

## File Structure

| EXPLORER                    | ... |
|-----------------------------|-----|
| ✓ <b>SORTING-VISUALIZER</b> |     |
| ✓ backend                   | ●   |
| > .github                   | ●   |
| > data                      |     |
| > node_modules              |     |
| > routes                    |     |
| JS index.js                 |     |
| {} package-lock.json        |     |
| {} package.json             |     |
| ✓ frontend                  | ●   |
| > .github                   |     |
| > node_modules              |     |
| ✓ src                       | ●   |
| ✓ algos                     |     |
| JS bubble.js                |     |
| JS insertion.js             |     |
| JS merge.js                 |     |
| JS selection.js             |     |
| > components                |     |
| > lib                       |     |
| 🌀 App.jsx                   |     |
| # index.css                 | 9+  |
| 🌀 main.jsx                  |     |
| <> index.html               |     |
| {} package-lock.json        | M   |
| {} package.json             | M   |
| JS postcss.config.cjs       |     |
| JS postcss.config.js        |     |
| JS tailwind.config.cjs      |     |
| JS tailwind.config.js       |     |
| ⚡ vite.config.js            |     |

## CODE:

### Backend:

```
JS index.js ×
backend > JS index.js > ...
1  const express = require('express')
2  const cors = require('cors')
3  const bodyParser = require('body-parser')
4  const logsRouter = require('./routes/logs')
5
6  const app = express()
7  const PORT = process.env.PORT || 5000
8
9  // Middleware
10 app.use(cors())
11 app.use(bodyParser.json())
12
13 // Routes
14 app.use('/api/logs', logsRouter)
15
16 // Health check
17 app.get('/', (req, res) => {
18   res.send('Sorting Visualizer Backend Running!')
19 })
20
21 app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
--
```

```
JS logs.js ×
backend > routes > JS logs.js > ...
1  const express = require('express')
2  const fs = require('fs')
3  const path = require('path')
4
5  const router = express.Router()
6  const filePath = path.join(__dirname, '../data/logs.json')
7
8  // GET all logs
9  router.get('/', (req, res) => {
10   try {
11     const logs = JSON.parse(fs.readFileSync(filePath, 'utf8'))
12     res.json(logs)
13   } catch (err) {
14     console.error(err)
15     res.status(500).json({ error: 'Failed to read logs' })
16   }
17 })
18
19 // POST a new log
20 router.post('/', (req, res) => {
21   try {
22     const logs = JSON.parse(fs.readFileSync(filePath, 'utf8'))
23     const newLog = req.body
24     logs.push(newLog)
25     fs.writeFileSync(filePath, JSON.stringify(logs, null, 2))
26     res.status(201).json({ message: 'Log saved!' })
27   } catch (err) {
28     console.error(err)
29     res.status(500).json({ error: 'Failed to save log' })
30   }
31 })
32
33 module.exports = router
34
```

```
{ logs.json X
backend > data > {} logs.json > ...
1  [
2    {
3      "algorithm": "Bubble",
4      "size": 40,
5      "duration": 45767,
6      "timestamp": "2025-10-11T20:17:27.698Z"
7    },
8    {
9      "algorithm": "Bubble",
10     "size": 10,
11     "duration": 2336,
12     "timestamp": "2025-10-11T20:23:04.693Z"
13   },
14   {
15     "algorithm": "Bubble",
16     "size": 10,
17     "duration": 2129,
18     "timestamp": "2025-10-11T20:23:10.750Z"
19   },
20   {
21     "algorithm": "Selection",
22     "size": 10,
23     "duration": 1623,
24     "timestamp": "2025-10-11T20:23:15.965Z"
25   },
26   {
27     "algorithm": "Insertion",
28     "size": 10,
29     "duration": 1463,
30     "timestamp": "2025-10-11T20:23:41.626Z"
31   },
32   {
33     "algorithm": "Bubble",
34     "size": 40,
35     "duration": 36132,
36     "timestamp": "2025-10-11T20:25:18.578Z"
37   },
38   {
39     "algorithm": "Bubble",
40     "size": 10,
41     "duration": 2310,
42     "timestamp": "2025-10-11T20:30:25.888Z"
43   },
44   {
45     "algorithm": "Bubble",
46     "size": 36,
47     "duration": 30857,
48     "timestamp": "2025-10-11T20:31:38.157Z"
49   },
50   {
51     "algorithm": "Bubble",
52     "size": 10,
53     "duration": 2027,
54     "timestamp": "2025-10-11T20:32:30.895Z"
55   },
56   {
57     "algorithm": "Selection",
58     "size": 10,
59     "duration": 1685,
60     "timestamp": "2025-10-11T20:33:03.101Z"
61   },
62   {
63     "algorithm": "Insertion",
64     "size": 10,
65     "duration": 2095,
66     "timestamp": "2025-10-11T20:33:37.780Z"
67   },
68   {
69     "algorithm": "Merge",
70     "size": 10,
71     "duration": 1651,
72     "timestamp": "2025-10-11T20:33:50.552Z"
73   },
74   {
75     "algorithm": "Merge",
76     "size": 10,
77     "duration": 1698,
78     "timestamp": "2025-10-11T20:33:55.775Z"
79   },
80   {
81     "algorithm": "Bubble"
```

