Algorithm Visualizer Project Documentation

🚀 Algorithm Visualizer: Complete Project Guide

📁 Project Structure & Setup

Recommended Tech Stack
- **Frontend**: React.js + Vite (faster than Create React App)
- **Styling**: TailwindCSS + Framer Motion
- **State Management**: Redux Toolkit
- **Database**: Firebase Firestore
- **Graph Visualization**: Cytoscape.js
- **Deployment**: Vercel/Netlify + Firebase Hosting

Folder Structure
```
algorithm-visualizer/
├── public/
│   └── index.html
├── src/
│   ├── assets/              # Images, icons, static files
│   ├── components/          # Reusable UI components
│   │   ├── common/
│   │   │   ├── Button.jsx
│   │   │   ├── Modal.jsx
│   │   │   └── Loader.jsx
│   │   ├── visualization/
│   │   │   ├── ArrayVisualizer.jsx
│   │   │   ├── GraphVisualizer.jsx
│   │   │   └── CodeHighlighter.jsx
│   │   └── layout/
│   │       ├── Header.jsx
│   │       ├── Sidebar.jsx
│   │       └── Footer.jsx
│   ├── features/            # Feature-based modules
│   │   ├── algorithms/
│   │   │   ├── sorting/
│   │   │   │   ├── bubbleSort.js
```

```
│   │   │   │   ├── quickSort.js
│   │   │   │   └── index.js
│   │   │   ├── searching/
│   │   │   ├── graph/
│   │   │   └── index.js
│   │   ├── auth/
│   │   │   ├── Login.jsx
│   │   │   └── Register.jsx
│   │   └── dashboard/
│   │       ├── Dashboard.jsx
│   │       └── UserProfile.jsx
│   ├── hooks/                # Custom React hooks
│   │   ├── useAlgorithm.js
│   │   ├── useFirebase.js
│   │   └── useVisualization.js
│   ├── lib/                  # Third-party integrations
│   │   ├── firebase.js
│   │   ├── cytoscape.js
│   │   └── animations.js
│   ├── store/                # Redux store
│   │   ├── slices/
│   │   │   ├── algorithmSlice.js
│   │   │   ├── uiSlice.js
│   │   │   └── userSlice.js
│   │   └── store.js
│   ├── utils/                # Utility functions
│   │   ├── algorithmUtils.js
│   │   ├── validation.js
│   │   └── constants.js
│   ├── styles/               # Global styles
│   │   └── globals.css
│   ├── App.jsx
│   ├── main.jsx
│   └── routes.jsx
├── tests/                    # Test files
├── docs/                     # Documentation
├── .github/                  # GitHub workflows
├── package.json
├── vite.config.js
├── tailwind.config.js
└── firebase.json
```

🚀 Initial Setup Commands

```bash
# Create project with Vite
npm create vite@latest algorithm-visualizer --template react
cd algorithm-visualizer

# Install dependencies
npm install @reduxjs/toolkit react-redux firebase cytoscape cytoscape-fcose cytoscape-dagre
framer-motion lucide-react

# Install dev dependencies
npm install -D tailwindcss postcss autoprefixer @testing-library/jest-dom

# Initialize TailwindCSS
npx tailwindcss init -p
```

🔥 Firebase Setup

1. Create Firebase Project
1. Go to [Firebase Console](https://console.firebase.google.com/)
2. Click "Create Project"
3. Enter project name "algorithm-visualizer"
4. Enable Google Analytics (optional)
5. Click "Create Project"

2. Set Up Authentication
1. In Firebase Console, go to Authentication → Sign-in method
2. Enable Email/Password and Google sign-in

3. Set Up Firestore Database
1. Go to Firestore Database → Create Database
2. Start in test mode (for development)
3. Choose location closest to your users

