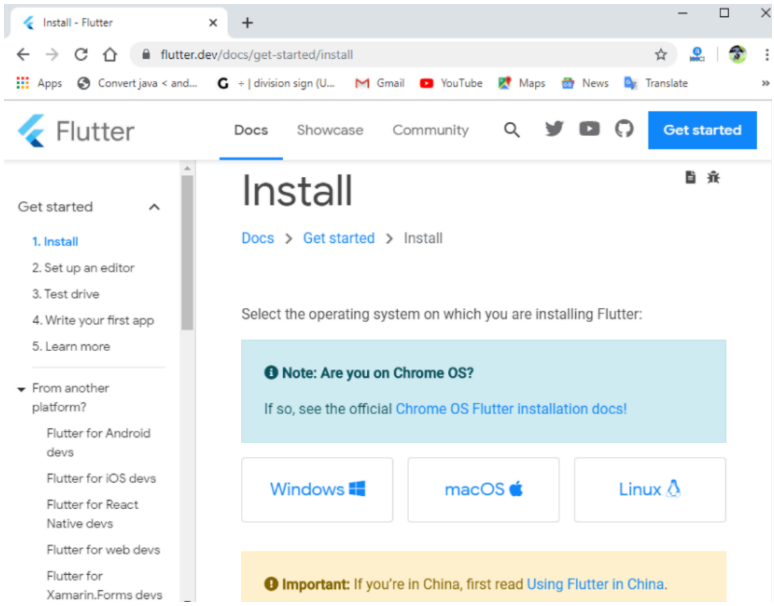
Experiment 1	
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

EXP 1: Installation and Configuration of Flutter Environment.
To install and configure the Flutter development environment on a local system.
This includes setting up Flutter SDK, Android Studio, and necessary plugins.
It enables developers to build and run Flutter applications on emulators or physical devices.



conclusion:The Flutter environment setup, including the SDK, Android Studio, and plugins, enables developers to build and run apps smoothly on emulators or devices, forming the foundation for efficient Flutter development.

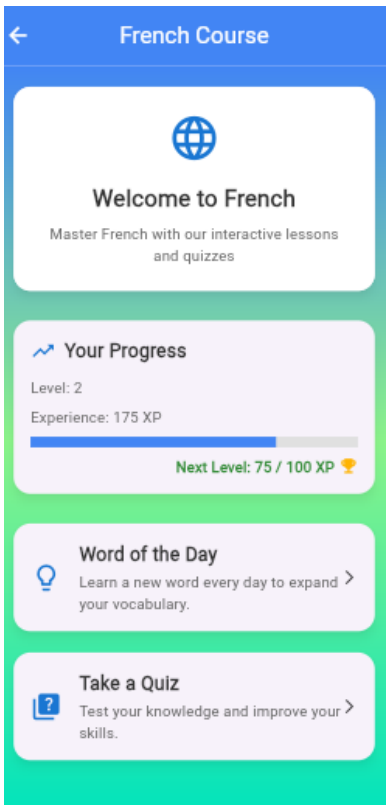
Github link: https://github.com/laukik86/language_app

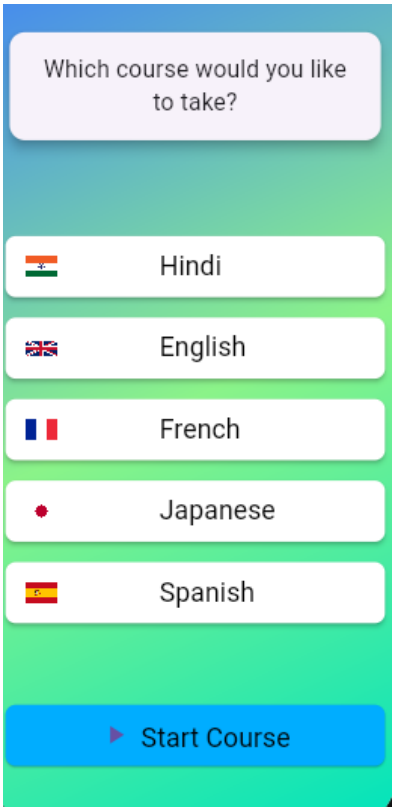
Experiment 2	
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

Experiment 02 : To design Flutter UI by including common widgets.

