// service-worker.js

const CACHE\_NAME = "vista-charm-pwa-v1";

const OFFLINE\_URL = "/offline-fallback.html"; // Provide your offline HTML page if you have one.

self.addEventListener("install", (event) => {

// Pre-cache critical assets

event.waitUntil(

caches.open(CACHE\_NAME).then((cache) => {

return cache.addAll([

// You can add your critical routes, images, CSS, etc. here

// e.g. "/css/styles.css", "/js/main.js", "/offline-fallback.html"

]);

})

);

self.skipWaiting();

});

self.addEventListener("activate", (event) => {

// Cleanup old caches if needed

event.waitUntil(

caches.keys().then((cacheNames) => {

return Promise.all(

cacheNames.map((cache) => {

if (cache !== CACHE\_NAME) {

return caches.delete(cache);

}

})

);

})

);

self.clients.claim();

});

self.addEventListener("fetch", (event) => {

// Basic strategy: try network first, fallback to cache, then offline page

event.respondWith(

fetch(event.request)

.then((response) => {

// Optional: add the response to the cache for future offline use

const clone = response.clone();

caches.open(CACHE\_NAME).then((cache) => cache.put(event.request, clone));

return response;

})

.catch(() => {

return caches.match(event.request).then((cached) => {

return cached || caches.match(OFFLINE\_URL);

});

})

);

});

// Example push event listener for background push notifications

self.addEventListener("push", (event) => {

if (event.data) {

const data = event.data.json();

const title = data.title || "Vista Charm Notification";

const options = data.options || {};

event.waitUntil(self.registration.showNotification(title, options));

}

});

// Listen for messages from the main thread (e.g., show a notification)

self.addEventListener("message", (event) => {

if (event.data && event.data.type === "SHOW\_NOTIFICATION") {

const payload = event.data.payload;

event.waitUntil(

self.registration.showNotification(payload.title, payload.options)

);

}

});