Multi-Agent Codex Collaboration — Full Bundle

This repository bundles a **Node.js multi-agent backend**, a **Vue 3 realtime frontend**, Docker/Compose deployment, and a CLI setup script for Ubuntu 24 / Fedora. It integrates OpenAI (SDK) and optional Codex CLI, supports RBAC + OAuth SSO, realtime code playground, chat with agents, force-directed graph, and a sandbox preview.

How to use

```
# 1) Clone and enter
git clone <your-repo> multi-agent-codex
cd multi-agent-codex

# 2) Create .env files from examples
cp .env.example .env
cp apps/server/.env.example apps/server/.env
cp apps/web/.env.example apps/web/.env

# 3) Start with Docker
docker compose up --build

# 4) Open the app
# Frontend: http://localhost:5173   (Vite dev)  or http://localhost:
8080 via Nginx in compose-prod
# API:      http://localhost:3000
```

Ubuntu/Fedora bare-metal: run `scripts/setup.sh` then `pnpm i` and `pnpm dev`.

Repository Structure

```
multi-agent-codex/
├── .env.example
├── docker-compose.yml
├── docker-compose.prod.yml
├── Dockerfile.server
├── Dockerfile.web
├── nginx/
│   └── default.conf
├── scripts/
│   ├── setup.sh
│   └── seed-demo.sh
├── packages/
```

```
│   └── shared/
│       ├── package.json
│       └── src/types.ts
├── apps/
│   ├── server/
│   │   ├── package.json
│   │   ├── .env.example
│   │   └── src/
│   │       ├── index.ts
│   │       ├── config.ts
│   │       ├── auth/
│   │       │   ├── auth.ts
│   │       │   └── rbac.ts
│   │       ├── agents/
│   │       │   ├── orchestrator.ts
│   │       │   ├── roles/
│   │       │   │   ├── planner.ts
│   │       │   │   ├── coder.ts
│   │       │   │   └── tester.ts
│   │       │   └── providers/
│   │       │       ├── openai.ts
│   │       │       └── codexCli.ts
│   │       ├── routes/
│   │       │   ├── projects.ts
│   │       │   ├── chat.ts
│   │       │   └── auth.ts
│   │       ├── services/
│   │       │   ├── projects.ts
│   │       │   ├── messages.ts
│   │       │   └── graph.ts
│   │       ├── sockets/
│   │       │   └── io.ts
│   │       └── util/
│   │           ├── fsx.ts
│   │           └── logger.ts
│   └── web/
│       ├── package.json
│       ├── index.html
│       ├── vite.config.ts
│       └── src/
│           ├── main.ts
│           ├── app.d.ts
│           ├── router.ts
│           ├── store/
│           │   ├── index.ts
│           │   ├── auth.ts
```

```
│           │   └── project.ts
│           ├── api/
│           │   └── client.ts
│           ├── components/
│           │   ├── ChatPanel.vue
│           │   ├── CodeEditors.vue
│           │   ├── PreviewPane.vue
│           │   ├── GraphView.vue
│           │   ├── ForceFieldBg.vue
│           │   └── TopBar.vue
│           └── views/
│               ├── LoginView.vue
│               ├── DashboardView.vue
│               └── ProjectView.vue
└── README.md
```

Root `.env.example`

```
# Shared
OPENAI_API_KEY=
# Optional: OpenAI base URL / organization
OPENAI_BASE_URL=
OPENAI_ORG=

# OAuth SSO (example using Auth0 or any OIDC provider)
OIDC_ISSUER_URL=
OIDC_CLIENT_ID=
OIDC_CLIENT_SECRET=
OIDC_CALLBACK_URL=http://localhost:3000/api/auth/callback/oidc
SESSION_SECRET=change-me-please

# Storage paths
DATA_DIR=./data
PROJECTS_DIR=./data/projects
LOGS_DIR=./data/logs

# CORS
WEB_ORIGIN=http://localhost:5173
```

`docker-compose.yml` (dev)

```yaml
version: "3.8"
services:
  server:
    build:
      context: .
      dockerfile: Dockerfile.server
    env_file:
      - .env
      - apps/server/.env
    ports:
      - "3000:3000"
    volumes:
      - ./apps/server:/app/apps/server
      - ./packages/shared:/app/packages/shared
      - ./data:/app/data
    command: ["pnpm","--filter","@app/server","dev"]
    depends_on: []

  web:
    build:
      context: .
      dockerfile: Dockerfile.web
    env_file:
      - .env
      - apps/web/.env
    ports:
      - "5173:5173"
    volumes:
      - ./apps/web:/app/apps/web
      - ./packages/shared:/app/packages/shared
    command: ["pnpm","--filter","@app/web","dev"]
    depends_on:
      - server
```

`docker-compose.prod.yml` (Nginx)

```yaml
version: "3.8"
services:
  server:
    build:
      context: .
```

```
      dockerfile: Dockerfile.server
    env_file:
      - .env
      - apps/server/.env
    volumes:
      - ./data:/app/data
    expose:
      - "3000"

  web:
    build:
      context: .
      dockerfile: Dockerfile.web
    env_file:
      - .env
      - apps/web/.env
    expose:
      - "4173"   # preview server

  nginx:
    image: nginx:stable-alpine
    volumes:
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf:ro
    ports:
      - "8080:80"
    depends_on:
      - server
      - web
```

---

nginx/default.conf

```
server {
  listen 80;
  server_name _;

  # Frontend (built preview) proxied to Vite preview or static if you export
  location / {
    proxy_pass http://web:4173/;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
  }

  # API
  location /api/ {
```

```
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_pass http://server:3000$request_uri;
  }

  # Websocket
  location /socket.io/ {
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_pass http://server:3000$request_uri;
  }
}
```

