🚀 CodeVerse — Complete System Architecture & Deployment Guide

> **Documentation for CodeVerse** covering architecture, networking, WebSockets, HMR, CORS, environment setup, and execution flow.

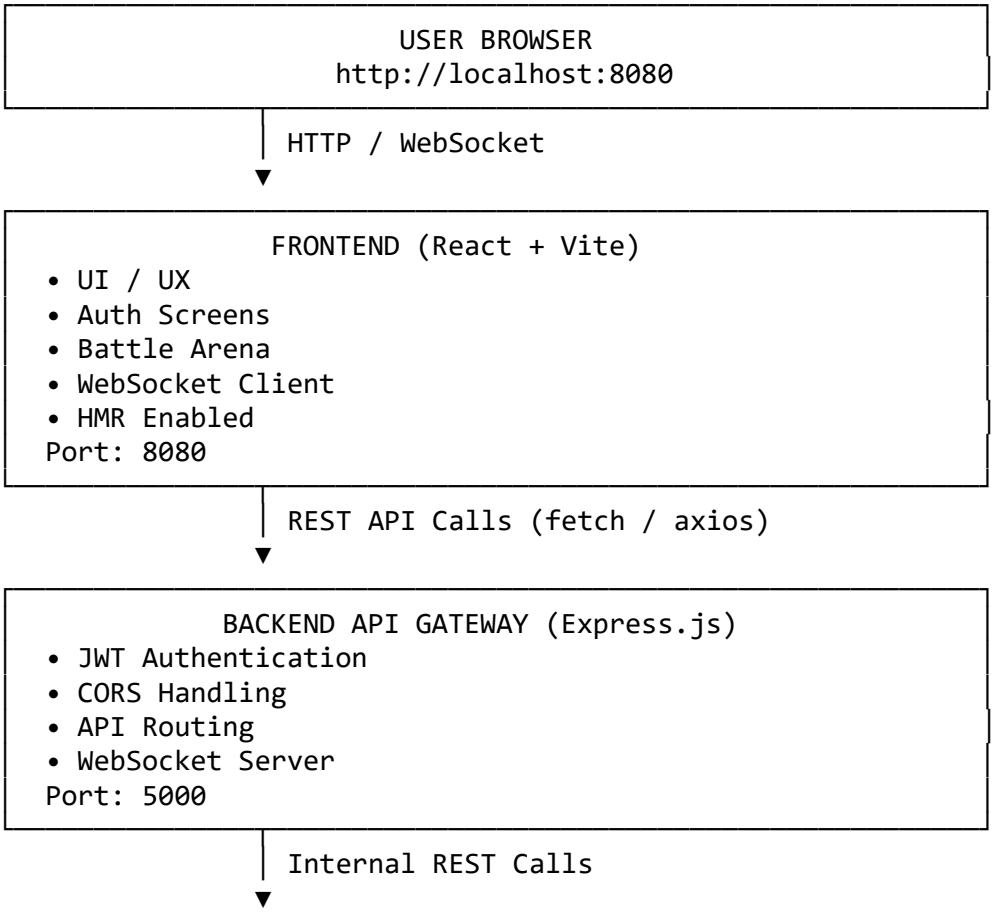
1️⃣ Introduction

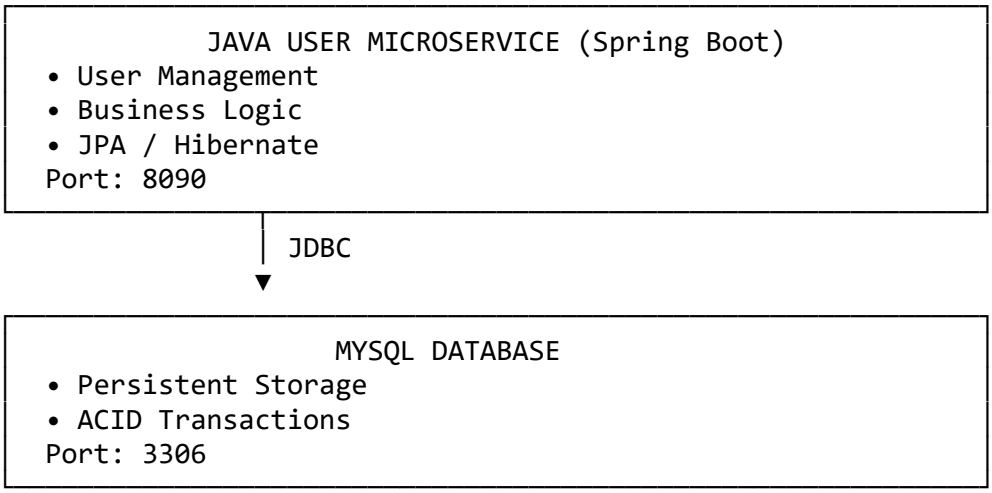
CodeVerse is a full-stack, microservices-based competitive coding platform designed for scalability, real-time interaction, and clean separation of concerns.