In this experiment, we created a simple user interface using common Flutter widgets like Text, Image, Container, Row, and Column. It helped us understand how widgets are the building blocks of Flutter apps and how they can be combined to design responsive UIs.

Screen Shots:





Conclusion:

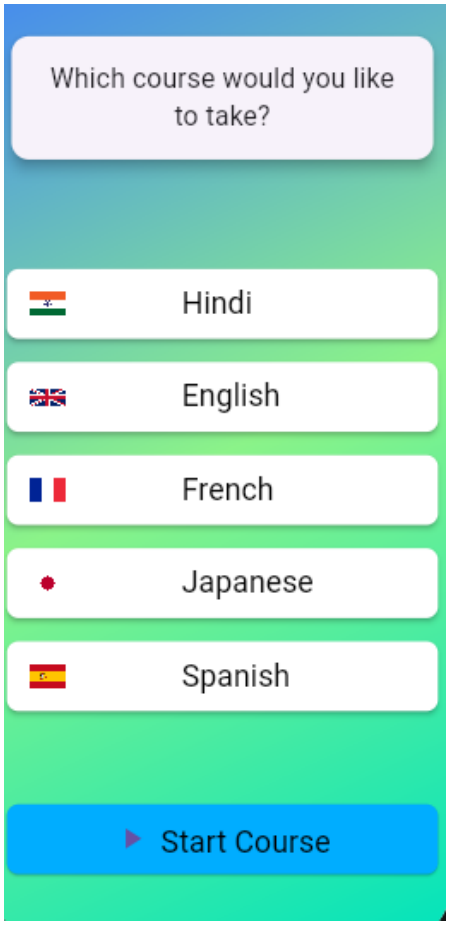
In this experiment, we successfully designed a Flutter user interface by incorporating commonly used widgets such as Text, Image, Row, Column, Container, ElevatedButton, and others. This hands-on experience provided a deeper understanding of how Flutter widgets work together to build responsive and visually appealing UIs. Mastering these basic widgets is essential for creating complex and interactive mobile applications using Flutter.

Experiment 3	
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

Experiment 03 : To include icons, images, fonts in Flutter app

In this experiment, we learned how to enhance the visual appeal of a Flutter app by adding custom icons, images, and fonts. This helps in creating a more engaging and personalized user interface that aligns with the app's branding and design goals.

ScreenShots:



Conclusion:

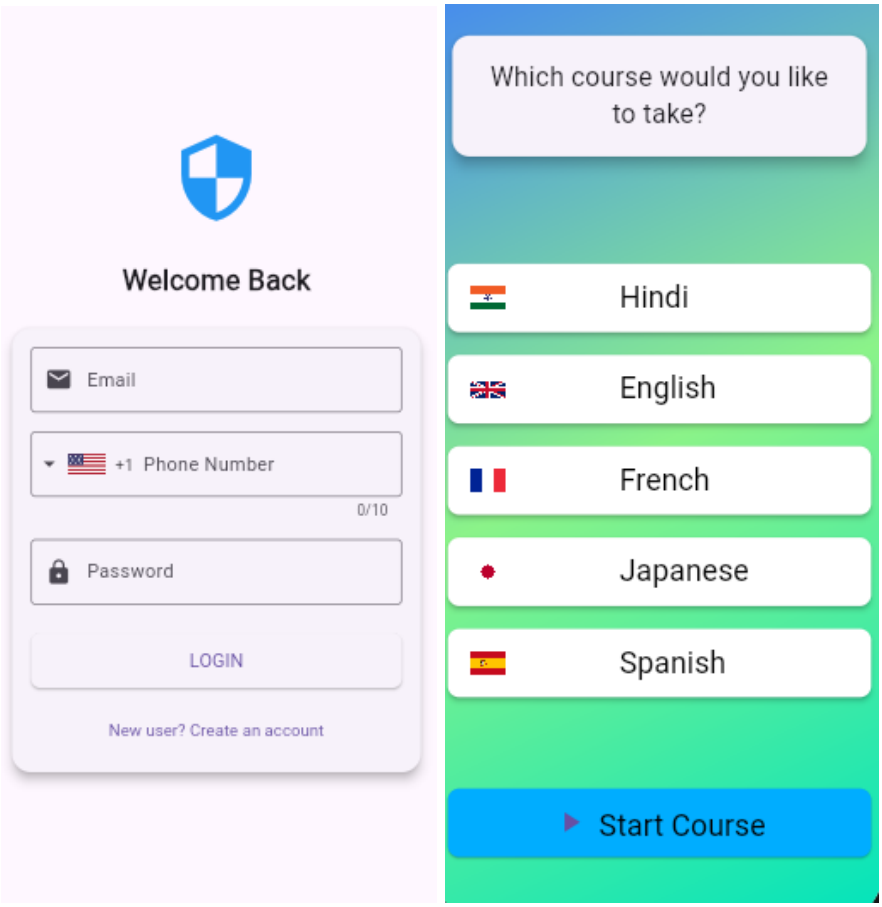
By successfully including icons, images, and custom fonts in the Flutter app, we gained practical knowledge of enhancing the app's visual design and user experience. This experiment highlighted the importance of media and typography in making apps more attractive and user-friendly.