Dockerfile.server

```
FROM node:20-bullseye
WORKDIR /app

# System deps for codex CLI optional utils
RUN apt-get update && apt-get install -y git ripgrep bash && rm -rf /var/lib/
apt/lists/*

COPY package.json pnpm-lock.yaml .npmrc* ./
COPY packages ./packages
COPY apps/server ./apps/server

# Use pnpm for workspace installs
RUN npm i -g pnpm@9 && pnpm i --frozen-lockfile

EXPOSE 3000
CMD ["pnpm","--filter","@app/server","start"]
```

Dockerfile.web

```
FROM node:20-alpine
WORKDIR /app

COPY package.json pnpm-lock.yaml .npmrc* ./
```

```
COPY packages ./packages
COPY apps/web ./apps/web

RUN npm i -g pnpm@9 && pnpm i --frozen-lockfile

# Dev runs Vite on 5173; prod uses preview on 4173
EXPOSE 5173 4173
CMD ["pnpm","--filter","@app/web","dev"]
```

Root `package.json` (pnpm workspace)

```
{
  "name": "multi-agent-codex",
  "private": true,
  "packageManager": "pnpm@9.6.0",
  "workspaces": [
    "apps/*",
    "packages/*"
  ],
  "scripts": {
    "dev": "pnpm -r --parallel dev",
    "build": "pnpm -r build",
    "start": "pnpm --filter @app/server start",
    "lint": "eslint ."
  }
}
```

`packages/shared/src/types.ts`

```
export type Role = "user" | "admin";

export interface User {
  id: string;
  name: string;
  email: string;
  image?: string;
  role: Role;
}

export interface ProjectFile { id: string; name: string; path: string; content:
string; }
```

```ts
export interface Project {
  id: string;
  ownerId: string;
  name: string;
  createdAt: string;
  updatedAt: string;
  files: ProjectFile[];
}

export interface ChatMessage {
  id: string;
  projectId: string;
  author: "user" | "agent";
  agentRole?: "planner" | "coder" | "tester" | "orchestrator";
  userId?: string;
  content: string;
  createdAt: string;
}

export interface AgentRunEvent {
  type: "start" | "token" | "finish" | "error" | "file-change" | "graph";
  runId: string;
  data?: any;
}
```

Backend — `apps/server`

`apps/server/package.json`

```json
{
  "name": "@app/server",
  "version": "0.1.0",
  "type": "module",
  "main": "dist/index.js",
  "scripts": {
    "dev": "tsx src/index.ts",
    "build": "tsup src/index.ts --dts --format esm,cjs --out-dir dist",
    "start": "node dist/index.js"
  },
  "dependencies": {
    "express": "^4.19.2",
    "cors": "^2.8.5",
    "cookie-parser": "^1.4.6",
    "express-session": "^1.17.3",
```

```
    "passport": "^0.7.0",
    "passport-openidconnect": "^0.1.2",
    "socket.io": "^4.7.5",
    "openai": "^4.53.0",
    "nanoid": "^5.0.7",
    "zod": "^3.23.8",
    "fs-extra": "^11.2.0",
    "yaml": "^2.4.2"
  },
  "devDependencies": {
    "tsup": "^8.0.2",
    "tsx": "^4.15.7",
    "typescript": "^5.5.4"
  }
}
```

apps/server/src/config.ts

```
export const cfg = {
  port: Number(process.env.PORT || 3000),
  webOrigin: process.env.WEB_ORIGIN || "http://localhost:5173",
  dataDir: process.env.DATA_DIR || "./data",
  projectsDir: process.env.PROJECTS_DIR || "./data/projects",
  logsDir: process.env.LOGS_DIR || "./data/logs",
  openai: {
    apiKey: process.env.OPENAI_API_KEY || "",
    baseUrl: process.env.OPENAI_BASE_URL,
    organization: process.env.OPENAI_ORG
  },
  sessionSecret: process.env.SESSION_SECRET || "dev-secret",
  oidc: {
    issuer: process.env.OIDC_ISSUER_URL,
    clientId: process.env.OIDC_CLIENT_ID,
    clientSecret: process.env.OIDC_CLIENT_SECRET,
    callbackURL: process.env.OIDC_CALLBACK_URL || "http://localhost:3000/api/
auth/callback/oidc"
  }
};
```

apps/server/src/util/logger.ts

```
export function log(...args: any[]) { console.log(new Date().toISOString(),
"[srv]", ...args); }
export function warn(...args: any[]) { console.warn(new Date().toISOString(),
"[srv]", ...args); }
```

```
export function error(...args: any[]) { console.error(new Date().toISOString(),
"[srv]", ...args); }
```

apps/server/src/util/fsx.ts

```
import fs from "fs-extra";
import path from "path";
import { cfg } from "../config.js";

export async function ensureDirs() {
  await fs.ensureDir(cfg.projectsDir);
  await fs.ensureDir(cfg.logsDir);
}

export function projectPath(id: string) { return path.join(cfg.projectsDir,
id); }
export function projectFilePath(id: string, rel: string) { return
path.join(projectPath(id), rel); }
```

apps/server/src/auth/rbac.ts

```
import type { User } from "@shared/src/types";

export function canReadProject(user: User, ownerId: string) {
  return user.role === "admin" || user.id === ownerId;
}
export function canWriteProject(user: User, ownerId: string) {
  return user.role === "admin" || user.id === ownerId;
}
```