# Frontend:

```
SortingVisualizer.jsx X
frontend > src > components > SortingVisualizer.jsx > SortingVisualizer > run
1 import React, { useState, useEffect } from 'react'
2 import { bubble } from '../algos/bubble'
3 import { selection } from '../algos/selection'
4 import { insertion } from '../algos/insertion'
5 import { mergeSort } from '../algos/merge'
6 import { motion } from 'framer-motion'
7 import { logRun } from '../lib/api'
8
9 const ALGS = { Bubble: bubble, Selection: selection, Insertion: insertion, Merge: mergeSort }
10
11 function rand(n, max = 220) {
12   const a = []
13   for (let i = 0; i < n; i++) a.push(Math.floor(Math.random() * max) + 6)
14   return a
15 }
16
17 export default function SortingVisualizer() {
18   const [arr, setArr] = useState(() => rand(40))
19   const [size, setSize] = useState(40)
20   const [speed, setSpeed] = useState(30)
21   const [alg, setAlg] = useState('Bubble')
22   const [running, setRunning] = useState(false)
23   const [result, setResult] = useState(null)
24
25   useEffect(() => setArr(rand(size)), [size])
26
27   function reset() {
28     if (running) return
29     setArr(rand(size))
30     setResult(null)
31   }
32
33   async function run() {
34     if (running) return
35     setRunning(true)
36     setResult(null)
37
38     const fn = ALGS[alg]
39     const steps = fn(arr)
40     const copy = arr.slice()
41     const t0 = Date.now()
42
43     for (let s = 0; s < steps.length; s++) {
44       const it = steps[s]
45       if (it.type === 'compare') {
46         await sleep(speed)
47       } else if (it.type === 'swap') {
48         const { i, j } = it
49         const tmp = copy[i]
50         copy[i] = copy[j]
51         copy[j] = tmp
52         setArr(copy.slice())
53         await sleep(speed)
54       } else if (it.type === 'overwrite') {
55         copy[it.idx] = it.val
56         setArr(copy.slice())
57         await sleep(speed)
58       }
59     }
60
61     const duration = Date.now() - t0
62     logRun({ algorithm: alg, size, duration, timestamp: new Date().toISOString() })
63     setResult(`Algorithm: ${alg} | Size: ${size} | Time: ${duration} ms`)
64     setRunning(false)
65   }
66
67   function sleep(ms) {
68     return new Promise((r) => setTimeout(r, ms))
69   }
70
71   return (
72     <div className="p-4 max-w-6xl mx-auto">
73       {/* Controls */}
74       <div className="flex flex-wrap gap-4 items-center justify-between mb-4">
75         <div className="flex gap-4 items-center flex-wrap">
76           <label className="text-white font-semibold">Algorithm:</label>
77           <select
78             value={alg}
79             onChange={(e) => setAlg(e.target.value)}
80             className="p-2 rounded bg-gray-700 text-white"
81           >
```

