments	👁	.....	Added 11m ago		⋮



# Does The Tool Already Exist?

The screenshot shows the homepage of 'Timer Utilities'. At the top left is a green circular logo with a white wrench. To its right is a navigation menu with links for 'Home', 'Utilities', 'Subscribe', 'Update Subscription', and 'Contact Us'. A green 'Join' button and a user profile icon are also visible. The main heading 'Timer Utilities' is prominently displayed. Below the heading is a grid of eight utility icons, each in a green-bordered box with a label: 'Split Analyzer' (magnifying glass), 'Result Display' (television), 'Result Receipts' (printer), 'Awards Tracker' (trophy), 'RDS Utilities' (stopwatch), 'Manual Announcer' (microphone), 'Kiosk Browser' (computer monitor), and 'Make a Request' (question mark).

<https://runsignup.com/TimerUtilities>

The screenshot shows a race results page for 'Scott Coffee Moorestown Rotary 8K'. The page title is 'Race Results - Scott Coffee Moorestown Rotary 8K'. Below the title, it says 'Last updated: 1:37:45 PM • 499 results available' and has a 'Refresh' button. A search bar contains the name 'Sigwart' and a 'Search' button. Below the search bar, there are two columns of results. The first column shows 'STEPHEN SIGWART' with 'Bib #1114 • 37 • M' and 'BERLIN'. His times are 'Clock Time 29:56.6' and 'Chip Time 29:53.25'. The second column shows 'Andrew Sigwart' with 'Bib #1115 • 38 • M' and 'Berlin'. His times are 'Clock Time 44:04.22' and 'Chip Time 43:43.98'.



# The "Vibe Check" – A Security Checklist

## 1. The "No Naked Keys" Rule

- **The Action:** Never let the AI hardcode API keys or passwords.
- **The Prompt Fix:** Explicitly instruct the AI: *"Use environment variables for all secrets. Do not hardcode credentials."*

## 2. The 60-Second Syntax Audit

- **The Action:** Don't just run the code; read the logic blocks.
- **The Focus:** Look specifically at **imports** (Are these real libraries?) and **data flow** (Is user input going directly into a SQL query?).

## 3. "Adversarial" Prompting

- **The Action:** Use the AI to audit itself.
- **The Prompt Fix:** After the code is written, open a new chat window and paste the code back in with the prompt: *"Act as a security researcher. Find three potential vulnerabilities in this code and show me how to fix them."*



# Q/A