apps/server/src/auth/auth.ts

```
import session from "express-session";
import passport from "passport";
import { Strategy as OIDCStrategy } from "passport-openidconnect";
import type { Express } from "express";
import { cfg } from "../config.js";
import { nanoid } from "nanoid";
import type { User } from "@shared/src/types";

const users = new Map<string, User>();

export function setupAuth(app: Express) {
  app.use(session({
```

```
    secret: cfg.sessionSecret,
    resave: false,
    saveUninitialized: false
  }));
  app.use(passport.initialize());
  app.use(passport.session());

  if (cfg.oidc.issuer && cfg.oidc.clientId && cfg.oidc.clientSecret) {
    passport.use("oidc", new OIDCStrategy({
      issuer: cfg.oidc.issuer!,
      authorizationURL: cfg.oidc.issuer! + "/authorize",
      tokenURL: cfg.oidc.issuer! + "/oauth/token",
      userInfoURL: cfg.oidc.issuer! + "/userinfo",
      clientID: cfg.oidc.clientId!,
      clientSecret: cfg.oidc.clientSecret!,
      callbackURL: cfg.oidc.callbackURL,
      scope: "openid profile email"
    }, (issuer, profile, context, idToken, accessToken, refreshToken, done) => {
      const id = profile.id || nanoid();
      const u: User = {
        id,
        name: (profile as any).displayName || profile._json?.name || "User",
        email: (profile as any).emails?.[0]?.value || profile._json?.email || `${id}@example.com`,
        image: (profile as any).photos?.[0]?.value,
        role: "user"
      };
      users.set(id, u);
      return done(null, u);
    }));
  }

  passport.serializeUser((user: any, done) => done(null, user.id));
  passport.deserializeUser((id: string, done) => done(null, users.get(id) || null));
}

export function requireAuth(req: any, res: any, next: any) {
  if (req.isAuthenticated && req.isAuthenticated()) return next();
  res.status(401).json({ error: "unauthorized" });
}

export function getUser(req: any) { return req.user as User; }
```

apps/server/src/agents/providers/openai.ts

```
import OpenAI from "openai";
import { cfg } from "../../config.js";

export const openai = new OpenAI({
  apiKey: cfg.openai.apiKey,
  baseURL: cfg.openai.baseUrl,
  organization: cfg.openai.organization
});

export async function chat(messages:
OpenAI.Chat.Completions.ChatCompletionMessageParam[], opts?:
Partial<OpenAI.Chat.Completions.ChatCompletionCreateParams>) {
  const completion = await openai.chat.completions.create({
    model: "gpt-4o-mini",
    temperature: 0.2,
    max_tokens: 1000,
    messages,
    ...opts
  });
  return completion.choices[0]?.message?.content || "";
}
```

apps/server/src/agents/providers/codexCli.ts

```
import { spawn } from "node:child_process";
import { log } from "../../util/logger.js";

export interface CodexEvent { type: string; data?: any }

export function runCodex(command: string, cwd: string, onEvent: (e: CodexEvent)
=> void) {
  // Requires @openai/codex CLI installed inside container/VM
  const proc = spawn("codex", ["--approval-mode", "full-auto", command], {
cwd });
  proc.stdout.on("data", d => onEvent({ type: "stdout", data: d.toString() }));
  proc.stderr.on("data", d => onEvent({ type: "stderr", data: d.toString() }));
  proc.on("close", code => { log("codex exited", code); onEvent({ type:
"close", data: code }); });
  return proc;
}
```

Role Agents — `planner.ts`, `coder.ts`, `tester.ts`

```ts
// planner.ts
import { chat } from "../providers/openai.js";

export async function planTask(context: string, goal: string) {
  const content = await chat([
    { role: "system", content:
"You are Planner, a software architect. Produce a clear, numbered plan. Do not
write code." },
    { role: "user", content: `Context:\n${context}\n\nGoal: ${goal}` }
  ], { temperature: 0.1 });
  return content;
}
```

```ts
// coder.ts
import { chat } from "../providers/openai.js";

export async function writeCode(context: string, plan: string, fileHints:
string[]) {
  const content = await chat([
    { role: "system", content:
"You are Coder, a precise senior engineer. Modify or create files as needed.
Return unified diffs (patch) when possible." },
    { role: "user", content: `Plan:\n${plan}\n\nExisting context:\n${context}
\n\nTarget files: ${fileHints.join(", ")}` }
  ], { temperature: 0.2, max_tokens: 2000 });
  return content;
}
```

```ts
// tester.ts
import { chat } from "../providers/openai.js";

export async function testAndReview(context: string, recentChanges: string) {
  const content = await chat([
    { role: "system", content: "You are Tester/Reviewer. Write or update tests.
Point out defects. Return a concise report with actionable steps." },
    { role: "user", content: `Recent changes (diffs):\n${recentChanges}
\n\nCurrent code context:\n${context}` }
  ], { temperature: 0.2 });
  return content;
}
```

apps/server/src/agents/orchestrator.ts

```
import { nanoid } from "nanoid";
import * as fs from "fs-extra";
import path from "path";
import { planTask } from "./roles/planner.js";
import { writeCode } from "./roles/coder.js";
import { testAndReview } from "./roles/tester.js";
import { projectPath } from "../util/fsx.js";
import type { AgentRunEvent } from "@shared/src/types";

export interface OrchestratorOptions { projectId: string; goal: string; ioEmit:
(ev: AgentRunEvent) => void; }

async function readContext(dir: string) {
  const files = await fs.readdir(dir);
  const texts: string[] = [];
  for (const f of files) {
    const fp = path.join(dir, f);
    const stat = await fs.stat(fp);
    if (stat.isFile() && /\.(js|ts|vue|html|css|json)$/i.test(f)) {
      const content = await fs.readFile(fp, "utf8");
      texts.push(`// FILE: ${f}\n${content}`);
    }
  }
  return texts.join("\n\n");
}

export async function runOrchestration(opts: OrchestratorOptions) {
  const runId = nanoid();
  const dir = projectPath(opts.projectId);
  const ctx = await readContext(dir);
  opts.ioEmit({ type: "start", runId, data: { goal: opts.goal } });

  // 1) Plan
  const plan = await planTask(ctx, opts.goal);
  opts.ioEmit({ type: "token", runId, data: { agent: "planner", text: plan } });

  // 2) Code
  const codeReply = await writeCode(ctx, plan, ["index.html", "main.js",
"style.css"]);
  opts.ioEmit({ type: "token", runId, data: { agent: "coder", text:
codeReply } });

  // Apply simple patches if unified diffs provided (basic heuristic)
  await fs.writeFile(path.join(dir, `AI_PATCH_${Date.now()}.txt`), codeReply,
"utf8");
```

```
  // 3) Test/Review
  const review = await testAndReview(await readContext(dir), codeReply);
  opts.ioEmit({ type: "token", runId, data: { agent: "tester", text:
review } });

  opts.ioEmit({ type: "finish", runId });
}
```

