MustSolve Project Documentation

Technical Specification and Onboarding Guide

1. Project Overview

MustSolve is a modern coding practice platform designed to provide real-time coding problem-solving with an engaging and collaborative interface.

Mission Statement: Empower developers and students to master algorithms and data structures through realistic execution environments and social competition.

Key Differentiators:

- Real Java code execution via Node.js backend (not simulation)
- Modern UI with Next.js, TypeScript, and Tailwind CSS
- Integrated social features for friend progress tracking

Target Audience: Computer science students, interview candidates, and programming enthusiasts.

2. Technology Stack

Frontend

Technology	Version	Rationale
Next.js	15.3.3	Chosen for its App Router architecture and excellent developer experience. Provides fast page loads via server-side rendering (SSR) and static site generation (SSG), which improves SEO and user engagement for the MustSolve coding platform.
React	19.0.0	Core UI library enabling reusable components and efficient rendering. Chosen for its strong ecosystem and ease of managing

		interactive, dynamic UI states for problem pages and real-time execution feedback.
TypeScript	5.x	Adds static typing for better maintainability and fewer runtime errors. Ideal for a growing project like MustSolve where strict type checking helps prevent breaking changes.
Tailwind CSS	4.x	Utility-first CSS framework for rapidly building responsive, modern UI without writing large custom CSS files. Speeds up development while ensuring consistent styling.
Framer Motion	12.18.1	Used for high-quality animations and smooth UI transitions, enhancing the user experience when navigating between problems and viewing execution results.
Lucide React	0.525.0	Provides a clean, customizable icon set that integrates easily with React, ensuring a professional and consistent design.
Supabase Auth	latest	Handles authentication with OAuth support. Chosen for its ease of integration with Next.js and real-time database syncing, which fits MustSolve's user account and progress tracking needs.

Backend

Technology	Version	Rationale
Node.js	18.x+	Chosen for its non-blocking, event-driven architecture that handles concurrent code execution requests efficiently.
Express.js	4.18.2	Minimal yet powerful web framework for building REST APIs. Handles MustSolve's Java execution requests, health checks, and authentication endpoints.
Java JDK	11+	Required for real Java code compilation and execution — a core feature that sets MustSolve apart from simulation-only platforms.
CORS	2.8.5	Ensures secure cross-origin requests between the frontend and backend while preventing unauthorized access.
Child Process Module	built-in	Used to spawn Java compilation and execution processes in an isolated, sandboxed environment with timeouts for security.

Development Tools

Technology	Version	Rationale
Git	latest	Version control system used for collaborative development and feature branching.
GitHub	latest	Code hosting and collaboration platform with

		integrated CI/CD workflows.
npm	latest	Package manager for installing and managing JavaScript and Node.js dependencies.
Vercel	latest	Hosting and deployment platform optimized for Next.js, enabling fast and reliable production builds.
Postman	latest	API testing tool for verifying backend endpoints before frontend integration.

3. Architecture Overview

The MustSolve architecture follows a client-server model:

- Frontend: Next.js app served to browser, interacting with backend via REST APIs.
- Backend: Node.js + Express server hosting Java execution pipeline.
- Database/Auth: Supabase manages users, sessions, and problem data.

Data Flow: User writes code in frontend editor -> Backend compiles/executes -> Returns results -> Frontend displays.

4. Frontend Implementation

- Next.js App Router for modular routing.
- Component-based architecture for reusability.
- State management via React hooks & context API.
- Tailwind CSS + Framer Motion for UI/UX.

5. Backend Implementation

- Handles code compilation with Java JDK.
- Execution sandbox with 5s timeout and memory limits.
- API endpoints for code execution and health checks.
- Error handling and temporary file cleanup.

6. File Structure & Detailed Breakdown

```
src/
├── app/
│   ├── friends/page.tsx
│   ├── login/page.tsx
│   ├── practice/page.tsx
│   ├── problems/[slug]/page.tsx
│   ├── globals.css
│   ├── layout.tsx
│   └── page.tsx
├── components/Navbar.tsx
├── contexts/AuthContext.tsx
├── data/problem.ts
└── lib/supabase.ts

mustsolve-backend/
├── server.js
└── package.json
```

- `src/app/problems/[slug]/page.tsx` - Core problem solving UI
- `mustsolve-backend/server.js` - Execution engine
- `src/contexts/AuthContext.tsx` - Authentication
- `src/data/problem.ts` - Problem definitions
- `src/app/practice/page.tsx` - Problem list browser
- `src/app/friends/page.tsx` - Social features

7. Core Features Implemented

- Java code execution pipeline with compile-run.
- Professional code editor with syntax highlighting.
- Test case management.
- OAuth login via Supabase.
- Progress tracking and friend leaderboard.
- Animated UI.

8. To-Be-Implemented Features

High Priority:

- AWS Lambda/ECS/RDS/S3 migration
- Docker sandbox for execution

Medium Priority:

- AI-powered coding assistance
- Advanced social & analytics

Future Vision:

- Multi-language support
- Enterprise-level tools

9. Setup & Development Guide

- Install Node.js, Java JDK, Git.
- Clone repo, run `npm install` for frontend and backend.
- Configure Supabase credentials in `.env` files.
- Run `npm run dev` for frontend, `node server.js` for backend.
- Deploy via Vercel (frontend) & AWS/Heroku (backend).

10. API Documentation

Base URL: `https://api.mustsolve.com`

POST `/api/execute-java` - Execute Java code

```
Request:
{
  "code": "public class Main { ... }",
  "input": "test input"
}

Response:
{
  "output": "expected output",
  "status": "success",
  "errors": null
}
```

GET `/api/health` - Check API status.

```
Response:
{
  "status": "ok",
  "uptime": 10234
}
```

11. Getting Started & Dependencies

Frontend Dependencies

bash

```
cd mustsolve

npm install
```

Backend Dependencies

bash

```
cd mustsolve-backend

npm init -y

npm install express cors

npm install -D nodemon
```

Java Verification

bash

```
java -version

javac -version
```

Start Development Servers

bash

```
# Terminal 1 - Backend

cd mustsolve-backend

npm run dev

# Terminal 2 - Frontend

cd mustsolve

npm run dev
```