CodeViz

An AI-Powered Platform for Algorithm Visualization and Understanding

Shafır Cohen & Omer Efron

Table of Contents

Introduction: Bridging the Gap in CS Education

01

Mission Statement

Making complex algorithms accessible and understandable.

02

What is CodeViz?

An interactive web application designed to help users understand code through AI-generated visualizations and and explanations.

03

Core Technology

Leverages large language models to analyze code, visualize visualize execution, and provide detailed insights.

04

Workshop Context

Developed as a final project for an AI workshop focused on on educational technology applications.

The Challenge with Learning Algorithms

01

The Problem

Learning algorithms from static resources is difficult. The abstract nature of data structures poses a challenge for learners.

02

The Gap

There is a need for tools that offer deep, deep, contextual insights beyond simple simple interactivity.

03

Our Motivation

To build a dynamic, AI-enhanced platform platform that makes algorithm education education intuitive and effective.

Our Solution: CodeViz

01

Code Analysis

Input custom code for on-the-fly analysis.

02

Interactive Visualization

Generates animated, step-by-step visualizations.

03

AI-Powered Insights

Provides complexity analysis, explanations, and real-world metaphors. metaphors.

04

Project Goals

Demystify algorithms like Huffman Encoding and Binary Search

Enable hands-on learning

Showcase LLM applications in EdTech

eViz AI-Powered

Huffman Encoding | Code Analyzer | Examples

AI Code Analyzer

Paste any algorithm code for instant AI-powered analysis and visualization suggestions

thm Code Analyzer

Binary Search | Bubble Sort | Fibonacci Sequer

r algorithm code:

```
= 0, b = 1;
let i = 2; i <= n; i++) {
t temp = a + b;
= b;
= temp;

rn b;
```

Analyzing algorithm with AI...

Lightning storm of AI power

Analyzing with AI...

AI Code Analyzer

Paste any algorithm code for instant AI-powered analysis and visualization suggestions

Algorithm Code Analyzer

Binary Search | Bubble Sort | Fibonacci Sequence

Enter your algorithm code:

Paste or type your algorithm code here...

Analyze Code

Interactive Visualization

Binary Search Visualization

Searching for **7** in sorted array

Binary search efficiently finds elements by repeatedly dividing the search space in half

Play | Stop | Reset | Speed: Normal

Step 1 of 9

Starting binary search for 7 in sorted array

Left: 0 | Right: 9 | Mid: N/A

Previous | Step: 1/9 | Next

Algorithm Complexity

Time Complexity: O(log n) | Space Complexity: O(1)