REST & WebSocket — `index.ts`

```
import express from "express";
import http from "http";
import cors from "cors";
import cookieParser from "cookie-parser";
import { Server as IOServer } from "socket.io";
import { cfg } from "./config.js";
import { setupAuth } from "./auth/auth.js";
import { registerAuthRoutes } from "./routes/auth.js";
import { registerProjectRoutes } from "./routes/projects.js";
import { registerChatRoutes } from "./routes/chat.js";
import { setupIo } from "./sockets/io.js";
import { ensureDirs } from "./util/fsx.js";
import { log } from "./util/logger.js";

const app = express();
const server = http.createServer(app);
const io = new IOServer(server, { cors: { origin: cfg.webOrigin, credentials:
true } });
setupIo(io);

app.use(cors({ origin: cfg.webOrigin, credentials: true }));
app.use(express.json({ limit: "2mb" }));
app.use(cookieParser());
setupAuth(app);

registerAuthRoutes(app);
registerProjectRoutes(app, io);
registerChatRoutes(app, io);

app.get("/healthz", (_req, res) => res.json({ ok: true }));

await ensureDirs();
server.listen(cfg.port, () => log(`server listening on :${cfg.port}`));
```

routes/auth.ts

```
import type { Express } from "express";
import passport from "passport";
import { getUser } from "../auth/auth.js";

export function registerAuthRoutes(app: Express) {
  app.get("/api/auth/login", passport.authenticate("oidc"));
  app.get("/api/auth/callback/oidc", passport.authenticate("oidc", {
failureRedirect: "/?login=failed" }), (_req, res) => res.redirect("/") );
  app.post("/api/auth/logout", (req, res) => { req.logout(() => {}); res.json({
ok: true }); });
  app.get("/api/auth/me", (req, res) => { res.json({ user: getUser(req) ||
null }); });
}
```

routes/projects.ts

```
import type { Express } from "express";
import type { Server as IOServer } from "socket.io";
import { nanoid } from "nanoid";
import * as fs from "fs-extra";
import path from "path";
import { requireAuth, getUser } from "../auth/auth.js";
import { cfg } from "../config.js";
import { canReadProject, canWriteProject } from "../auth/rbac.js";
import { runOrchestration } from "../agents/orchestrator.js";
import { projectPath, projectFilePath } from "../util/fsx.js";

export function registerProjectRoutes(app: Express, io: IOServer) {
  app.get("/api/projects", requireAuth, async (req, res) => {
    const user = getUser(req)!;
    const entries = await fs.readdir(cfg.projectsDir);
    const out: any[] = [];
    for (const id of entries) {
      const metaPath = path.join(cfg.projectsDir, id, "project.json");
      if (await fs.pathExists(metaPath)) {
        const meta = await fs.readJSON(metaPath);
        if (canReadProject(user, meta.ownerId)) out.push(meta);
      }
    }
    res.json(out);
  });

  app.post("/api/projects", requireAuth, async (req, res) => {
    const user = getUser(req)!;
```

```
    const id = nanoid();
    const dir = projectPath(id);
    await fs.ensureDir(dir);
    const meta = { id, ownerId: user.id, name: req.body?.name || "Untitled",
createdAt: new Date().toISOString(), updatedAt: new Date().toISOString() };
    await fs.writeJSON(path.join(dir, "project.json"), meta, { spaces: 2 });
    // starter files
    await fs.writeFile(projectFilePath(id, "index.html"), "<!doctype
html>\n<html><head><meta charset=\"utf-8\"><title>Playground</title></
head><body><script src=\"main.js\"></script></body></html>");
    await fs.writeFile(projectFilePath(id, "main.js"), "console.log('hello');");
    await fs.writeFile(projectFilePath(id, "style.css"), "body{font-family:sans-
serif}");
    res.json(meta);
  });

  app.get("/api/projects/:id", requireAuth, async (req, res) => {
    const id = req.params.id;
    const meta = await fs.readJSON(path.join(cfg.projectsDir, id,
"project.json"));
    const user = getUser(req)!;
    if (!canReadProject(user, meta.ownerId)) return res.status(403).json({
error: "forbidden" });
    const files = await fs.readdir(projectPath(id));
    const outFiles: any[] = [];
    for (const f of files) {
      if (["project.json"].includes(f)) continue;
      const p = projectFilePath(id, f);
      const content = await fs.readFile(p, "utf8");
      outFiles.push({ id: f, name: f, path: f, content });
    }
    res.json({ ...meta, files: outFiles });
  });

  app.put("/api/projects/:id/files/:file", requireAuth, async (req, res) => {
    const id = req.params.id; const file = req.params.file;
    const meta = await fs.readJSON(path.join(cfg.projectsDir, id,
"project.json"));
    const user = getUser(req)!;
    if (!canWriteProject(user, meta.ownerId)) return res.status(403).json({
error: "forbidden" });
    await fs.writeFile(projectFilePath(id, file), req.body.content || "");
    meta.updatedAt = new Date().toISOString();
    await fs.writeJSON(path.join(cfg.projectsDir, id, "project.json"), meta, {
spaces: 2 });
    io.to(id).emit("file:update", { file, content: req.body.content });
    res.json({ ok: true });
  });
```

```
  app.post("/api/projects/:id/orchestrate", requireAuth, async (req, res) => {
    const id = req.params.id; const goal = req.body.goal || "";
    const meta = await fs.readJSON(path.join(cfg.projectsDir, id,
"project.json"));
    const user = getUser(req)!;
    if (!canWriteProject(user, meta.ownerId)) return res.status(403).json({
error: "forbidden" });

    runOrchestration({ projectId: id, goal, ioEmit: (ev) =>
io.to(id).emit("agent:event", ev) })
      .catch(err => io.to(id).emit("agent:event", { type: "error", runId: "n/
a", data: String(err) }));

    res.json({ started: true });
  });
}
```

