

Experiment No 1			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

Aim: Installation and Configuration of Flutter Environment.

Description: This experiment focuses on setting up the Flutter development environment to enable the creation of high-performance apps for both Android and iOS platforms. It includes downloading and installing the Flutter SDK, configuring an IDE such as Android Studio or Visual Studio Code, setting up an Android emulator or a physical device for testing, and running a sample Flutter application. This setup ensures a complete and ready-to-use environment for efficient app development, testing, and deployment using Flutter.

Output:

```
Flutter Console

#####  ##      ##      ## ##### ##### ##### #####
##      ##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##      ##
#####  ##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##      ##
##      ##### #####      ##      ##      ##      ##

WELCOME to the Flutter Console.
=====

Use the console below this message to interact with the "flutter" command.
Run "flutter doctor" to check if your system is ready to run Flutter apps.
Run "flutter create <app_name>" to create a new Flutter project.

Run "flutter help" to see all available commands.

Want to use an IDE to interact with Flutter? https://flutter.dev/ide-setup/

Want to run the "flutter" command from any Command Prompt or PowerShell window?
Add Flutter to your PATH: https://flutter.dev/setup-windows/#update-your-path

=====
```



Conclusion: After successfully installing and configuring the Flutter environment, you are now ready to begin developing cross-platform mobile applications. This setup enables you to write code once and deploy it on both Android and iOS devices, significantly speeding up the development process and improving efficiency. It lays a solid foundation for building visually appealing, high-performance apps using a single codebase, empowering developers to create seamless user experiences across multiple platforms.

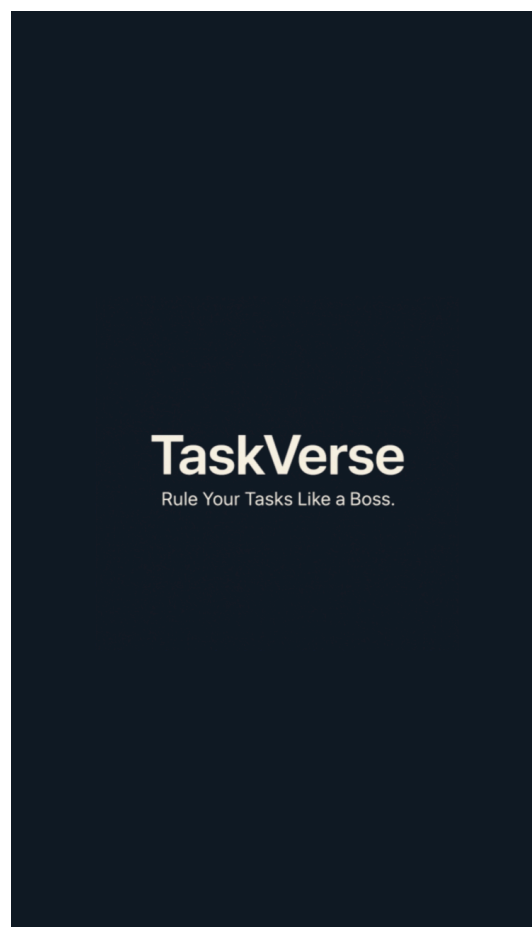
Experiment No 2			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