This document is intended for: - Developers - DevOps / Deployment teams - Academic reviewers - Internal technical audits

It explains **how CodeVerse works end-to-end**, how to **run it locally or on a network**, and how all services communicate.

2️⃣ High-Level System Architecture





---

3 Component Responsibilities

◇ Frontend (React + Vite)

Responsibilities: - User interface - Authentication forms - Battle UI - WebSocket client - Hot Module Replacement (HMR)

Runs on: `http://localhost:8080`

◇ Backend (Express.js API Gateway)

Responsibilities: - Central API gateway - JWT authentication - CORS enforcement - WebSocket server - Routing to microservices

Runs on: `http://localhost:5000`

◇ Java Microservice (Spring Boot)

Responsibilities: - User CRUD operations - Validation & persistence - Database abstraction layer

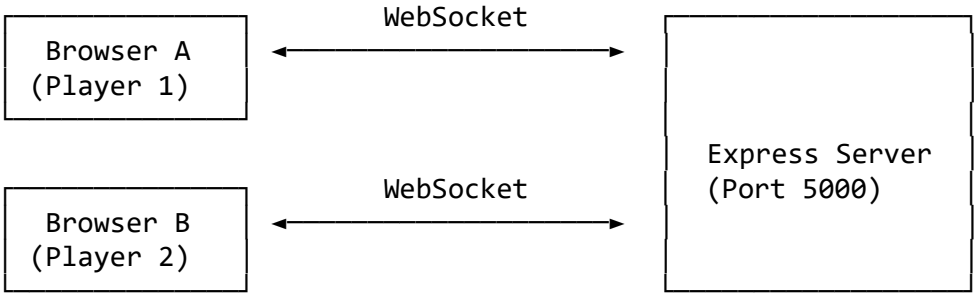
Runs on: `http://localhost:8090`

◇ Database (MySQL)

Responsibilities: - Store users, rooms, matches - Ensure data persistence

Runs on: `127.0.0.1:3306`

4 WebSocket Architecture (Real-Time Battles)



Used for: - Real-time code sync - Match state updates - Countdown timers - Live submissions

---

5 Request Flow (Flowchart)

User Signup Flow

```
User
 ↓
React Frontend
 ↓ POST /api/auth/signup
Express Backend
 ↓ REST
Java User Service
 ↓ JDBC
MySQL
 ↑
Java Service
 ↑
Express Backend (JWT issued)
 ↑
React Frontend (Token stored)
```

Battle Room Creation Flow

```
User clicks Create Room
 ↓
Frontend
 ↓ POST /api/match/room/create
Backend
 ↓ Database Insert
Room Created
 ↓
WebSocket Broadcast
```

```
 ↓
Other Players See Room
```

---

6 Network & Port Configuration

Service	Port	Protocol
Frontend	8080	HTTP / WS
Backend	5000	HTTP / WS
Java Service	8090	HTTP
MySQL	3306	TCP

---

7 Environment Variables & PATH Setup

Frontend (`.env.local`)

```
VITE_API_URL=http://localhost:5000/api
VITE_WS_URL=ws://localhost:5000
```

Backend (`server/.env`)

```
PORT=5000
CLIENT_URL=http://localhost:8080
DB_HOST=127.0.0.1
DB_PORT=3306
DB_USER= your_db_name
DB_PASSWORD= your_db_password
DB_NAME=codeverse
```

Java (`application.yml`)

```
spring.datasource.url=jdbc: mysql://127.0.0.1:3306/codeverse
spring.datasource.username= your_db_name
spring.datasource.password= your_db_password
server.port= 8090
```

---

8 Hot Module Replacement (HMR)

Vite HMR (Frontend)

Already enabled by default:

```
npm run dev
```

For network access:

```
// vite.config.ts
export default defineConfig({
```

```
  server: {
    host: true,
    port: 8080
  }
})
```

Now accessible via:

```
http://<LOCAL-IP>:8080
```

---

9 CORS Configuration (Backend)

```
app.use(cors({
  origin: 'http://localhost:8080',
  credentials: true
}));
```

For network:

```
origin: ['http://localhost:8080', 'http://192.168.x.x:8080']
```

---

10 Running Over Local Network

1. Get your IP address
2. Update frontend `.env.local`
3. Update backend CORS
4. Restart all services

Users on the same Wi-Fi can now access CodeVerse.

---

11 Startup Order (IMPORTANT)

1. MySQL
2. Java Microservice
3. Express Backend
4. React Frontend

---

12 Production Notes

- Use Nginx as reverse proxy
- Enable SSL
- Use non-root DB user
- Disable Vite dev server

☑ Final Status

✓ Clean architecture ✓ Real-time ready ✓ Network-enabled ✓ Production scalable ✓ Academic & industry ready

---

Project: CodeVerse
Architecture: Microservices + WebSockets
Documentation Level: Enterprise / Academic