SortingVisualizer.jsx ✕

frontend > src > components >  SortingVisualizer.jsx >  SortingVisualizer >  run

```

81     >
82     {Object.keys(ALGS).map((k) => (
83       <option key={k} value={k}>
84         {k}
85       </option>
86     ))}
87   </select>
88
89   <label className="text-white font-semibold">Array Size:</label>
90   <input
91     type="range"
92     min="10"
93     max="150"
94     value={size}
95     onChange={(e) => setSize(Number(e.target.value))}
96     className="w-32"
97   />
98   <span className="text-white">{size}</span>
99
100  <label className="text-white font-semibold">Speed (ms):</label>
101  <input
102    type="range"
103    min="5"
104    max="200"
105    value={speed}
106    onChange={(e) => setSpeed(Number(e.target.value))}
107    className="w-32"
108  />
109  <span className="text-white">{speed} ms</span>
110</div>
111
112<div className="flex gap-2">
113  <button
114    className="px-3 py-1 bg-red-600 rounded hover:bg-red-700 disabled:opacity-50"
115    onClick={reset}
116    disabled={running}
117  >
118    Randomize
119  </button>
120  <button
121    className="px-3 py-1 bg-green-600 rounded hover:bg-green-700 disabled:opacity-50"
122    onClick={run}
123    disabled={running}
124  >
125    Start
126  </button>
127</div>
128</div>
129
130{/* Bars */}
131<div
132  style={{ height: Math.min(520, Math.max(260, size * 3.5)) }}
133  className="relative bg-[rgba(255,255,255,0.02)] rounded-md p-3"
134>
135  <div className="flex items-end gap-1 h-full">
136    {arr.map((v, i) => (
137      <motion.div
138        key={i}
139        className="bar bg-gradient-to-b from-cyan-400/80 to-sky-600/80 rounded"
140        style={{ height: `${v}px`, width: `${Math.max(4, Math.floor(1000 / size))}px` }}
141        layout
142        transition={{ type: 'spring', stiffness: 300, damping: 30 }}
143      />
144    ))}
145  </div>
146</div>
147
148{/* Result */}
149{result && (
150  <div className="mt-4 p-2 bg-green-800 text-white rounded">
151    {result}
152  </div>
153)}
154</div>
155)
156}

```

JS bubble.js X

frontend > src > algos > JS bubble.js > ...

```
1 export function bubble(arr) {
2   const steps = []
3   const a = arr.slice()
4   const n = a.length
5
6   for (let i = 0; i < n - 1; i++) {
7     for (let j = 0; j < n - i - 1; j++) {
8       // Step: compare
9       steps.push({ type: 'compare', i: j, j: j + 1 })
10
11       if (a[j] > a[j + 1]) {
12         // Step: swap
13         [a[j], a[j + 1]] = [a[j + 1], a[j]]
14         steps.push({ type: 'swap', i: j, j: j + 1 })
15       }
16     }
17   }
18
19   return steps
20 }
21
```

JS insertion.js X

frontend > src > algos > JS insertion.js > insertion

```
1 export function insertion(a) {
2   const arr = a.slice(), steps = [];
3   for (let i = 1; i < arr.length; i++) {
4     let j = i;
5     while (j > 0) {
6       steps.push({ type: 'compare', i: j - 1, j });
7       if (arr[j - 1] > arr[j]) {
8         steps.push({ type: 'swap', i: j - 1, j });
9         [arr[j - 1], arr[j]] = [arr[j], arr[j - 1]];
10      } else break;
11      j--;
12    }
13  }
14  return steps;
15 }
```