odeViz AI-Powered

Huffman Encoding | Code Analyzer | Examples | Help

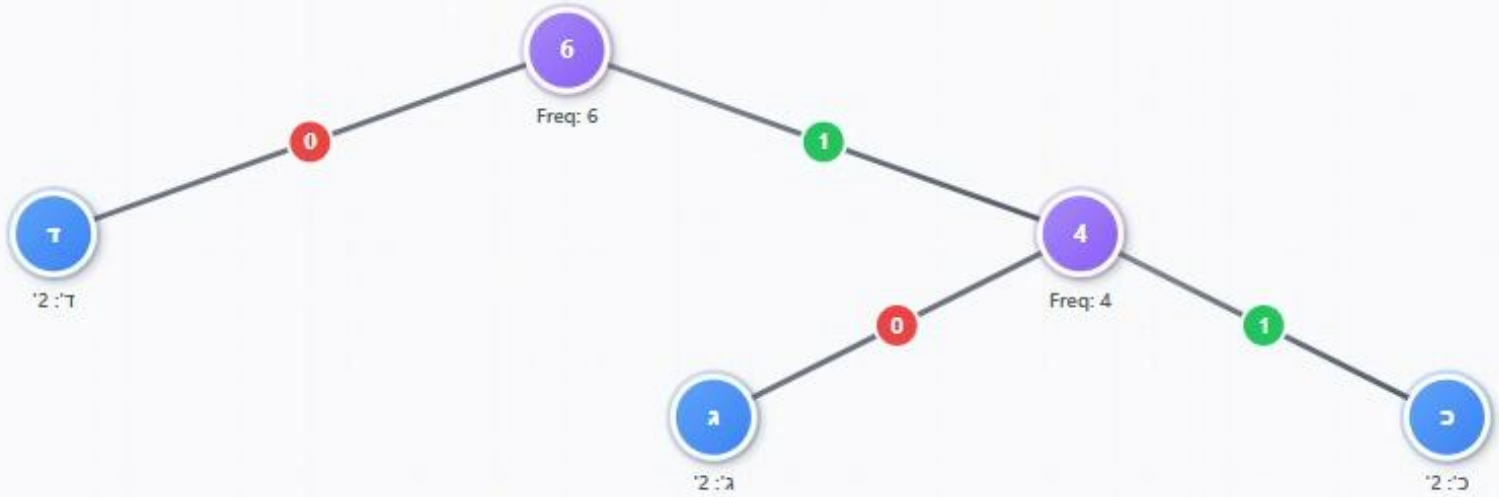
Huffman Encoding

Watch how text gets compressed using optimal binary codes. Enter any text below to see the step-by-step visualization.

Enter your text:

Type or paste your text here... Example: 'hello world' or 'aavdvdsvvsdvsdq'

Generate Huffman Code



Leaf Nodes (Characters) Internal Nodes 0 Left Edge 1 Right Edge

Algorithm Examples

Explore pre-analyzed algorithms with interactive visualizations

Search algorithms, descriptions, or tags...

All Categories

5 examples found

Easy Medium Hard

Binary Search Medium

Efficiently finds an element in a sorted array by repeatedly dividing the search space in half.

Time: O(log n)

Space: O(1)

divide-and-conquer | sorted-array | efficient

Searching Explore →

Bubble Sort Easy

A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order.

Time: $O(n^2)$

Space: O(1)

comparison-sort | in-place | stable

Sorting Explore →

Huffman Encoding Hard

A lossless data compression algorithm that assigns variable-length codes to characters based on their frequencies.

Time: O(n log n)

Space: O(n)

