Shift Relay

Backend Integration Plan

Google Sheets Integration for Dynamic Employee & Shift Management

Document Version 1.0 | February 2026

Prepared for: Shift Relay Operations

1. Overview

This document outlines how to connect the Shift Relay web app to Google Sheets so the Managing Director (MD) can manage all employee shifts, logins, and schedules from a single spreadsheet - no coding required.

When the MD updates the Google Sheet, the changes are automatically picked up by the backend server within 5 minutes and reflected in the live web application across all browsers and devices.

Architecture

The system has three main components:

```
GOOGLE SHEET         NODE.JS SERVER        REACT FRONTEND
  (MD updates)   --->  (server.cjs)    --->  (Web App)
                       Port 3001              Port 5173

  * MD edits           * Reads sheet          * Displays live
    shifts & users       every 5 min            relay track
                       * Handles login        * Shows shift
                       * Tracks state           progress
```

How It Works

- MD opens the Google Sheet and edits employee data or shift timings
- Backend server reads the sheet every 5 minutes via the Google Sheets API
- Employee list, shifts, and credentials are cached in server memory
- Frontend polls the backend every second for real-time state
- Employees log in with credentials from the sheet
- Live tracking (running, paused, waiting) works across all browsers

2. Google Sheet Structure

The MD maintains one Google Sheet with the following tabs. This is the only thing the MD needs to manage.

Tab 1: Employees

Contains all employee information and login credentials.

Name	Password	Department	Role	Status
SUHAIL	suhail123	Operations	employee	active
AZEEZ	azeez123	Operations	employee	active
IQBAL	iqbal123	Operations	employee	active
FARHAN	farhan123	Security	employee	active
ADMIN	admin123	Management	master	active

Tab 2: Schedule

Contains shift timings and relay order for each employee.

Name	Start	End	Days	From	Until	Order
SUHAIL	9:00 AM	6:00 PM	Mon-Fri	2026-02-01		1
AZEEZ	6:00 PM	2:00 AM	Mon-Fri	2026-02-01		2
IQBAL	2:00 AM	9:00 AM	Mon-Fri	2026-02-01		3
FARHAN	10:00 AM	7:00 PM	Mon-Fri	2026-02-15		4

Tab 3: Settings (Optional)

Global configuration for the application.

Key	Value	Description
company_name	Shift Relay Co.	Shown in app header
sync_interval	5	Sync frequency (minutes)
relay_mode	sequential	sequential or parallel

3. What the MD Does

The MD only needs to edit the Google Sheet. No coding, no server restarts, no app updates.

Adding a New Employee

Step 1: Add a row to the "Employees" tab with their name, password, department, role, and status.

Step 2: Add their shift to the "Schedule" tab with start time, end time, working days, and relay order.

Step 3: Wait 5 minutes. The employee can now log in and appears on the relay track.

Changing Shift Timings

Simply edit the "Shift Start" and "Shift End" columns in the Schedule tab. The app updates within 5 minutes.

Removing an Employee

Change the "Status" column to "inactive" in the Employees tab. The employee will no longer appear or be able to log in.

Force Sync

The Master Dashboard has a "Sync Now" button that forces an immediate re-read of the Google Sheet.

4. Backend API Endpoints

The Node.js server (port 3001) exposes these REST API endpoints:

Method	Endpoint	Description
GET	/api/state	Real-time state (logged in, paused)
GET	/api/employees	All active employees with shifts
GET	/api/schedule	Today's relay order & timings
POST	/api/login	Validate credentials & register
POST	/api/logout	Unregister & save pause state
POST	/api/sync	Force re-sync from Google Sheet

Example: GET /api/employees

```
{
  "employees": [
    { "name": "SUHAIL", "dept": "Operations",
      "shift": "9:00 AM - 6:00 PM", "order": 1 },
    { "name": "AZEEZ", "dept": "Operations",
      "shift": "6:00 PM - 2:00 AM", "order": 2 },
    { "name": "IQBAL", "dept": "Operations",
      "shift": "2:00 AM - 9:00 AM", "order": 3 }
  ],
  "lastSyncedAt": "2026-02-13T18:45:00Z"
}
```

Example: GET /api/schedule

```
{
  "relay": [
    { "name": "SUHAIL", "start": "09:00",
      "end": "18:00", "order": 1 },
    { "name": "AZEEZ", "start": "18:00",
      "end": "02:00", "order": 2 },
    { "name": "IQBAL", "start": "02:00",
      "end": "09:00", "order": 3 }
  ],
  "date": "2026-02-13",
  "totalEmployees": 3
}
```

5. Setup Steps

Step 1: Google Cloud Project

- Go to console.cloud.google.com
- Create a new project: "Shift Relay"
- Enable the Google Sheets API
- Create a Service Account (IAM & Admin > Service Accounts)
- Download the JSON key file as credentials.json
- Copy the service account email

Step 2: Google Sheet Setup

- Create a new Google Sheet with tabs: Employees, Schedule, Settings
- Fill in column headers as described in Section 2
- Share the sheet with the service account email (Viewer access)
- Copy the Sheet ID from the URL

Step 3: Server Configuration

Create a .env file in the project root:

```
GOOGLE_SHEET_ID=your_sheet_id_here
GOOGLE_CREDENTIALS_PATH=./credentials.json
SYNC_INTERVAL_MINUTES=5
PORT=3001
```

Step 4: Install Dependencies

```
npm install googleapis dotenv
```

Step 5: Deploy

- Update server.cjs with Google Sheets integration
- Update frontend to fetch data dynamically
- Test with real sheet data

6. Security Notes

Never commit credentials.json to git. Add it to .gitignore.

- Passwords in sheets: Fine for internal tools. For production, hash passwords.
- Service account: Read-only access to the sheet. The app only reads, never writes.
- HTTPS: Use HTTPS for all API calls when deploying to production.
- Authentication: Login validation moves to the server (not client-side).

7. Implementation Timeline

Phase	Task	Estimate
1	Google Cloud + Sheet setup	30 min
2	Backend: Sheets integration	2-3 hours
3	Backend: Dynamic login & APIs	1-2 hours
4	Frontend: Dynamic data fetching	2-3 hours
5	Frontend: Dynamic track (N users)	1-2 hours
6	Testing & polish	1-2 hours
	TOTAL	8-12 hours

Questions?

If you have any questions about this plan or need changes to the approach, please discuss before implementation begins.