CuraConnect: System Documentation & Implementation Guide

Version 1.0 | October 18, 2025

1. Project Overview

CuraConnect is a zero-cost, modern, and serverless patient registration and queuing system designed for clinics. It streamlines the patient intake process, reduces administrative workload, and improves the patient experience without any recurring software or hardware costs.

The system allows patients to check in by simply scanning a QR code with their smartphone. Their details are then automatically organized into a live, digital queue that the clinic staff can monitor in real-time.

Core Technologies:

- **Frontend (UI):** GitHub Pages (hosting a static HTML, CSS, & JavaScript file)
- **Backend (API):** Google Apps Script
- **Database:** Google Sheets

---

2. System Architecture & How It Works

The system is built on a **decoupled architecture**, which means the user interface (frontend) is separate from the data processing logic (backend). This makes the application more stable, secure, and much easier to update.

A. The Frontend (The Patient's View)

- **What it is:** A single index.html file hosted for free on GitHub Pages.
- **How it works:** When a patient scans the QR code, it opens this webpage. The page contains the registration form and the JavaScript logic required to capture the patient's data and send it to the backend. It also handles the language switching (English/Marathi).

B. The Backend (The "Engine")

- **What it is:** A Google Apps Script that acts as a secure, private API.
- **How it works:** The script's only job is to listen for incoming data from the frontend. When it receives a submission, it performs all the critical logic:
    1. Validates the phone number to ensure it is 10 digits.
    2. Searches the database to see if the patient is new or returning.
    3. Calculates the correct token number for the day.
    4. Saves the complete, verified record into the Google Sheet.
    5. Sends a success message back to the patient's phone.

C. The Database (The Staff's View)

- **What it is:** A Google Sheet in your Google Drive.
- **How it works:** This spreadsheet is the single source of truth. It acts as a simple, powerful database where all patient records are stored. For the clinic staff, it serves as a live, real-time dashboard of the patient queue.

---

3. Implementation Guide: Step-by-Step Setup

Follow these steps to set up the entire system from scratch.

Phase 1: Backend Setup (Google)

1. **Create the Google Sheet:**
    - Create a new Google Sheet.
    - Rename the sheet tab at the bottom to be **exactly Patient Data**.
    - In the first row, set up these exact headers in this order:

Timestamp, PatientID, Name, Phone, Age, TokenNumber, Date

2. **Create the Apps Script:**
    - In your sheet, go to **Extensions -> Apps Script**.
    - Delete any existing code in the Code.gs file and paste your final, robust backend code.
3. **Deploy the Script:**
    - Click **Deploy -> New deployment**.
    - Set **"Execute as"** to **Me (your email)**.
    - Set **"Who has access"** to **Anyone**.
    - Click **Deploy**.
    - Authorize the permissions.
    - **Copy the final "Web app" URL that ends in /exec**. This is your permanent API endpoint.

Phase 2: Frontend Setup (GitHub)

1. **Create GitHub Repository:**
    - Go to GitHub and create a new **public** repository (e.g., CuraConnect-UI).
2. **Create index.html file:**
    - On your computer, create an index.html file. Paste your final, robust frontend code into it.
3. **Link Frontend to Backend:**
    - In your index.html file, find the line const SCRIPT_URL = "...".

    - Paste the /exec URL you copied from the Apps Script deployment.
4. **Upload to GitHub:**
    - Upload this completed index.html file to your new GitHub repository.
5. **Enable GitHub Pages:**
    - In the repository settings, go to the **Pages** tab.
    - Select the main branch as the source and click **Save**.
    - Wait a minute for the site to go live and copy the final GitHub Pages URL.

Phase 3: Finalization

1. **Generate QR Code:** Use a free online QR code generator to create a code from your final **GitHub Pages URL**.
2. **Print & Display:** Print this QR code and place it at the clinic's reception.

---

4. Precautions & Best Practices

- **Security:** The Google Sheet contains sensitive patient data. **Never** make the sheet itself public. Only share it directly with authorized clinic staff by inviting them via their email address.
- **URL Management:** Your Apps Script /exec URL is like a secret key. While it's used publicly in the HTML, avoid posting it unnecessarily. The GitHub Pages URL is the only one you need to share.
- **Data Backup:** Once a month, it is good practice to make a copy of your Google Sheet (File -> Make a copy) as a simple backup.
- **Timezone:** Ensure your Google Sheet's timezone (File -> Settings) is set to the same timezone as your script (Asia/Kolkata) to prevent date-related bugs.

---

5. Troubleshooting Common & Critical Issues

This section documents the issues we solved together. If the app breaks, the cause is almost certainly listed here.

Issue	Symptoms	Root Cause	Solution
1. "An error occurred"	The error alert appears on the screen. The browser console shows a 500 Internal Server Error.	The backend Apps Script crashed while running.	**Check the Executions Log in Apps Script.** The most common cause is the sheet tab is not named Patient Data or the column headers are in the wrong order.

Issue	**Symptoms**	**Root Cause**	**Solution**
2. "Failed to fetch"	The error alert appears. The browser console shows a TypeError: Failed to fetch or a CORS policy error.	A communication breakdown. The frontend can't talk to the backend.	This is a deployment issue. **Perform a "Triple Check Redeployment"**: 1. Copy the /exec URL. 2. Paste it into your index.html file. 3. Push to GitHub. 4. Go back to Apps Script and deploy a **"New version"**.
3. New Patient ID Every Time	A returning patient submits the form but gets a new Patient ID instead of their old one.	The script can't match the patient. This is caused by a "Number vs. Text" mismatch with the phone number.	Use the final, robust getPatientId function that converts both the form's phone number and the sheet's phone number to **strings** before comparing them.
4. Token Number is Always "1"	Every patient for the day gets the token number 1.	The script can't match the dates. This is caused by a date format mismatch between the sheet and the script.	Use the final, robust getNextToken function that uses Utilities.formatDate to force both dates into the exact same format (dd/MM/yyyy) before comparing.