Experiment 4	
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

Exp 4 To create an interactive Form using form widget

In this experiment, we created an interactive form in Flutter using Form, TextFormField, and ElevatedButton widgets. It helped us understand how to collect, validate, and manage user input effectively within a Flutter app.

ScreenShots:



Conclusion:

By building an interactive form, we learned how to handle user input, perform validations, and manage form state in Flutter. This experiment is essential for developing apps that require user data input, such as login, registration, or feedback forms.

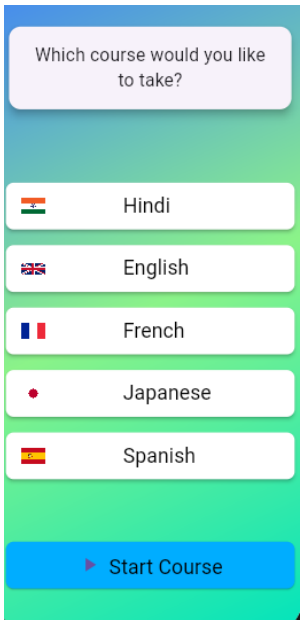
Experiment 5	
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

Exp 5 To apply navigation, routing and gestures in Flutter App

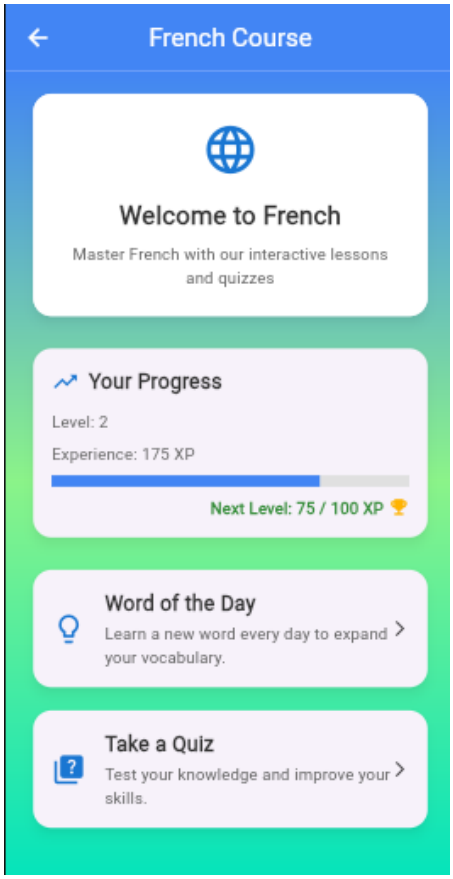
In this experiment, we implemented navigation and routing to switch between different screens in a Flutter app. We also added gesture detection to make the app more interactive and responsive to user actions.