4. Firebase Configuration File
```javascript
// src/lib/firebase.js
import { initializeApp } from 'firebase/app';
import { getAuth } from 'firebase/auth';
import { getFirestore } from 'firebase/firestore';
import { getStorage } from 'firebase/storage';

const firebaseConfig = {
```

```
  apiKey: import.meta.env.VITE_FIREBASE_API_KEY,
  authDomain: import.meta.env.VITE_FIREBASE_AUTH_DOMAIN,
  projectId: import.meta.env.VITE_FIREBASE_PROJECT_ID,
  storageBucket: import.meta.env.VITE_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: import.meta.env.VITE_FIREBASE_MESSAGING_SENDER_ID,
  appId: import.meta.env.VITE_FIREBASE_APP_ID
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

// Initialize Firebase services
export const auth = getAuth(app);
export const db = getFirestore(app);
export const storage = getStorage(app);

export default app;
```

5. Environment Variables (.env.local)
```
VITE_FIREBASE_API_KEY=your_api_key
VITE_FIREBASE_AUTH_DOMAIN=your_project.firebaseapp.com
VITE_FIREBASE_PROJECT_ID=your_project_id
VITE_FIREBASE_STORAGE_BUCKET=your_project.appspot.com
VITE_FIREBASE_MESSAGING_SENDER_ID=your_sender_id
VITE_FIREBASE_APP_ID=your_app_id
```

📦 Package Explanations

Core Dependencies
Package	Purpose	Usage Example
React	UI framework	Component-based architecture
Redux Toolkit	State management	Global app state management
Firebase	Backend services	Auth, database, storage
Cytoscape	Graph visualization	Network/graph algorithms
Framer Motion	Animations	Smooth UI transitions
TailwindCSS	Styling	Utility-first CSS framework
Lucide React	Icons	Consistent iconography

🧠 Algorithm Implementation Structure

Example: Bubble Sort Implementation
```javascript
// src/features/algorithms/sorting/bubbleSort.js
/**
 * Bubble Sort Algorithm with step-by-step visualization
 * Purpose: Demonstrates comparison-based sorting with O(n²) complexity
 */

export function* bubbleSort(arr) {
  let n = arr.length;
  let steps = [];

  for (let i = 0; i < n - 1; i++) {
    for (let j = 0; j < n - i - 1; j++) {
      // Yield current state for visualization
      yield {
        array: [...arr],
        compared: [j, j + 1],
        swapped: false,
        description: `Comparing ${arr[j]} and ${arr[j + 1]}`
      };

      if (arr[j] > arr[j + 1]) {
        // Swap elements
        [arr[j], arr[j + 1]] = [arr[j + 1], arr[j]];

        // Yield swapped state
        yield {
          array: [...arr],
          compared: [j, j + 1],
          swapped: true,
          description: `Swapped ${arr[j]} and ${arr[j + 1]}`
        };
      }
    }
  }

  yield {
    array: [...arr],
    compared: [],
    swapped: false,
    description: "Sorting complete!"
  };
}
```

```javascript
// Complexity analysis
export const bubbleSortInfo = {
  timeComplexity: {
    best: "O(n)",
    average: "O(n²)",
    worst: "O(n²)"
  },
  spaceComplexity: "O(1)",
  stable: true,
  inPlace: true
};
```

Algorithm Index File
```javascript
// src/features/algorithms/sorting/index.js
import { bubbleSort, bubbleSortInfo } from './bubbleSort';
import { quickSort, quickSortInfo } from './quickSort';
import { mergeSort, mergeSortInfo } from './mergeSort';

export const sortingAlgorithms = {
  bubbleSort: {
    function: bubbleSort,
    info: bubbleSortInfo,
    category: 'sorting',
    name: 'Bubble Sort'
  },
  quickSort: {
    function: quickSort,
    info: quickSortInfo,
    category: 'sorting',
    name: 'Quick Sort'
  },
  mergeSort: {
    function: mergeSort,
    info: mergeSortInfo,
    category: 'sorting',
    name: 'Merge Sort'
  }
};
```

