# Technical Design Document

## 1. Introduction

This Technical Design Document describes the architecture, functionality, and design decisions for the 'CPOF Member Maintenance' Google Apps Script Web Application. The application is designed to manage member data for the Correctional Peace Officers Foundation (CPOF) by integrating a Google Sheets backend with a web-based frontend for searching, viewing, editing, and verifying member information.

## 2. System Overview

The system consists of a Google Apps Script backend connected to a Google Sheet containing member data, and an HTML/CSS/JavaScript frontend rendered as a web app. Bootstrap is used for responsive styling. The application supports dynamic searching, inline filtering, and modal-based editing of member details.

## 3. Architecture

The application follows a client-server architecture where the client (HTML/JavaScript) interacts with the server (Google Apps Script) using the google.script.run API. The backend retrieves, filters, and updates data in the Google Sheet, while the frontend manages user interactions and renders data.

## 4. Frontend Components

### 4.1 HTML Structure

The HTML page contains:  
- A title bar with the CPOF logo on the left, the application title in the center, and a QR code on the right.  
- Filter inputs arranged in two rows for search criteria such as First Name, Last Name, Last 4 SSN, Street Address, City, State, Zip Code, and Facility.  
- A search button and a status message label.  
- A responsive Bootstrap table for displaying member data.  
- A hidden modal form for editing member information.

### 4.2 CSS Styling

Custom CSS styles define spacing, table cell width limits, and modal appearance. Notable styling includes:  
- Limiting table cell width with ellipsis overflow.  
- Modal blur effect on background content.  
- Special formatting for certain labels (bold and larger font).

### 4.3 JavaScript Functionality

JavaScript functions manage:  
- Fetching headers and data from the backend.  
- Rendering data into the table with clickable links on the first column to open the modal.  
- Dynamically enabling/disabling the Search button based on filter input length.  
- Displaying filtered or empty data sets.  
- Creating modal fields dynamically, including support for textarea resizing.  
- Saving and verifying member information through backend calls.  
- Applying blur to background when the modal is open.

## 5. Backend Components

The Google Apps Script backend handles the following server-side tasks:  
- Retrieving table headers from the Google Sheet.  
- Filtering data based on search criteria.  
- Returning empty datasets when no filters are applied.  
- Updating member data upon save.  
- Updating verification dates for members.

## 6. Data Flow

1. The user enters search criteria and clicks 'Search'.  
2. JavaScript collects filter values and determines if the search button should be enabled.  
3. The getFilteredData backend function is called via google.script.run.  
4. The backend filters and returns results to the frontend.  
5. The frontend renders results in the table with hyperlinks on the ID column.  
6. Clicking a hyperlink opens the modal with member details pre-filled.  
7. The user can edit fields, then click 'Save' or 'Verify'.  
8. The backend updates the Google Sheet and returns success.  
9. The modal closes, and the table refreshes.

## 7. Security Considerations

- All interactions occur within the authenticated Google Workspace environment.  
- Backend functions are restricted to necessary operations only.  
- User inputs are validated before being written to the sheet.

## 8. Future Enhancements

- Pagination support for large datasets.  
- Export to CSV functionality.  
- Additional validation rules for input fields.  
- Role-based access control for different editing permissions.