Screenshots:

Before clicking submit:



After clicking the Start button:



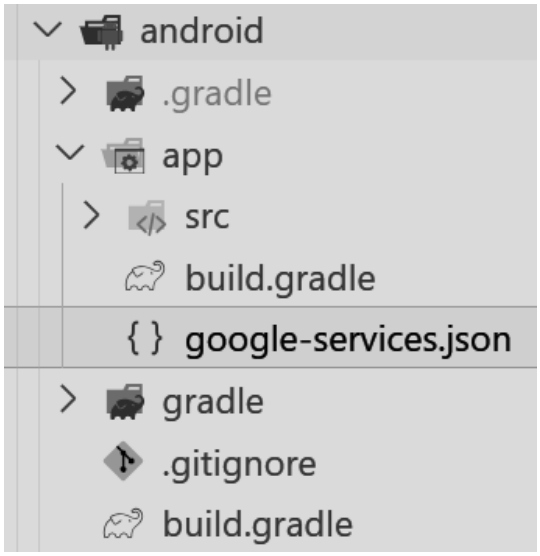
Conclusion:

This experiment helped us understand the concepts of navigation and routing in Flutter, allowing smooth transitions between screens. Additionally, by using gesture detectors, we enhanced user interaction, making the app more dynamic and user-friendly.

	Experiment 6
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

Exp6: How To Set Up Firebase with Flutter for iOS and Android Apps

In this experiment, we connected a Flutter app to Firebase to use cloud-based features like authentication, database, and analytics. It involved setting up Firebase SDKs, adding configuration files, and initializing Firebase in the app.



Conclusion:

This experiment provided hands-on experience in integrating Firebase with a Flutter app for both Android and iOS. It enables the use of powerful backend features like authentication, Firestore, and real-time updates, which are essential for modern app development.

	Experiment 7
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

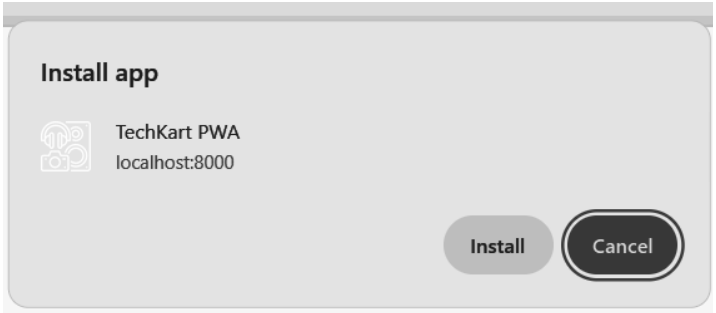
Experiment No. 7
To write meta data of your language app PWA

Aim:- To write meta data of your language app PWA in a Web app manifest file to enable “add to homescreen feature”.

ScreenShots:

{} manifest.json > ...

```
{
    "name": "LangLearn - Language Learning App",
    "short_name": "LangLearn",
    "start_url": ".",
    "display": "standalone",
    "background_color": "#ffffff",
    "theme_color": "#6200EE",
    "description": "Learn languages with vocabulary quizzes and audio practice!",
    "icons": [
      {
        "src": "icons/icon-192.png",
        "sizes": "192x192",
        "type": "image/png"
      },
      {
        "src": "icons/icon-512.png",
        "sizes": "512x512",
        "type": "image/png"
      }
    ]
}
```

