Dashboard & Pipeline Enhancement Specification

Chosen Tech Stack

- **Frontend**: Next.js (React) + Tailwind CSS + TypeScript
- **Real-Time**: Socket.IO via a Node.js gateway
- **Backend**: Next.js API routes + Node.js server using `ioredis` to subscribe/publish Redis
- **Storage**: Redis (Pub/Sub & TimeSeries) and optional Postgres for long-term archive

Directory Scaffold

```
/agents              # existing AI agents
/dashboard           # new dashboard project
  /components        # React UI components (LiveFeed, IdeaCard, Controls)
  /pages             # Next.js pages
    index.tsx        # Main dashboard view
    api/             # Next.js API routes
      submitIdea.ts  # handles manual submissions
      submitFeedback.ts # collects pilot results
  /lib               # shared utilities (redis client, socket setup)
  /styles            # Tailwind CSS config and globals
  socketServer.js    # Socket.IO server gateway
  tsconfig.json      # TypeScript config
  tailwind.config.js # Tailwind CSS config
  package.json       # project deps and scripts
```

Immediate Next Steps (Copy-Paste)

1. **Scaffold Dashboard Project**

```
cd ~/Desktop/NovaOS
npx create-next-app@latest dashboard --typescript --eslint
cd dashboard
npm install tailwindcss postcss autoprefixer socket.io-client ioredis
socket.io
npx tailwindcss init -p
```

2. **Configure Tailwind**
3. In `tailwind.config.js`, set content paths:

```
module.exports = {
  content: ['./pages/**/*.{ts,tsx}', './components/**/*.{ts,tsx}'],
```

```
  theme: { extend: {} },
  plugins: [],
}
```

4. Add Tailwind directives to `styles/globals.css`:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

5. **Build Socket Gateway** (`socketServer.js` in `/dashboard`)

```
const { createServer } = require('http');
const { Server } = require('socket.io');
const Redis = require('ioredis');

const httpServer = createServer();
const io = new Server(httpServer, { cors: { origin: '*' } });
const sub = new Redis(process.env.REDIS_URL);

sub.subscribe('novaos:queue', () => {});
sub.on('message', (_, message) => {
  io.emit('message', JSON.parse(message));
});

httpServer.listen(4001, () => console.log('Socket server on :4001'));
```

6. **Implement LiveFeed Component**
7. In `/dashboard/components/LiveFeed.tsx`, connect to Socket.IO:

```
import { useEffect, useState } from 'react';
import io from 'socket.io-client';
const socket = io('http://localhost:4001');

export default function LiveFeed() {
  const [items, setItems] = useState<any[]>([]);
  useEffect(() => {
    socket.on('message', (msg) => setItems((prev) => [msg, ...prev]));
    return () => { socket.off('message'); };
  }, []);
  return (
    <div className="space-y-2 overflow-auto h-96">
      {items.map((m, i) => (
        <div key={i} className="p-2 bg-gray-100 rounded">{m.type}:
{JSON.stringify(m)}</div>
```

```
      ))}
    </div>
  );
}
```

8. **Add Manual Submission API**
9. Create `/dashboard/pages/api/submitIdea.ts`:

```
import type { NextApiRequest, NextApiResponse } from 'next';
import Redis from 'ioredis';
const r = new Redis(process.env.REDIS_URL);

export default async function handler(req: NextApiRequest, res:
NextApiResponse) {
  const { text, category } = req.body;
  await r.rpush('novaos:queue', JSON.stringify({ type: 'pulse', source:
'manual', text, category }));
  res.status(200).json({ status: 'ok' });
}
```

10. **Wire Up UI and Test**
11. In `pages/index.tsx`, import `LiveFeed` and add a simple form to POST to `/api/submitIdea`.
12. Run both servers:

```
# Terminal 1
cd dashboard
npm run dev

# Terminal 2
cd dashboard
node socketServer.js
```

Follow these steps to see live pulses, manually inject ideas, and have an operational dashboard within minutes—no deep code expertise required.