JS insertion.js M

JS merge.js M X

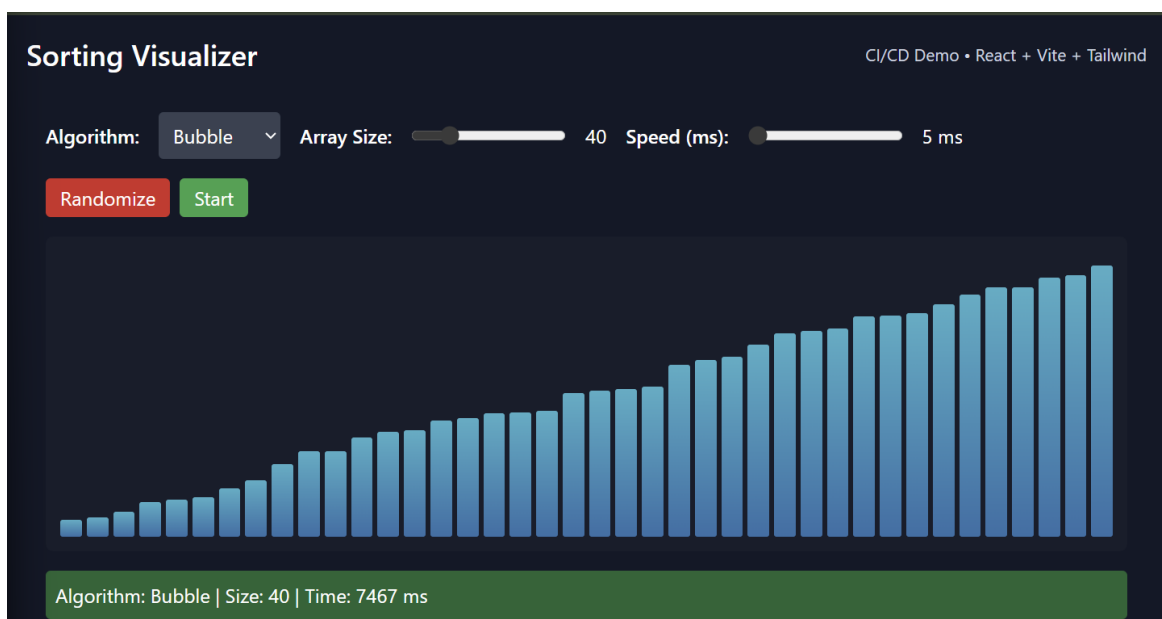
frontend > src > algos > JS merge.js > ...

```
1 export function mergeSort(a) {
2   const arr = a.slice(), steps = [];
3   function merge(l, m, r) {
4     const L = arr.slice(l, m + 1), R = arr.slice(m + 1, r + 1);
5     let i = 0, j = 0, k = l;
6     while (i < L.length && j < R.length) {
7       steps.push({ type: 'compare', i: l + i, j: m + 1 + j });
8       if (L[i] <= R[j]) steps.push({ type: 'overwrite', idx: k, val: L[i] }), arr[k++] = L[i++];
9       else steps.push({ type: 'overwrite', idx: k, val: R[j] }), arr[k++] = R[j++];
10    }
11    while (i < L.length) steps.push({ type: 'overwrite', idx: k, val: L[i] }), arr[k++] = L[i++];
12    while (j < R.length) steps.push({ type: 'overwrite', idx: k, val: R[j] }), arr[k++] = R[j++];
13  }
14  function ms(l, r) { if (l < r) { const m = (l + r) >> 1; ms(l, m); ms(m + 1, r); merge(l, m, r); } }
15  ms(0, arr.length - 1);
16  return steps;
17 }
18 // --- IGNORE ---
```

```
JS insertion.js M JS merge.js M JS selection.js X
frontend > src > algos > JS selection.js > selection
1 export function selection(a){
2   const arr=a.slice(); const steps=[]; const n=arr.length;
3   for(let i=0;i<n-1;i++){
4     let min=i;
5     for(let j=i+1;j<n;j++){
6       steps.push({type:'compare', i:min, j});
7       if(arr[j]<arr[min]) min=j;
8     }
9     if(min!==i){ steps.push({type:'swap', i, j:min}); const t=arr[i]; arr[i]=arr[min]; arr[min]=t }
10  }
11  return steps;
12 }
```

```
JS insertion.js M JS merge.js M App.jsx X
frontend > src > App.jsx > ...
1 import React from 'react'
2 import Navbar from './components/Navbar'
3 import SortingVisualizer from './components/SortingVisualizer'
4
5 function App() {
6   return (
7     <div className="min-h-screen bg-gray-900 text-white p-4">
8       <Navbar />
9       <SortingVisualizer />
10    </div>
11  )
12 }
13
14 export default App
15
```