routes/chat.ts

```
import type { Express } from "express";
import type { Server as IOServer } from "socket.io";
import { getUser, requireAuth } from "../auth/auth.js";
import { chat } from "../agents/providers/openai.js";

export function registerChatRoutes(app: Express, io: IOServer) {
  app.post("/api/chat", requireAuth, async (req, res) => {
    const user = getUser(req)!;
    const { projectId, message, agent } = req.body;
    const sys = agent === "planner"
      ? "You are Planner. Make a short plan."
      : agent === "tester"
      ? "You are Tester. Provide concise defects and tests."
      : "You are Coder. Provide concrete code suggestions.";

    const content = await chat([
      { role: "system", content: sys },
      { role: "user", content: message }
    ]);

    io.to(projectId).emit("chat:message", {
      id: Date.now().toString(),
      projectId,
      author: "agent",
      agentRole: agent || "orchestrator",
      content,
```

```
      createdAt: new Date().toISOString()
    });

    res.json({ content });
  });
}
```

`sockets/io.ts`

```
import type { Server } from "socket.io";
import { log } from "../util/logger.js";

export function setupIo(io: Server) {
  io.on("connection", socket => {
    log("socket connected", socket.id);
    socket.on("join:project", (projectId: string) => {
      socket.join(projectId);
      log("joined", projectId);
    });
  });
}
```

Frontend — `apps/web`

`apps/web/package.json`

```
{
  "name": "@app/web",
  "version": "0.1.0",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview --port 4173"
  },
  "dependencies": {
    "axios": "^1.7.2",
    "pinia": "^2.1.7",
    "socket.io-client": "^4.7.5",
    "vue": "^3.4.31",
    "monaco-editor": "^0.49.0"
  },
```

```
  "devDependencies": {
    "@types/node": "^20.14.10",
    "typescript": "^5.5.4",
    "vite": "^5.3.3",
    "vue-tsc": "^2.0.21"
  }
}
```

apps/web/.env.example

```
VITE_API_BASE=http://localhost:3000
```

apps/web/src/api/client.ts

```
import axios from "axios";
export const api = axios.create({ baseURL: import.meta.env.VITE_API_BASE,
withCredentials: true });
```

apps/web/src/store/index.ts

```
import { createPinia } from "pinia";
export const pinia = createPinia();
```

apps/web/src/store/auth.ts

```
import { defineStore } from "pinia";
import { api } from "../api/client";

export const useAuth = defineStore("auth", {
  state: () => ({ user: null as any }),
  actions: {
    async load() {
      const { data } = await api.get("/api/auth/me");
      this.user = data.user;
    },
    login() { window.location.href = `${import.meta.env.VITE_API_BASE}/api/auth/
login`; },
    async logout() { await api.post("/api/auth/logout"); this.user = null; }
  }
});
```

apps/web/src/store/project.ts

```
import { defineStore } from "pinia";
import { api } from "../api/client";
import { io } from "socket.io-client";

export const useProject = defineStore("project", {
  state: () => ({
    list: [] as any[],
    current: null as any,
    files: new Map<string,string>(),
    socket: null as any,
    messages: [] as any[]
  }),
  actions: {
    async loadList() { const { data } = await api.get("/api/projects");
this.list = data; },
    async create(name: string) { const { data } = await api.post("/api/
projects", { name }); this.list.push(data); return data; },
    async open(id: string) {
      const { data } = await api.get(`/api/projects/${id}`);
      this.current = data;
      this.files.clear();
      for (const f of data.files) this.files.set(f.name, f.content);
      // connect socket
      if (!this.socket) this.socket = io(import.meta.env.VITE_API_BASE, {
withCredentials: true });
      this.socket.emit("join:project", id);
      this.socket.on("file:update", ({ file, content }: any) =>
this.files.set(file, content));
      this.socket.on("chat:message", (m: any) => this.messages.push(m));
      this.socket.on("agent:event", (ev: any) => this.messages.push({ author:
"agent", agentRole: ev.data?.agent || "orchestrator", content:
JSON.stringify(ev, null, 2) }));
    },
    getFile(name: string) { return this.files.get(name) || ""; },
    async saveFile(name: string, content: string) {
      if (!this.current) return;
      await api.put(`/api/projects/${this.current.id}/files/${name}`, {
content });
      this.files.set(name, content);
    },
    async orchestrate(goal: string) {
      if (!this.current) return;
      await api.post(`/api/projects/${this.current.id}/orchestrate`, { goal });
    },
    async sendChat(agent: string, message: string) {
```

```
      if (!this.current) return;
      const { data } = await api.post(`/api/chat`, { projectId:
this.current.id, agent, message });
      this.messages.push({ author: "agent", agentRole: agent, content:
data.content, createdAt: new Date().toISOString() });
    }
  }
});
```

apps/web/src/main.ts

```
import { createApp } from "vue";
import { pinia } from "./store";
import router from "./router";
import App from "./App.vue";

createApp(App).use(pinia).use(router).mount("#app");
```

apps/web/src/router.ts

```
import { createRouter, createWebHistory } from "vue-router";
import LoginView from "./views/LoginView.vue";
import DashboardView from "./views/DashboardView.vue";
import ProjectView from "./views/ProjectView.vue";

const router = createRouter({
  history: createWebHistory(),
  routes: [
    { path: "/", component: DashboardView },
    { path: "/login", component: LoginView },
    { path: "/project/:id", component: ProjectView }
  ]
});
export default router;
```