greedy | compression | binary-tree

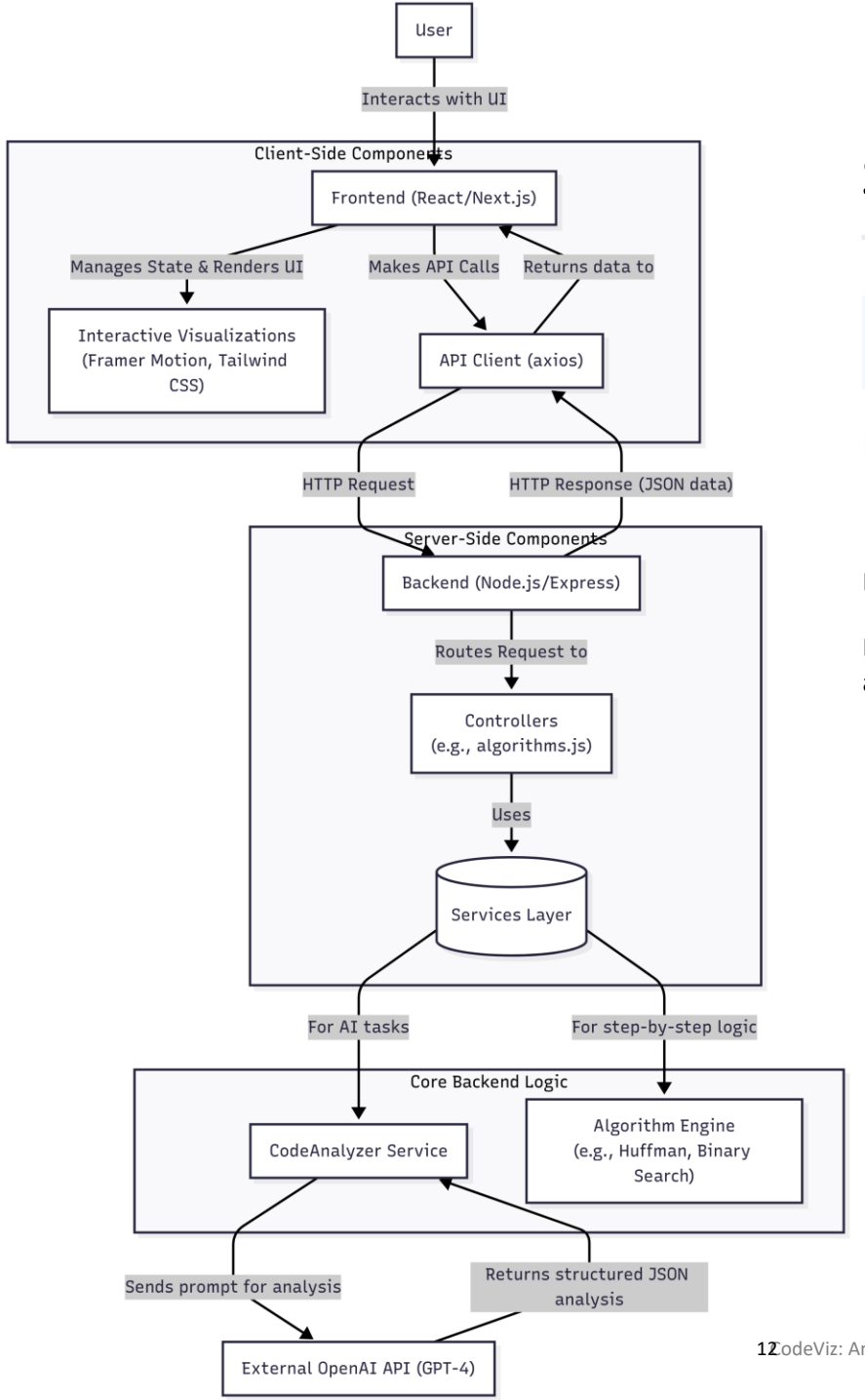
Compression Explore →

Fibonacci Sequence Easy

Computes the nth Fibonacci number using dynamic programming approach.

Quick Sort Hard

A divide-and-conquer sorting algorithm that picks a pivot and partitions the array around it.