## OUTPUT:



STATIC SITE

## Frontend FSD

Manual Deploy

Service ID: `srv-d3lnumqdbo4c73b8r7u0`

Sameer2-coder / FSD-EXP main

<https://frontend-fsd-cwo7.onrender.com>October 12, 2025 at 3:58 PM ✓ Live`4bcd814` Initial commit: Sorting Visualizer frontend and backend

... All logs

Live tail

GMT+5:30



Search

```
Oct 12 03:58:31 PM ⓘ
Oct 12 03:58:31 PM ⓘ > frontend@1.0.0 build
Oct 12 03:58:31 PM ⓘ > vite build
Oct 12 03:58:31 PM ⓘ
Oct 12 03:58:32 PM ⓘ vite v4.5.14 building for production...
Oct 12 03:58:32 PM ⓘ transforming...
Oct 12 03:58:34 PM ⓘ ✓ 429 modules transformed.
Oct 12 03:58:34 PM ⓘ rendering chunks...
Oct 12 03:58:34 PM ⓘ computing gzip size...
Oct 12 03:58:34 PM ⓘ dist/index.html          0.46 kB | gzip: 0.3
2 kB
Oct 12 03:58:34 PM ⓘ dist/assets/index-37b42b16.css  8.07 kB | gzip: 2.3
9 kB
Oct 12 03:58:34 PM ⓘ dist/assets/index-caf00119.js 261.16 kB | gzip: 84.8
8 kB
Oct 12 03:58:34 PM ⓘ ✓ built in 2.06s
Oct 12 03:58:35 PM ⓘ ==> Uploading build...
Oct 12 03:58:42 PM ⓘ ==> Your site is live 🎉
```



Node

Free

Upgrade your instance →

Service ID: `srv-d3lo2mqdbo4c73b8uim0`

Sameer2-coder / FSD-EXP main

<https://backend-fsd-724r.onrender.com>

ⓘ Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.

[Upgrade now](#)

October 12, 2025 at 3:52 PM ✓ Live

`4bcd814` Initial commit: Sorting Visualizer frontend and backend

...

All logs

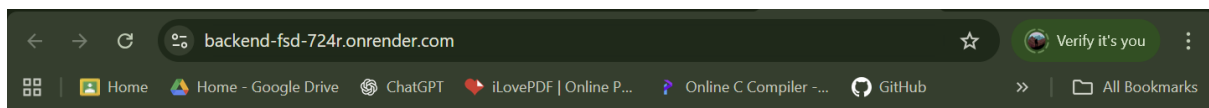
Live tail

GMT+5:30



Search

```
Oct 12 03:53:05 PM ⓘ ==>
Oct 12 03:53:05 PM ⓘ ==>
////////////////////////////////////
Oct 12 03:53:05 PM ⓘ ==>
Oct 12 03:53:05 PM ⓘ ==> Available at your primary URL https://back
end-fsd-724r.onrender.com
Oct 12 03:53:05 PM ⓘ ==>
Oct 12 03:53:05 PM ⓘ ==>
////////////////////////////////////
Oct 12 03:58:08 PM ⓘ ==> Detected service running on port 10000
Oct 12 03:58:08 PM ⓘ ==> Docs on specifying a port: https://render.
com/docs/web-services#port-binding
Oct 12 03:58:08 PM ⓘ ==> Detected service running on port 10000
Oct 12 03:58:08 PM ⓘ ==> Docs on specifying a port: https://render.
com/docs/web-services#port-binding
```



## LINKS:-

<https://frontend-fsd-cwo7.onrender.com/>

<https://github.com/Sameer2-coder/FSD-EXP>