apps/web/src/components/TopBar.vue

```
<template>
  <header class="top">
    <div class="brand">⚡ Multi-Agent Codex</div>
    <div class="spacer"/>
    <button v-if="!auth.user" @click="auth.login">SSO Login</button>
    <div v-else class="user">
      <img v-if="auth.user.image" :src="auth.user.image"/>
```

```
      <span>{{auth.user.name}}</span>
      <button @click="auth.logout">Logout</button>
    </div>
  </header>
</template>
<script setup lang="ts">
import { useAuth } from "../store/auth";
const auth = useAuth();
</script>
<style scoped>
.top{display:flex;align-items:center;padding:8px
12px;background:#0f172a;color:#e2e8f0}
.brand{font-weight:600}
.spacer{flex:1}
.user{display:flex;gap:8px;align-items:center}
.user img{width:28px;height:28px;border-radius:50%}
button{background:#1e293b;color:#e2e8f0;border:none;border-radius:6px;padding:
6px 10px;cursor:pointer}
button:hover{background:#334155}
</style>
```

apps/web/src/components/ChatPanel.vue

```
<template>
  <div class="chat">
    <div class="messages">
      <div v-for="m in project.messages" :key="m.id || Math.random()"
class="msg" :class="m.author">
        <div class="meta">{{ m.agentRole || m.author }}</div>
        <pre>{{ m.content }}</pre>
      </div>
    </div>
    <div class="composer">
      <select v-model="agent">
        <option value="orchestrator">orchestrator</option>
        <option value="planner">planner</option>
        <option value="coder">coder</option>
        <option value="tester">tester</option>
      </select>
      <input v-model="text" @keydown.enter="send" placeholder="Ask an agent…"/>
      <button @click="send">Send</button>
    </div>
  </div>
</template>
<script setup lang="ts">
import { ref } from "vue";
```

```
import { useProject } from "../store/project";
const project = useProject();
const text = ref("");
const agent = ref("orchestrator");
function send(){ if(!text.value) return; project.sendChat(agent.value,
text.value); text.value=""; }
</script>
<style scoped>
.chat{display:flex;flex-direction:column;height:100%}
.messages{flex:1;overflow:auto;padding:8px;gap:8px;display:flex;flex-
direction:column;background:#0b1120}
.msg{background:#111827;color:#e5e7eb;border-radius:8px;padding:8px}
.msg.user{background:#1f2937}
.meta{font-size:12px;opacity:.7;margin-bottom:4px}
pre{white-space:pre-wrap;font-family:ui-monospace,SFMono-
Regular,Menlo,monospace}
.composer{display:flex;gap:8px;padding:8px;background:#0f172a}
input,select{flex:1;background:#111827;color:#e5e7eb;border:none;border-radius:
6px;padding:8px}
button{background:#1e293b;color:#e5e7eb;border:none;border-radius:6px;padding:
8px 12px}
button:hover{background:#334155}
</style>
```

apps/web/src/components/CodeEditors.vue

```
<template>
  <div class="wrap">
    <div class="tabs">
      <button v-for="f in files" :key="f" :class="{active:f===active}"
@click="open(f)">{{f}}</button>
      <div class="spacer"/>
      <button @click="save">Save</button>
    </div>
    <div ref="container" class="editor"></div>
  </div>
</template>
<script setup lang="ts">
import * as monaco from "monaco-editor";
import { onMounted, ref, watch } from "vue";
import { useProject } from "../store/project";
const project = useProject();
const container = ref<HTMLDivElement|null>(null);
let editor: monaco.editor.IStandaloneCodeEditor;
const active = ref("index.html");
const files = ["index.html","main.js","style.css"];
```

```
function open(f:string){ active.value=f; const v = project.getFile(f);
editor?.setValue(v); }
async function save(){ if(!editor) return; await project.saveFile(active.value,
editor.getValue()); }

onMounted(()=>{
  editor = monaco.editor.create(container.value!, { value:
project.getFile(active.value), language: "javascript", theme: "vs-dark",
automaticLayout: true });
});

watch(()=>active.value, (f)=>{
  const lang = f.endsWith(".html")?"html":
f.endsWith(".css")?"css":"javascript";
  monaco.editor.setModelLanguage(editor.getModel()!, lang);
  editor.setValue(project.getFile(f));
});

// live updates from store
watch(()=>project.files.get(active.value), (val)=>{ if(val!=null && editor &&
editor.getValue()!==val) editor.setValue(val); });
</script>
<style scoped>
.wrap{display:flex;flex-direction:column;height:100%}
.tabs{display:flex;gap:6px;background:#0f172a;padding:6px}
.tabs button{background:#111827;color:#e5e7eb;border:none;border-radius:
6px;padding:6px 10px}
.tabs button.active{background:#1f2937}
.editor{flex:1}
.spacer{flex:1}
</style>
```

apps/web/src/components/PreviewPane.vue

```
<template>
  <iframe ref="frame" class="frame"></iframe>
</template>
<script setup lang="ts">
import { onMounted, ref, watch } from "vue";
import { useProject } from "../store/project";
const project = useProject();
const frame = ref<HTMLIFrameElement|null>(null);
function render(){
  if(!project.current || !frame.value) return;
  const html = project.getFile("index.html");
  const css  = `<style>${project.getFile("style.css")}</style>`;
```

```
  const js   = `<script>${project.getFile("main.js")}<\/script>`;
  const doc = `<!doctype html><html><head><meta charset='utf-8'>${css}</
head><body>${html}${js}</body></html>`;
  const blob = new Blob([doc], { type: "text/html" });
  frame.value.src = URL.createObjectURL(blob);
}
onMounted(()=>render());
watch(()=>[...project.files.values()], ()=>render(), { deep:true });
</script>
<style scoped>
.frame{width:100%;height:100%;border:none;background:#fff}
</style>
```

