Name:Sahil Motiramani Div:D15C Roll No.:35

Experiment:8

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the Fitness-app PWA.

Theory:

A **Service Worker** is a JavaScript file that runs in the background and acts as a network proxy to intercept and manage network requests. It enables key Progressive Web App (PWA) features like **offline support**, **asset caching**, **background sync**, and **faster load times**.
In the context of a **Fitness App**, a service worker can cache:

- Workout images and background assets
- Daily routine or schedule pages
- Fonts and icons
- Home screen content

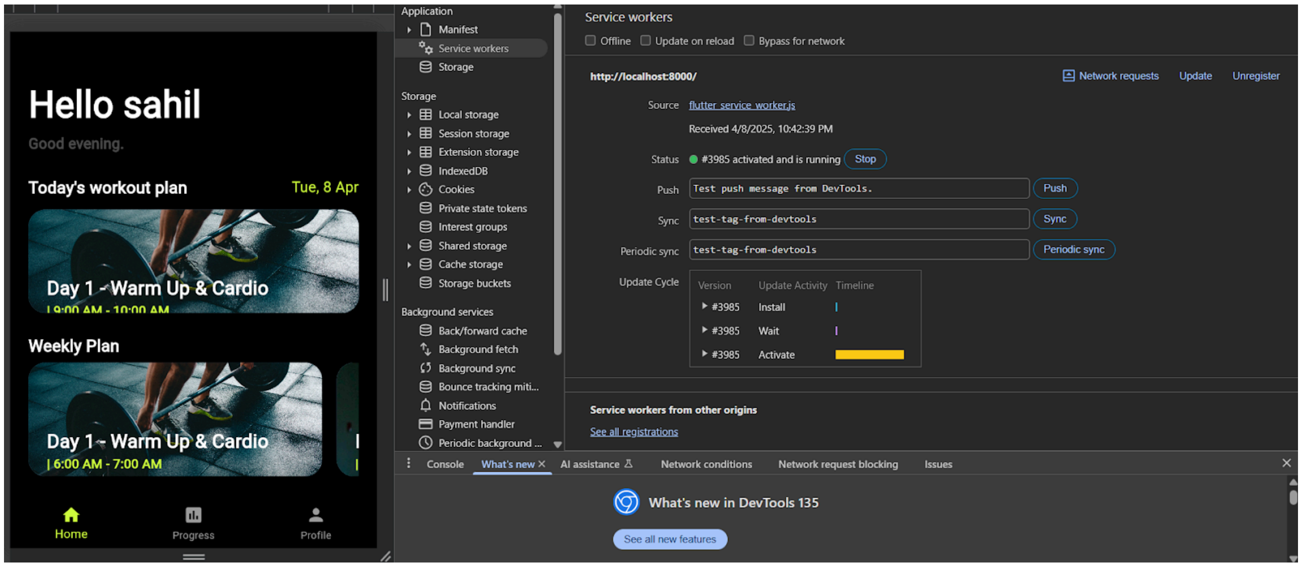
This ensures users can still access their workout routines and schedules even when offline or facing poor internet connectivity.
The service worker lifecycle includes:

1. **Registration** – You register the service worker in the main Dart-to-JS compiled HTML/JS file (index.html or main.dart.js).
2. **Installation** – Assets are cached during this phase.
3. **Activation** – Old caches are cleared and the new service worker takes control.

Using a flutter_service_worker.js file generated during flutter build web, you can control how resources are cached and fetched.

Output:

Name:Sahil Motiramani Div:D15C Roll No.:35



Service workers

Offline Update on reload Bypass for network

http://localhost:8000/ — Network requests Update Unregister

Source	flutter_service_worker.js
	Received 4/8/2025, 10:42:39 PM
Status	#3985 activated and is running Stop
Push	Test push message from DevTools. Push
Sync	test-tag-from-devtools Sync
Periodic sync	test-tag-from-devtools Periodic sync
Update Cycle	

Version	Update Activity	Timeline
#3985	Install	
#3985	Wait	
#3985	Activate	

Service workers from other origins

See all registrations

Conclusion:

By registering and coding a service worker in the **Fitness App PWA**, the app becomes more reliable, responsive, and capable of functioning without an internet connection. The service worker caches key UI elements, background images, and workout data to improve performance and offer a seamless experience to users. This is a critical step in making the fitness app a true PWA that can be installed and run like a native mobile app.