package.json

```json
{
  "name": "sdl-alpha",
  "private": true,
  "version": "1.0.0",
  "type": "module",
  "engines": { "node": ">=18.17.0" },
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "start": "node server.js",
    "postinstall": "vite build"
  },
  "dependencies": {
    "express": "^4.19.2",
    "compression": "^1.7.4",
    "react": "^18.3.1",
    "react-dom": "^18.3.1"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.3.1",
    "vite": "^5.4.0"
  }
}
```

vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  base: '/',
  plugins: [react()],
  build: {
    outDir: 'dist',
    sourcemap: true
  },
  server: {
    port: 5173,
    open: false
  }
})
```

server.js

```
import express from 'express'
import compression from 'compression'
import path from 'path'
import { fileURLToPath } from 'url'
import fs from 'fs'

const __filename = fileURLToPath(import.meta.url)
const __dirname = path.dirname(__filename)

const app = express()
const port = process.env.PORT || 3000

app.use(compression())
app.use(express.json())

// Health check
app.get(['/','/health'], (_req, res) => {
  res.json({ ok: true, app: 'sdl-alpha', version: '1.0.0' })
})

// Serve static assets from dist
const distPath = path.join(__dirname, 'dist')
app.use(express.static(distPath, { index: false }))

// SPA fallback: send index.html for all non-file routes
app.get('*', (req, res) => {
  const indexFile = path.join(distPath, 'index.html')
  if (fs.existsSync(indexFile)) {
    res.sendFile(indexFile)
  } else {
    res
      .status(503)
      .send('<h1>Build not ready</h1><p>dist/index.html not found. Did postinstall run?</p>')
  }
})

app.listen(port, () => {
  console.log(`SDL alpha server listening on port ${port}`)
})
```

index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>SDL — #alpha</title>
    <meta name="description" content="SDL alpha frontend" />
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

.gitignore

```
# Dependencies
node_modules/

# Build output
dist/

# Logs
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*

# OS
.DS_Store
```

README.md

~~~markdown
# SDL — #alpha Frontend

This is a production-ready Vite + React app configured for Railway.
It includes:
- Vite build pipeline
- Express server for production (serves `dist/`)
- SPA fallback routing
- `/health` endpoint
- ErrorBoundary to avoid blank screens

## Run locally
```bash
npm install
npm run dev
# open http://localhost:5173
```

## Build + run production server locally
```bash
npm run build
npm start
# open http://localhost:3000
```

## Deploy on Railway
1. Connect this repo to your Railway **frontend service**.
2. Ensure Start Command is `npm start` (Railway runs `npm install` automatically).
3. The `postinstall` script builds the app during deploy.
4. Hit the service URL → `/health` should return `{ ok: true }`.
~~~

src/main.jsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import ErrorBoundary from './shared/ErrorBoundary.jsx'

const rootEl = document.getElementById('root')
if (!rootEl) {
  const msg = 'Root element #root not found — cannot mount React.'
  console.error(msg)
  document.body.innerHTML = '<pre style="color:red">' + msg + '</pre>'
} else {
  ReactDOM.createRoot(rootEl).render(
    <React.StrictMode>
      <ErrorBoundary>
        <App />
      </ErrorBoundary>
    </React.StrictMode>
  )
}
```

src/shared/ErrorBoundary.jsx

```
import React from 'react'

export default class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props)
    this.state = { hasError: false, error: null }
  }
  static getDerivedStateFromError(error) {
    return { hasError: true, error }
  }
  componentDidCatch(error, errorInfo) {
    console.error('App crashed:', error, errorInfo)
  }
  render() {
    if (this.state.hasError) {
      return (
        <div style={{ fontFamily: 'system-ui, Arial, sans-serif', padding: 24 }}>
          <h1 style={{ color: '#b00020' }}>Something went wrong.</h1>
          <pre style={{ background: '#f6f6f6', padding: 12, borderRadius: 8, overflowX: 'auto' }}>
{String(this.state.error)}
          </pre>
        </div>
      )
    }
    return this.props.children
  }
}
```

src/App.jsx

```
import React, { useEffect, useState } from 'react'

export default function App() {
  const [health, setHealth] = useState(null)
  const [err, setErr] = useState(null)

  useEffect(() => {
    const controller = new AbortController()
    const url = '/health'
    fetch(url, { signal: controller.signal })
      .then(r => r.ok ? r.json() : Promise.reject(new Error('Health fetch failed: ' + r.status)))
      .then(setHealth)
      .catch(e => setErr(String(e)))
    return () => controller.abort()
  }, [])

  return (
    <div style={{ fontFamily: 'Inter, system-ui, Arial, sans-serif', padding: 24 }}>
      <h1 style={{ margin: 0 }}>SDL — #alpha</h1>
      <p style={{ marginTop: 8 }}>Vite + React is wired up and compiling JSX.</p>

      <section style={{ marginTop: 20 }}>
        <h2 style={{ marginBottom: 8, fontSize: 16 }}>Health</h2>
        {err && <div style={{ color: '#b00020' }}>Error: {err}</div>}
        <pre style={{ background: '#f6f6f6', padding: 12, borderRadius: 8, overflowX: 'auto' }}>
{JSON.stringify(health, null, 2)}
        </pre>
      </section>
    </div>
  )
}
```