apps/web/src/components/GraphView.vue

```
<template>
  <svg ref="svg" class="graph"></svg>
</template>
<script setup lang="ts">
import { ref, onMounted } from "vue";
const svg = ref<SVGSVGElement|null>(null);
interface Node { id:string; x:number; y:number; vx:number; vy:number }
interface Link { a:string; b:string; label?:string }
const nodes: Node[] = [];
const links: Link[] = [];
function init(){
  // demo small graph; backend can push real graphs via socket "agent:event"
type "graph"
  const ids = ["User","Orchestrator","Planner","Coder","Tester"];
  ids.forEach((id,i)=>nodes.push({ id, x: 200+ i*40, y: 120, vx:0, vy:0 }));
  links.push({ a:"User", b:"Orchestrator", label:"asks" }, { a:"Orchestrator",
b:"Planner" }, { a:"Orchestrator", b:"Coder" }, { a:"Orchestrator",
b:"Tester" });
}
function tick(){
  // simple force
  for(const l of links){
    const A = nodes.find(n=>n.id===l.a)!; const B = nodes.find(n=>n.id===l.b)!;
    const dx=B.x-A.x, dy=B.y-A.y; const d=Math.hypot(dx,dy)||1; const
k=0.02*(d-100);
    const ux=dx/d, uy=dy/d; A.vx+=k*ux; A.vy+=k*uy; B.vx-=k*ux; B.vy-=k*uy;
  }
  for(const n of nodes){ // repulsion
    for(const m of nodes){ if(n===m) continue; const dx=n.x-m.x, dy=n.y-m.y;
const d=Math.hypot(dx,dy)||1; const f=1000/(d*d); n.vx+= (dx/d)*f; n.vy+= (dy/
d)*f; }
```

```
    n.vx*=0.9; n.vy*=0.9; n.x+=n.vx; n.y+=n.vy;
  }
  render(); requestAnimationFrame(tick);
}
function render(){
  const s = svg.value!; s.innerHTML="";
  const g = document.createElementNS("http://www.w3.org/2000/svg","g");
s.appendChild(g);
  for(const l of links){
    const A = nodes.find(n=>n.id===l.a)!; const B = nodes.find(n=>n.id===l.b)!;
    const line = document.createElementNS(s.namespaceURI, "line");
    line.setAttribute("x1", String(A.x)); line.setAttribute("y1", String(A.y));
    line.setAttribute("x2", String(B.x)); line.setAttribute("y2", String(B.y));
    line.setAttribute("stroke", "#94a3b8"); line.setAttribute("stroke-
width","1.5"); g.appendChild(line);
  }
  for(const n of nodes){
    const circle = document.createElementNS(s.namespaceURI, "circle");
circle.setAttribute("cx", String(n.x)); circle.setAttribute("cy", String(n.y));
circle.setAttribute("r","16"); circle.setAttribute("fill","#0ea5e9");
g.appendChild(circle);
    const text = document.createElementNS(s.namespaceURI, "text");
text.setAttribute("x", String(n.x)); text.setAttribute("y", String(n.y+4));
text.setAttribute("fill","#0f172a"); text.setAttribute("text-anchor","middle");
text.setAttribute("font-size","10"); text.textContent=n.id; g.appendChild(text);
  }
}

onMounted(()=>{ init(); tick(); });
</script>
<style scoped>
.graph{width:100%;height:100%;background:#020617}
</style>
```

apps/web/src/components/ForceFieldBg.vue

```
<template>
  <canvas ref="cv" class="bg"></canvas>
</template>
<script setup lang="ts">
import { onMounted, ref } from "vue";
const cv = ref<HTMLCanvasElement|null>(null);
function rnd(n=1){ return Math.random()*n; }
function animate(ctx:CanvasRenderingContext2D,W:number,H:number){
  const pts = Array.from({length:600},
()=>({x:rnd(W),y:rnd(H),vx:rnd(2)-1,vy:rnd(2)-1}));
```

```
  function loop(){
    ctx.fillStyle = "rgba(2,6,23,0.08)"; ctx.fillRect(0,0,W,H);
    ctx.fillStyle = "rgba(64,242,208,0.9)";
    for(const p of pts){ p.x+=p.vx; p.y+=p.vy; if(p.x<0||p.x>W) p.vx*=-1;
if(p.y<0||p.y>H) p.vy*=-1; ctx.fillRect(p.x,p.y,1.5,1.5); }
    requestAnimationFrame(loop);
  }
  loop();
}
onMounted(()=>{ const c=cv.value!, ctx=c.getContext("2d")!; const
W=c.width=window.innerWidth, H=c.height=window.innerHeight;
c.style.width=W+"px"; c.style.height=H+"px"; animate(ctx,W,H); });
</script>
<style scoped>
.bg{position:fixed;inset:0;z-index:-1}
</style>
```

apps/web/src/views/DashboardView.vue

```
<template>
  <ForceFieldBg/>
  <TopBar/>
  <main class="dash">
    <section class="left">
      <h2>Your Projects</h2>
      <button @click="create">+ New Project</button>
      <ul>
        <li v-for="p in project.list" :key="p.id" @click="open(p.id)">{{p.name}}
</li>
      </ul>
    </section>
    <section class="right">
      <h2>Activity</h2>
      <p>Welcome! Select or create a project to begin. Chat with agents and
watch the graph animate as they collaborate.</p>
    </section>
  </main>
</template>
<script setup lang="ts">
import TopBar from "../components/TopBar.vue";
import ForceFieldBg from "../components/ForceFieldBg.vue";
import { useProject } from "../store/project";
import { onMounted } from "vue";
import { useRouter } from "vue-router";
const project = useProject();
const router = useRouter();
```

```
async function create(){ const p = await project.create("Untitled");
router.push(`/project/${p.id}`); }
async function open(id:string){ router.push(`/project/${id}`); }
onMounted(()=>project.loadList());
</script>
<style scoped>
.dash{display:grid;grid-template-columns:320px 1fr;gap:16px;padding:
16px;color:#e2e8f0}
.left,.right{background:#0f172a;border-radius:12px;padding:16px}
li{list-style:none;padding:8px;border-radius:8px;background:#111827;margin:6px
0;cursor:pointer}
li:hover{background:#1f2937}
button{background:#06b6d4;border:none;border-radius:8px;padding:8px
12px;color:#0f172a;font-weight:600;margin-bottom:12px}
</style>
```