🎨 UI Component with Animations

```jsx
// src/components/visualization/ArrayVisualizer.jsx
import { motion } from 'framer-motion';
import { useAlgorithm } from '../../hooks/useAlgorithm';

const ArrayVisualizer = ({ algorithm, inputArray }) => {
  const { currentStep, isPlaying, speed, play, pause, reset } = useAlgorithm(algorithm,
inputArray);

  return (
    <div className="w-full bg-white rounded-lg shadow-lg p-6">
      {/* Controls */}
      <div className="flex gap-4 mb-6">
        <button
          onClick={play}
          disabled={isPlaying}
          className="px-4 py-2 bg-green-500 text-white rounded-lg disabled:bg-gray-400"
        >
          Play
        </button>
        <button
          onClick={pause}
          disabled={!isPlaying}
          className="px-4 py-2 bg-yellow-500 text-white rounded-lg disabled:bg-gray-400"
        >
          Pause
        </button>
        <button
          onClick={reset}
          className="px-4 py-2 bg-red-500 text-white rounded-lg"
        >
          Reset
        </button>
      </div>

      {/* Visualization */}
      <div className="flex items-end justify-center h-64 gap-1">
        {currentStep?.array.map((value, index) => (
          <motion.div
            key={index}
            layout
            initial={{ opacity: 0, scale: 0.8 }}
            animate={{
```

```
            opacity: 1,
            scale: 1,
            height: `${value * 10}px`,
            backgroundColor: currentStep.compared.includes(index)
              ? '#ff4d4f'
              : currentStep.swapped && currentStep.compared.includes(index)
              ? '#52c41a'
              : '#1890ff'
          }}
          transition={{ type: 'spring', damping: 15 }}
          className="w-8 rounded-t-lg flex items-center justify-center text-white text-xs
font-medium"
        >
          {value}
        </motion.div>
      ))}
    </div>

    {/* Status */}
    {currentStep?.description && (
      <motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        className="mt-4 p-3 bg-gray-100 rounded-lg text-center"
      >
        {currentStep.description}
      </motion.div>
    )}
  </div>
 );
};

export default ArrayVisualizer;
```

📊 Database Structure

Firestore Collections
```javascript
// Users collection
users/{userId} = {
  uid: string,
  email: string,
  displayName: string,
```

```
  photoURL: string,
  createdAt: timestamp,
  lastLogin: timestamp
}

// User saved states
userStates/{stateId} = {
  userId: string,
  algorithm: string,
  inputData: any,
  currentStep: number,
  createdAt: timestamp,
  updatedAt: timestamp
}

// Algorithm metadata
algorithms/{algorithmId} = {
  name: string,
  category: string,
  timeComplexity: {
    best: string,
    average: string,
    worst: string
  },
  spaceComplexity: string,
  stable: boolean,
  inPlace: boolean
}
```

🚀 Deployment Strategy

1. Vercel Deployment
```bash
# Install Vercel CLI
npm i -g vercel

# Deploy
vercel --prod
```

2. Firebase Hosting
```bash
# Install Firebase CLI
```

npm install -g firebase-tools

Login to Firebase
firebase login

Initialize hosting
firebase init hosting

Deploy
firebase deploy
```

### 3. Environment Setup for Deployment
```bash
# Set environment variables in Vercel/Netlify dashboard
VITE_FIREBASE_API_KEY=your_api_key
VITE_FIREBASE_AUTH_DOMAIN=your_project.firebaseapp.com
VITE_FIREBASE_PROJECT_ID=your_project_id
VITE_FIREBASE_STORAGE_BUCKET=your_project.appspot.com
VITE_FIREBASE_MESSAGING_SENDER_ID=your_sender_id
VITE_FIREBASE_APP_ID=your_app_id
```

📝 Documentation

README.md Template
```markdown
# Algorithm Visualizer

A comprehensive algorithm visualization tool with 200+ algorithms.

## Features

- 🎯 Visualize 200+ algorithms
- 🎨 Beautiful animations with Framer Motion
- 💾 Save progress with Firebase
- 📱 Responsive design
- 🔍 Code explanation and complexity analysis