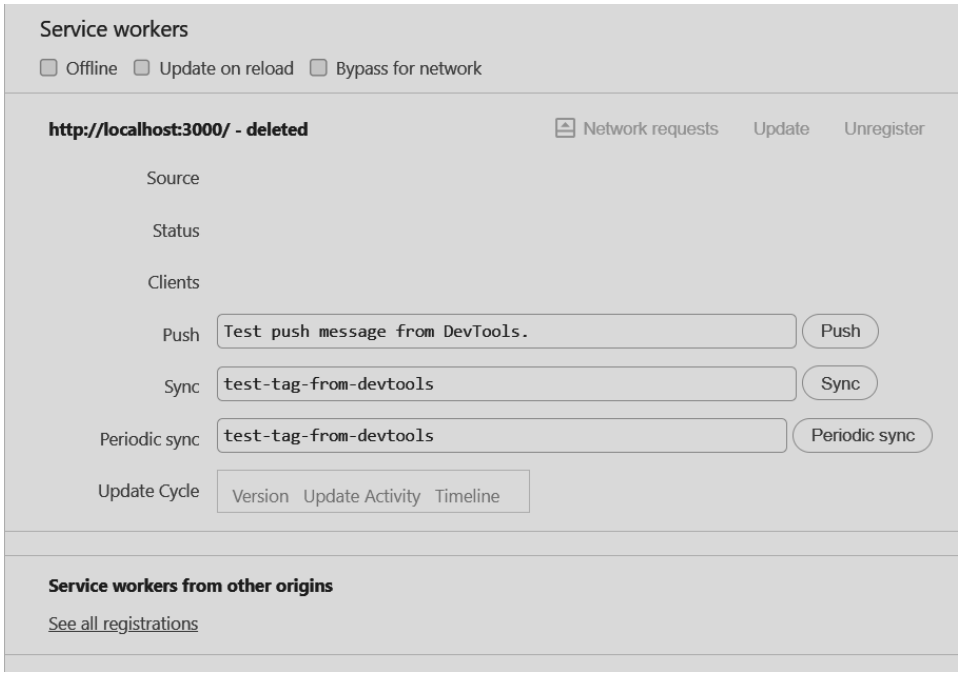


Conclusion:
By adding metadata in the manifest.json file, we enabled our Ecommerce PWA to support the "Add to Home Screen" feature, improving user accessibility and enhancing the app-like experience on mobile devices.

	Experiment 8
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

Experiment No. 8
Aim:
To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

In this experiment, we wrote and registered a service worker for our E-commerce PWA. The service worker handles caching and runs in the background, allowing the app to work offline and load faster by intercepting network requests.




```
Service Worker registered: http://localhost:3000/
```

```
// main.js
if ('serviceWorker' in navigator) {
    window.addEventListener('load', () => {
      navigator.serviceWorker
        .register('/service-worker.js')
        .then(reg => console.log("Service Worker registered:", reg.scope))
        .catch(err => console.log("Service Worker registration failed:", err));
    });
  }
```

Conclusion:

By coding and registering a service worker, we enabled offline support and caching capabilities for the E-commerce PWA. The successful installation and activation of the service worker enhanced the app's reliability and performance, especially in low or no internet connectivity.

	Experiment 9
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

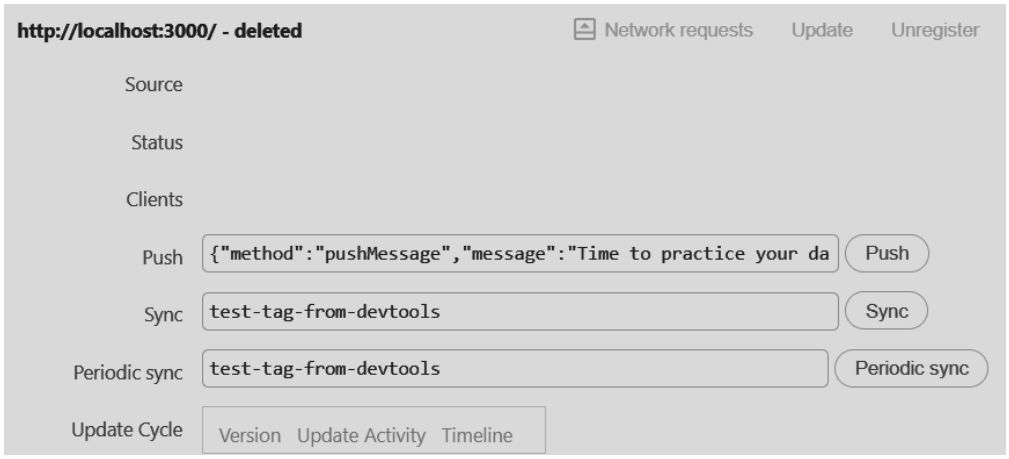
Experiment No. 9

Aim:

To implement Service Worker events like fetch, sync, and push for the E-commerce PWA.