apps/web/src/views/ProjectView.vue

```
<template>
  <TopBar/>
  <div class="layout">
    <div class="col code">
      <CodeEditors/>
    </div>
    <div class="col preview">
      <PreviewPane/>
    </div>
    <div class="col chat">
      <ChatPanel/>
      <div class="goal">
        <input v-model="goal" placeholder="High-level goal for orchestrator…"/>
        <button @click="run">Orchestrate</button>
      </div>
      <div class="graph"><GraphView/></div>
    </div>
  </div>
</template>
<script setup lang="ts">
import TopBar from "../components/TopBar.vue";
import CodeEditors from "../components/CodeEditors.vue";
import PreviewPane from "../components/PreviewPane.vue";
import ChatPanel from "../components/ChatPanel.vue";
import GraphView from "../components/GraphView.vue";
import { useRoute } from "vue-router";
import { useProject } from "../store/project";
import { onMounted, ref } from "vue";
```

```
const route = useRoute();
const project = useProject();
const goal = ref("");
async function run(){ await project.orchestrate(goal.value); }
onMounted(()=>{ project.open(String(route.params.id)); });
</script>
<style scoped>
.layout{display:grid;grid-template-columns:1.2fr 1fr 0.9fr;grid-template-rows:
1fr;gap:8px;height:calc(100vh - 56px);background:#020617}
.col{display:flex;background:#0b1120;border-radius:12px;overflow:hidden}
.code{grid-column:1}
.preview{grid-column:2}
.chat{grid-column:3;flex-direction:column}
.goal{display:flex;gap:8px;padding:8px;background:#0f172a}
.goal input{flex:1;background:#111827;color:#e5e7eb;border:none;border-radius:
8px;padding:8px}
.goal button{background:#06b6d4;border:none;border-radius:8px;padding:8px
12px;color:#0f172a;font-weight:600}
.graph{flex:1}
</style>
```

apps/web/src/views/LoginView.vue

```
<template>
  <div class="login">
    <h1>Login</h1>
    <button @click="auth.login">Continue with SSO</button>
  </div>
</template>
<script setup lang="ts">
import { useAuth } from "../store/auth";
const auth = useAuth();
</script>
<style scoped>
.login{height:100vh;display:grid;place-
items:center;background:#020617;color:#e2e8f0}
button{background:#06b6d4;border:none;border-radius:8px;padding:12px
18px;color:#0f172a;font-weight:600}
</style>
```

Scripts

`scripts/setup.sh`

```
#!/usr/bin/env bash
set -euo pipefail

# Ubuntu 24 / Fedora basics
if [ -f /etc/debian_version ]; then
  sudo apt-get update -y
  sudo apt-get install -y curl git build-essential ripgrep
elif [ -f /etc/fedora-release ]; then
  sudo dnf install -y curl git @development-tools ripgrep
fi

# Node & pnpm
if ! command -v node >/dev/null; then
  curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
  sudo apt-get install -y nodejs || true
fi
npm i -g pnpm@9

# Install workspace deps
pnpm i

echo "✓ Setup complete. Configure .env and run: pnpm dev"
```

`scripts/seed-demo.sh`

```
#!/usr/bin/env bash
set -e
DATA=./data/projects/demo
mkdir -p "$DATA"
cat > "$DATA/project.json" <<JSON
{ "id": "demo", "ownerId": "demo-user", "name": "Electrodynamic Sketch",
"createdAt": "$(date -Iseconds)", "updatedAt": "$(date -Iseconds)" }
JSON
cat > "$DATA/index.html" <<'HTML'
<div id="app"></div>
HTML
cat > "$DATA/style.css" <<'CSS'
html,body,#app{height:100%;margin:0;background:#000;color:#40f2d0;font-
family:system-ui}
canvas{position:fixed;inset:0}
CSS
```

```
cat > "$DATA/main.js" <<'JS'
const c=document.createElement('canvas');document.body.appendChild(c);const
x=c.getContext('2d');function R(){c.width=innerWidth;c.height=innerHeight}
addEventListener('resize',R);R();const pts=Array.from({length:888},
()=>({x:Math.random()*c.width,y:Math.random()*c.height,vx:Math.random()-0.5,vy:Math.random()-0.5}
step()
{x.fillStyle='rgba(0,0,0,0.08)';x.fillRect(0,0,c.width,c.height);x.fillStyle='#40f2d0';for(const
p of pts){p.x+=p.vx;p.y+=p.vy;if(p.x<0||p.x>c.width)p.vx*=-1;if(p.y<0||
p.y>c.height)p.vy*=-1;x.fillRect(p.x,p.y,1.2,1.2)}requestAnimationFrame(step)}
step();
JS
echo "✓ Demo project seeded at $DATA"
```

Notes & Next Steps

- **Codex CLI**: To enable local Codex CLI execution, install `@openai/codex` inside the server container or host, and wire `codexCli.ts` into orchestrator when `USE_CODEX_CLI=1` is set. The CLI can apply diffs and run tests autonomously.
- **Security**: Tighten sandboxing, rate limits, and diff application (parse unified diffs and patch files precisely). Add git commits per AI change.
- **Persistence**: Swap file-based storage to a DB (Postgres + Prisma) if you need multi-tenant scale and audit trails.
- **SSO Providers**: Replace the placeholder OIDC config with your provider (Auth0, Okta, Keycloak). Ensure HTTPS in production and set correct callback URLs.
- **Visuals**: Bring in more of your uploaded sketches (color palettes, PRNG seeds, audio modules) in `ForceFieldBg.vue` or a dedicated “Studio” view.
- **Testing**: Add end-to-end tests (Playwright) and unit tests for orchestrator flows.

Enjoy building — and watching agents collaborate ✨