System Architecture

01

Frontend

Built using Next.js and React

Enhanced with Framer Motion for animations

02

Backend

Powered by Node.js and Express.js

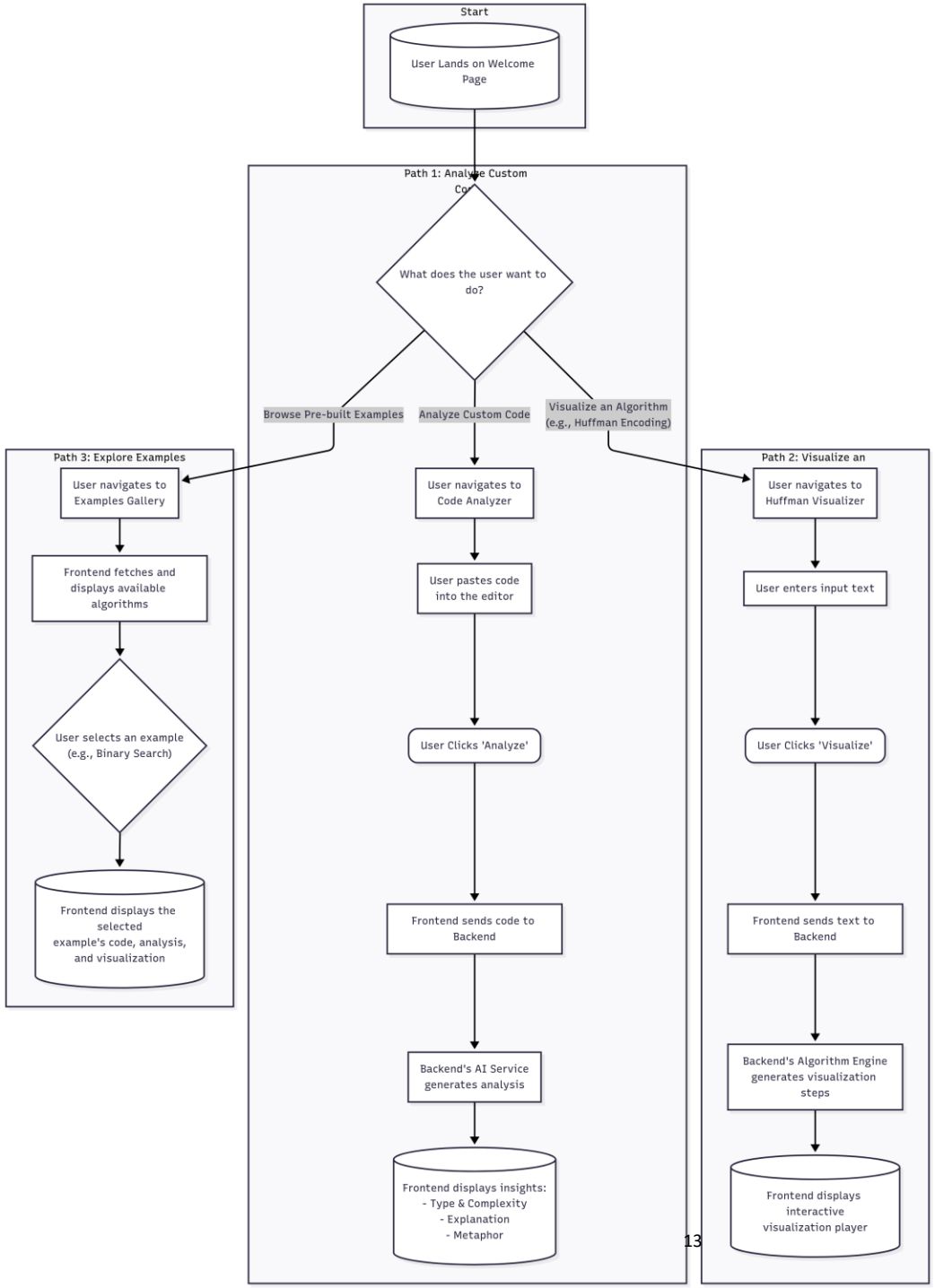
Handles processing and communication with AI service

03

AI Service & Engine

Integrates GPT-4 via OpenAI API

Custom algorithm engine for step generation



The User Journey

01

Step 1: Landing & Selection

User chooses a mode: analyze, visualize, or view example.

02

Step 2: Input

User inputs code or query for processing.

03

Step 3: Processing

Backend and AI service analyze input and generate insights.

04

Step 4: Output

Frontend displays results with visualizations and explanations.

Inside the Backend

01

API Routes

Manage all HTTP requests

Examples: /api/analyze, /api/algorithms/hoffman

02

Controllers

Includes logic in files like visualizationController.js and algorithms.js

03

Services

Factory-based LLM service

CodeAnalyzer service for handling parsing

04

Parsers & Models

cParser.js identifies structures

Algorithm.js creates execution state

UI & Visualization Showcase

01

Code Input

Component: CodeAnalyzer

User-friendly editor for code input

02

AI Analysis Results

Displays complexity metrics and real-world metaphors

03

Live Visualization

Animated rendering of algorithm steps (e.g. Huffman tree)

04

Interactive Controls

Playback, step navigation, and parameter tweaking

Challenges & Key Learnings

01

Challenges

Building a generic visualization engine

Prompt engineering for AI

Frontend state management

02

Key Learnings

Power and flexibility of LLMs

Full-stack integration skills

Significance of user experience design

AI derived developing

The Road Ahead: Future Work

01

Algorithm Support

AI generated games for algorithms

Debugging visualization

02

AI Features

Add AI-generated test cases

Introduce a code improvement advisor

03

Platform Features

Implement user account system

Enable collaboration features

Conclusion

01

Project Summary

CodeViz demonstrates how AI can revolutionize algorithm education with visual, hands-on learning.

02

Outcomes

Built a full-stack app with LLM integration

Developed a custom step-wise algorithm engine

03

Final Thoughts

AI tools like CodeViz are paving the way for the future of technical education. education.

CodeViz

An AI-Powered Platform for Algorithm Visualization and Understanding

Shafır Cohen & Omer Efron