

HOW AI CODE EDITORS WORK (100% REAL ARCHITECTURE)

AI editors look magical — but they are ALL built from **5 core components**:

★ 1. Monaco Editor (VS Code in the browser)

Every AI tool uses **Monaco Editor**, which is the engine of Visual Studio Code.

- ✓ Used by: Bolt, Lovable, Cursor, Replit, Stackblitz
- ✓ Runs directly in browser
- ✓ Supports 100+ languages
- ✓ Syntax highlight
- ✓ Auto-complete
- ✓ Tabs & multi-file support
- ✓ Git diff support
- ✓ Perfect for AI patches

Why Monaco?

It gives **full VS Code experience** inside your web app.

★ 2. Virtual File System (VFS) in Browser

AI tools load your entire project into memory inside the browser:

```
{  
  "src/App.js": "...",  
  "src/index.js": "...",  
  "package.json": "...",  
}
```

- ✓ Fast
- ✓ Reduces server load
- ✓ Instant updates in editor

Then they sync back changes to backend storage.

★ 3. AI Patch / Diff Editing System

This is the **heart** of all AI coding tools.

When user asks AI:

- Fix code
- Generate feature
- Debug
- Refactor project

The tool sends:

- File code
- Folder structure
- All open files
- User prompt

to the AI model (Claude/GPT).

AI returns a diff/patch:

--- src/App.js

+++ src/App.js

@@@ -10,3 +10,7 @@@

```
+function greet(name){  
+ return `Hello ${name}`;  
+}  
+  
export default greet;
```

Monaco applies this patch automatically.

This gives the illusion of AI “directly editing” your code.

★ 4. Code Execution in Sandbox Containers

AI tools DO NOT run code in the browser.

Instead they run user code on servers, inside **isolated containers**:

- ✓ Docker
- ✓ Firecracker microVM
- ✓ Node/Python/Java runtimes
- ✓ Memory & timeout limits

Flow:

Button: Run →

Backend → Spawn sandbox →

Mount project files →

Execute →

Return output/logs →

Display in editor terminal

This is how Lovable, Replit, Bolt run user code safely.

★ 5. Backend for Storage + AI + Sandbox

Every AI tool has a backend (Node, Go, Python or Java):

Backend handles:

- ✓ AI requests to Claude/GPT
 - ✓ Applying patches
 - ✓ File read/write
 - ✓ Sandbox execution
 - ✓ User login
 - ✓ Project save/load
 - ✓ Token/plan limits
 - ✓ Logging activities
-

★ 6. Token / Credit System (Like Bolt)

This is to prevent unlimited usage.

- Each AI request costs X credits
- User has credit balance
- Backend deducts credits
- If 0 → ask to buy plan

All AI tools use virtual tokens, NOT Claude tokens.

★ 7. Version Control (Mini Internal Git)

Every AI tool secretly uses a Git-like system:

- ✓ Store file versions
- ✓ Undo/redo
- ✓ Compare changes
- ✓ Show diffs
- ✓ Save checkpoints

Bolt and Lovable both use internal commits.

★ 8. Live Preview (iframe + bundler)

For React/JS projects:

- Code processed by esbuild/Vite/Parcel
- Output is sent to an iframe
- Hot reload via WebSocket
- Instant UI update

Stackblitz started this model; now all AI tools use it.

★ 9. Project Export (ZIP)

When user clicks "Download Project":

- Backend zips project files
- Sends as download

Simple but essential.

⭐ 10. Multi-File AI Context Manager

AI needs the context of multiple files.

Tools send:

- Folder tree
- All related files
- Current file
- Errors
- Tasks needed

This lets Claude/GPT make multi-file changes.

✳ SUMMARY: How AI Code Editors Work (Same in All Tools)

Component	Purpose
Monaco Editor	Main editor UI (VS Code in browser)
Virtual File System	Stores files in browser
AI Patch System	Claude/GPT sends unified diff patches
Sandbox Container	Safe code execution
Backend APIs	AI calls, file saving, token control
Virtual Credits	Limits usage, allows pricing
Mini Git Engine	Version history, undo/redo

Component	Purpose
Live Preview	Shows frontend output instantly

ALL tools use EXACTLY this architecture.

❤️ YES, you can integrate this in YOUR application

You can add the same editor to your app with:

✓ Frontend

- React + Monaco Editor
- File Explorer
- AI patch apply
- Diff viewer
- Terminal + Preview

✓ Backend

- Spring Boot / Node.js
- Claude AI integration
- Patch calculation
- Docker sandbox
- Token deduction
- Project storage

✓ Infrastructure

- PostgreSQL
- S3/MinIO
- Docker Host
- Load Balancer