Node Coder

Tech Stack Server

- **TypeScript**
- **Node.js & Express**: Server framework for handling requests and responses.
- **Socket.io**: For real-time socket communication.
- **PostgreSQL** (with Prisma ORM): Database for storing and managing data.
- **Threads.js**: Library for managing worker threads.
- **SharedArrayBuffer**: For handling shared variables.

Tech Stack Client

- **TypeScript**
- **React** (bundler: Vite): For building the user interface.
- **Socket.io**: For real-time socket communication.
- **React Router**: For client-side routing.
- **Tailwind CSS**: For styling.

Flow Process

1. **User Login**
2. **Input Details**: Personnel, Product, Batch Number, and Estimated Quantity.
3. **Initialize**: Spawn worker threads and open connections (Printer, PLC, Barcode).
4. **Dashboard**: Open the print dashboard page.
5. **Start Printing**: Click the start print button to begin printing.

Queues

- **print_queue**: Queue for unique codes from the DB that haven't been sent to the printer.
- **printed_queue**: Queue for unique codes that have already been sent to the printer.
- **DB_update_queue**: Queue for unique codes that have been printed and need to be updated in the database.

Barcode Worker Thread

For every interval of 1 second, the worker will:

1. Check if the connection is established.
2. Send commands to the barcode.
3. Update the barcode count.

PLC Process

For every interval of 1 second, the worker will:

1. Check if the connection is established.
2. Send commands to check if there is any errors
3. Update status based on response

DB Worker Thread

For every interval of 1 second, the worker will:

1. Check if `DB_update_queue` (already printed) is not empty, update unique codes in the queue.
2. Check if there is space available in `print_queue`, get unique codes (use update) and push to `print_queue`.

Printer Worker Thread

For every interval of 1 second, the worker will:

1. Check the connection.
2. Send a command to check status.

   If a printer status is received, the worker will:

   - Check for every status change or error.
   - Show info to the UI (client) if there is an error or warning.

   If a mailing status is received, the worker will:

   - Check for already printed unique codes and move them from `printed_queue` to `DB_update_queue`.
   - Check if there is space in the printer buffer, send a command to print the unique code from `print_queue`, and move the unique code to `printed_queue`.

Problems

- Multiple printer integration.
- Development tools: Printer simulator, PLC simulator, Barcode Scanner simulator.

Timeline

Phase 1: Planning & Setup (1 day)

- Define project requirements.
- Set up the development environment.
- Create initial project structure.

Phase 2: Backend Development (1 days)

- Implement user authentication.
- Set up PostgreSQL database with Prisma ORM.
- Develop API endpoints using Node.js & Express.
- Develop realtime socket communication using Socket.io
- Implement worker thread management with Threads.js.

Phase 2.5: Testing & Debugging (1 days)

- Write unit tests for backend.
- Test API endpoints
- Debug and fix any issues.

Phase 3: Frontend Development (2 days)

- Set up React project with Vite.
- Implement user login and input forms.
- Develop the print dashboard page.
- Integrate Socket.io for real-time updates.

Phase 3.5: Testing & Debugging (2 days)

- Write unit tests for frontend.
- Test Login , Batch Creation and Print Dashboard Feature
- Test real-time communication between Client and Server using API & Websocket.
- Debug and fix any issues.

Phase 4: Worker Threads Integration (2 days)

- Develop Barcode Worker Thread.
- Develop DB Worker Thread.
- Develop Printer Worker Thread.
- Implement SharedArrayBuffer for shared variables.

Phase 4.5: Testing & Debugging (1 days)

- Write unit tests for thread integration.
- Test worker threads and real-time communication.
- Debug and fix any issues.

Phase 5: Final Testing & Deployment (1 days)

- Perform end-to-end testing.
- Optimize performance and fix any final bugs.
- Deploy the application to the production environment.