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	App Router for scalability
React	19.0.0	Declarative UI framework
TypeScript	5.x	Static typing
Tailwind CSS	4.x	Utility-first CSS
Framer Motion	12.18.1	Animations
Lucide React	0.525.0	Icon set
Supabase	-	Auth & backend services

Backend:

Technology	Version	Rationale
Node.js	-	Event-driven backend
Express.js	4.18.2	Routing & APIs
Java JDK	11+	Compile & run Java code
CORS	2.8.5	Security for cross-origin requests

Development Tools:

Tool	Purpose
Git	Version control

VS Code	Development environment
Postman	API testing

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
```