**PAYTRAX FUNCTIONAL DESCRIPTION**

It is a **standalone, browser-based payroll management system** designed for small businesses. It operates entirely on the client-side, meaning it doesn't require a server or internet connection after the initial page load. All data is stored locally in the user's browser, making it a private and self-contained tool.

The application's architecture is modern and modular, promoting a clear separation of concerns which will make future debugging and feature additions much more manageable.

**Core Architecture and Concepts**

The application is built on a few key principles:

* **Single Source of Truth:** The entire state of the application (all settings, employee data, pay periods, etc.) is held within a single JavaScript object called appData (defined in js/state.js). Every calculation and UI display reads from this object, and every user action that changes data modifies this object.
* **Modular Design:** The JavaScript code is logically split into modules, each with a distinct responsibility:
  + main.js: The central controller that initializes the application and orchestrates events, connecting user actions (like clicks and input changes) to the appropriate logic and UI functions.
  + state.js: Manages the appData object and handles loading from and saving to the browser's storage.
  + logic.js: The "brains" of the application. It contains all the business logic for payroll calculations, generating pay periods, creating report data, and managing employee/transaction data. It does not directly interact with the user interface.
  + ui.js: Manages all DOM manipulation. It's responsible for taking data from the appData state object and rendering it on the screen, populating forms, updating tables, and controlling which tab is visible.
  + db.js: A dedicated module for handling data persistence using **IndexedDB**, a robust browser database. It provides a clean API for saving and loading the appData object.
  + data-io.js & migration.js: These handle data portability (importing/exporting JSON backups) and ensuring that older backup files can be updated to the latest data structure, which is a crucial feature for long-term usability.
  + utils.js: A collection of helper functions (like date formatting) used across the application.
* **Event-Driven Calculation:** The application feels dynamic because calculations are triggered by user input events. When a user changes the hours in a field, an event listener in main.js calls functions in logic.js to recalculate pay, which in turn updates the appData object. Then, functions in ui.js are called to refresh the dashboard with the new data.
* **Robust Data Persistence:** The application intelligently saves data. It prioritizes **IndexedDB** for storage, which is ideal for structured data. It also includes a graceful fallback to localStorage if IndexedDB isn't available, and it can migrate data from localStorage to IndexedDB, ensuring data isn't lost. All data is saved automatically after any change.

**Functional Breakdown by Tab**

The application's functionality is organized into intuitive tabs:

**Dashboard**

This is the primary workspace for running payroll.

1. The user selects an **Employee** and a **Pay Period** from dropdowns.
2. The application loads any existing hours for that period.
3. The user enters or modifies **Regular, Overtime, PTO, and Holiday hours**.
4. As soon as the hours are changed, the calculatePay function is triggered:
   * It calculates gross pay based on the employee's specific pay rates and multipliers.
   * It computes all relevant taxes (Federal, State, Local, FICA, Medicare, SUTA, FUTA). A notable feature here is the **"running remainder"** logic, which tracks fractions of a cent across pay periods to ensure tax amounts are rounded accurately over time.
   * It calculates the net pay and updates the employee's PTO balance.
   * It automatically creates a corresponding debit transaction in the banking module for the total payroll cost (gross pay + employer taxes).
5. The **Pay Period Details** section updates instantly to show the new Gross Pay, Net Pay, and total cost.
6. The **Bank Funds Required** widgets provide a forward-looking projection of cash needed for upcoming payrolls based on historical averages.

**Settings**

This is the configuration hub for the entire application.

* **Company Settings:** Basic company information, tax year, and crucially, the **Pay Frequency** and the **start date of the first pay period**. These two inputs are used by generateBasePayPeriods to create the entire payroll calendar for the year.
* **Tax Settings:** Global tax rates (like Social Security and Medicare) and the payment frequencies for various taxes, which are used by the Tax Deposit report.
* **Employee Management:** The user can add new employees or select an existing one to edit. The form captures personal details, pay rates, tax withholding percentages, and PTO information. This is also where employees can be deleted.
* **Data Management:** Users can **Export** the entire application state into a single JSON file as a backup and **Import** data from such a file, which will completely overwrite the existing data.

**Pay Periods**

This tab provides a read-only, detailed table view of all pay periods for the currently selected employee. It's a quick way to review the payroll history, showing hours, gross pay, net pay, and a breakdown of all taxes for every period in the year.

**Pay Stub**

After calculating a period on the dashboard, the user can click "Generate Pay Stub". This populates a professional-looking, printable pay stub with all relevant information:

* Company and employee details.
* A breakdown of current and Year-to-Date (YTD) earnings.
* A breakdown of current and YTD taxes.
* A summary of the employee's PTO usage and remaining balance for the period.

**Reports**

This is a powerful compliance section that aggregates payroll data into several key financial reports.

* **Tax Deposit Report:** Calculates the total tax liability (e.g., for Form 941 payments) for a given period (weekly, monthly, quarterly) based on the frequencies set in Settings.
* **Annual W-2 Data:** Generates a summary for each employee that directly corresponds to the boxes on an IRS Form W-2.
* **IRS Form 941 Data (Quarterly):** Aggregates all data for a specific quarter to help the user fill out their quarterly federal tax return.
* **IRS Form 940 Data (Annual):** Aggregates data for the year to help fill out the annual federal unemployment (FUTA) tax return.
* **Custom Reports:** Allows the user to generate detailed reports on employee wages or employer expenses for any given date range.

**Banking**

This module functions as a simple bank register.

* It displays a running balance.
* Users can manually add debit (withdrawal) and credit (deposit) transactions.
* As mentioned, payroll runs from the dashboard automatically add debit transactions, tying payroll expenses directly to the bank ledger.
* A thoughtful user experience feature is the **Insufficient Funds Alert**, a modal window that warns the user if a payroll run causes the bank balance to drop below zero.