Explanation :

In this experiment, we added support for service worker events such as fetch to cache and serve requests, sync to handle background synchronization, and push to display notifications. These events make the PWA more efficient, reliable, and interactive.



```
// Background sync registration (simulate saving vocab progress)
if ('serviceWorker' in navigator && 'SyncManager' in window) {
    navigator.serviceWorker.ready.then(registration => {
      return registration.sync.register('sync-vocab-progress');
    }).then(() => {
      console.log("Background Sync Registered ☑");
    }).catch(console.error);
  }

  // Ask for push notification permission
  Notification.requestPermission().then(permission => {
    if (permission === "granted") {
      console.log("🔔 Notification permission granted.");
    }
  });
```

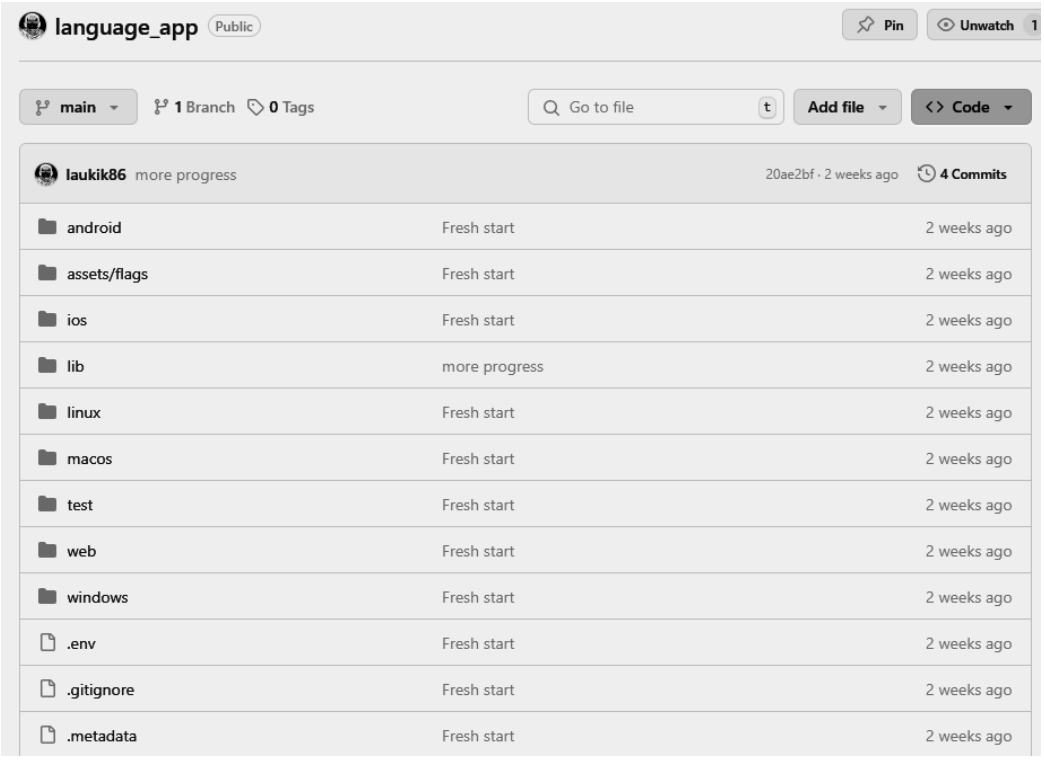
Conclusion:

By implementing fetch, sync, and push events in the service worker, we improved the performance, offline support, and user engagement of the E-commerce PWA. These features enable real-time updates, background tasks, and seamless user experiences even with limited connectivity.

	Experiment 10
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

Experiment No. 10
Aim:
To study and implement the deployment of the E-commerce PWA to GitHub Pages.