## Installation

1. Clone the repository
2. Install dependencies: `npm install`
3. Set up Firebase environment variables

4. Run development server: `npm run dev`

## Contributing

1. Fork the repository
2. Create a feature branch: `git checkout -b feature/amazing-feature`
3. Commit changes: `git commit -m 'Add amazing feature'`
4. Push to branch: `git push origin feature/amazing-feature`
5. Open a Pull Request
```

🔮 Future Expansion Plans

Plan A: Current Project
- Complete 200+ algorithm implementations
- Add user authentication and profiles
- Implement code sharing features
- Add algorithm comparisons

Plan B: AI-powered Code Interpreter
```javascript
// src/features/ai-code-interpreter/
import { GoogleGenerativeAI } from "@google/generative-ai";

const genAI = new GoogleGenerativeAI(import.meta.env.VITE_GEMINI_API_KEY);

export async function analyzeCode(codeSnippet) {
  const model = genAI.getGenerativeModel({ model: "gemini-pro" });

  const prompt = `
    Analyze this code and generate a visualization plan:
    ${codeSnippet}

    Return JSON with steps for visualization.
  `;

  const result = await model.generateContent(prompt);
  return JSON.parse(result.response.text());
}
```

Plan C: Mobile App
- Use React Native or Flutter
- Reuse business logic from web version
```

- Platform-specific UI components
- App store deployment guidelines

🎯 Git Strategy

Commit Convention
```bash
# Feature implementation
git commit -m "feat: add bubble sort visualization"

# Bug fix
git commit -m "fix: resolve array index issue"

# Documentation
git commit -m "docs: update README with installation steps"

# Performance
git commit -m "perf: optimize rendering performance"
```

Branch Strategy
```
main        → Production ready code
develop     → Integration branch
feature/*   → New features
bugfix/*    → Bug fixes
release/*   → Release preparation
```

📊 Progress Tracking

Initial 20 Algorithms

Category	Algorithms
Sorting	Bubble Sort, Quick Sort, Merge Sort, Insertion Sort, Selection Sort
Searching	Linear Search, Binary Search, DFS, BFS
Graph	Dijkstra, A*, Bellman-Ford, Floyd-Warshall
Dynamic Programming	Fibonacci, Knapsack, LCS
Pathfinding	A*, Dijkstra, BFS, DFS

This structure provides a solid foundation for your algorithm visualizer project. Each component is modular, scalable, and follows best practices. The project is set up for easy expansion and maintenance.

Sep 1, 2025 Added logo and changed the Header name with the respective Algorithm, and view the changes in the GitHub, you will get to know.
Added things the header code in the mainPage.jsx:

```
<header className="bg-white shadow-sm border-b p-4">
```

```
<div className="max-w-7xl mx-auto flex items-center justify-between">
            <div className="flex items-center gap-3">
              <div className="h-10 w-10">
                <svg viewBox="0 0 100 100" className="h-full w-full"
role="img" aria-label="AV logo">
                  <defs>
                    <linearGradient id="g1" x1="0" x2="1" y1="0" y2="1">
                      <stop offset="0" stopColor="#4f46e5" />
                      <stop offset="1" stopColor="#06b6d4" />
                    </linearGradient>
                  </defs>
                  <circle cx="50" cy="50" r="48" fill="url(#g1)" />
                  <text x="50" y="58" textAnchor="middle" fontWeight="700"
fontSize="44" fill="white" fontFamily="Inter, Arial, sans-serif">AV</text>
                </svg>
              </div>

              <h1 className="text-2xl font-bold text-gray-800">Algorithm
Visualizer</h1>
            </div>

            <div>
              <button
                onClick={handleLogout}
                className="bg-red-500 text-white px-4 py-2 rounded-lg
hover:bg-red-600"
              >
```

```
                Logout
              </button>
            </div>
          </div>
        </header>
```

Update this with the logo in the future:
Replace later: to swap the AV badge with a real image just replace the badge `<div>` / `<svg>` with:
`<img src="/path/to/logo.png" alt="Logo" className="h-10 w-10 rounded-full" />`

Here is the colors we need to use: ["#3d348b","#7678ed","#f7b801","#f18701","#f35b04"]

--tekhelet: #3d348bff;
--medium-slate-blue: #7678edff;
--selective-yellow: #f7b801ff;
--tangerine: #f18701ff;
--persimmon: #f35b04ff;

Sep 8, 2025 Updated the Side bar js file with UI UX styles and added some more algorithms into the sorting algorithms..


Rajendhar