Explanation :
In this experiment, we deployed our E-commerce Progressive Web App (PWA) to GitHub Pages using the gh-pages package. This allowed us to host and share our PWA publicly through a GitHub repository.

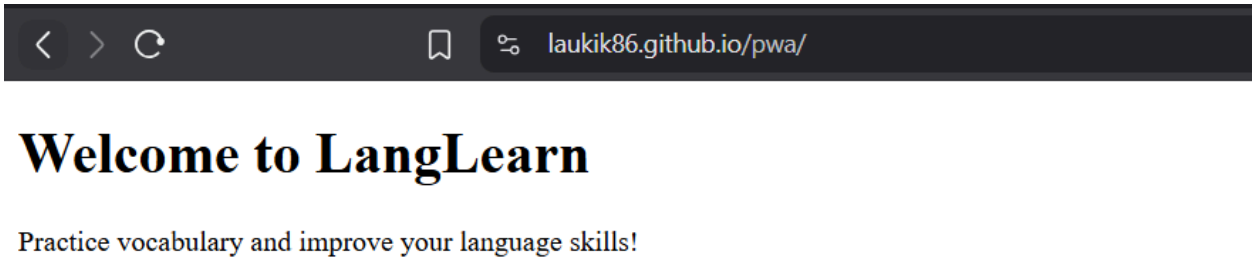


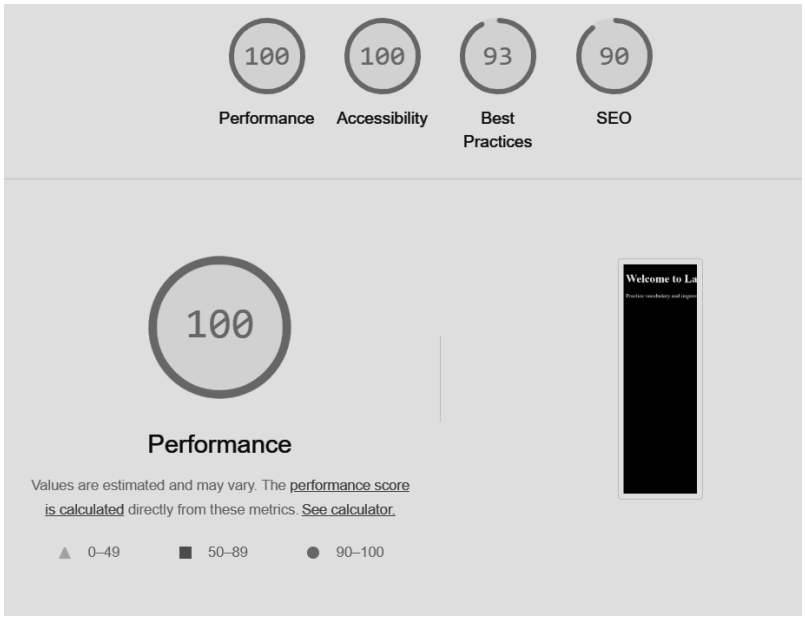
https://github.com/laukik86/language_app
https://github.com/laukik86/pwa

Conclusion:
By successfully deploying the E-commerce PWA to GitHub Pages, we made the app accessible online through a live URL. This experiment helped us understand the deployment process and the importance of version control and static hosting for PWAs.

	Experiment 11
Name	Laukik Padgaonkar
Roll No	37
DOP	
DOS	
Sign	
Grade	

Experiment No. 11
Aim:
To use Google Lighthouse PWA Analysis Tool to test the PWA functioning.
Explanation :
In this experiment, we used Google Lighthouse to audit and analyze the performance, accessibility, best practices, and PWA compliance of our E-commerce app. It provided a detailed report with scores and suggestions for improvement.
Hosting link: https://laukik86.github.io/pwa/





Conclusion:

By using the Google Lighthouse tool, we evaluated the effectiveness and quality of our E-commerce PWA. The audit helped us identify areas of improvement, ensuring better performance, offline support, and overall user experience.