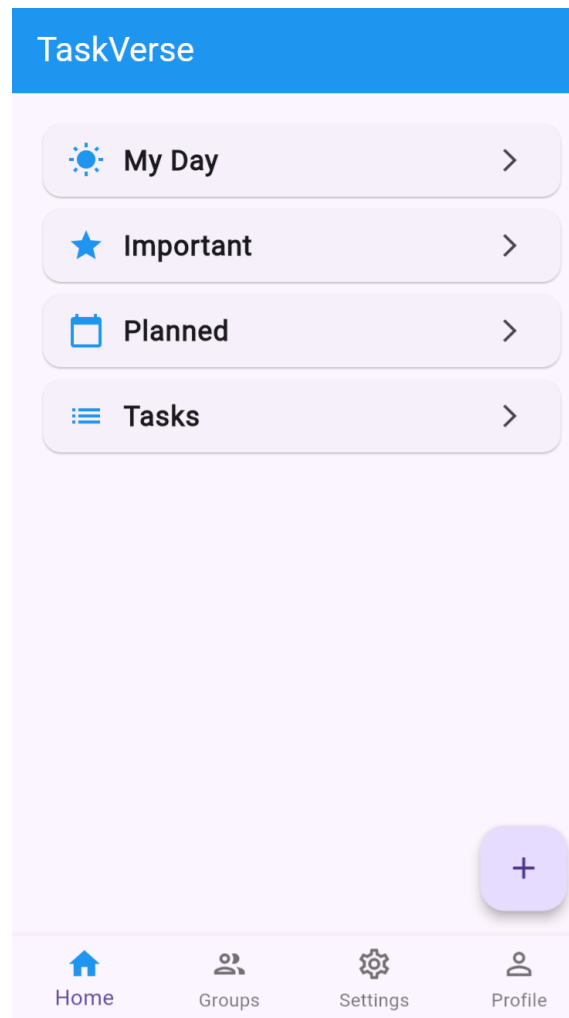
Aim: To design Flutter UI by including common widgets.

GitHub Link: <https://github.com/BKCODE2003/ToDo>

Description: This experiment focuses on designing user interfaces in Flutter using commonly used widgets. Flutter provides a rich set of pre-built widgets that help developers create responsive and visually appealing UIs. In this task, you will explore and implement essential Flutter widgets such as Container, Row, Column, Text, ListView, Button, Images and more. The goal is to understand the structure of Flutter's widget tree and learn how to build flexible, interactive, and reusable UI components for mobile applications.

Output:





Conclusion: By designing the Flutter UI using common widgets, we have gained hands-on experience in building structured, responsive, and visually engaging user interfaces. Understanding and implementing widgets like Container, Row, Column, Text, and ListView lays the groundwork for more complex UI designs. This knowledge is essential for developing interactive and user-friendly mobile applications using Flutter's widget-based architecture.

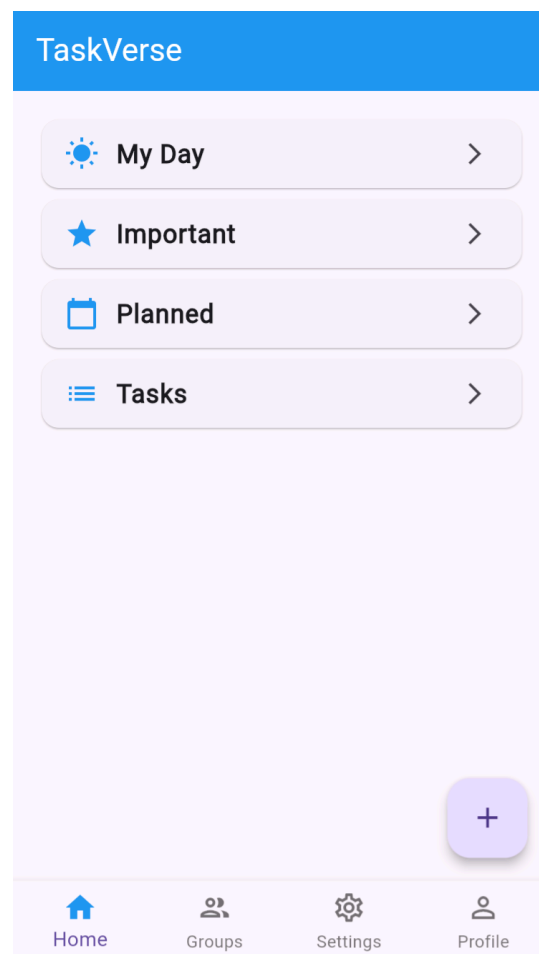
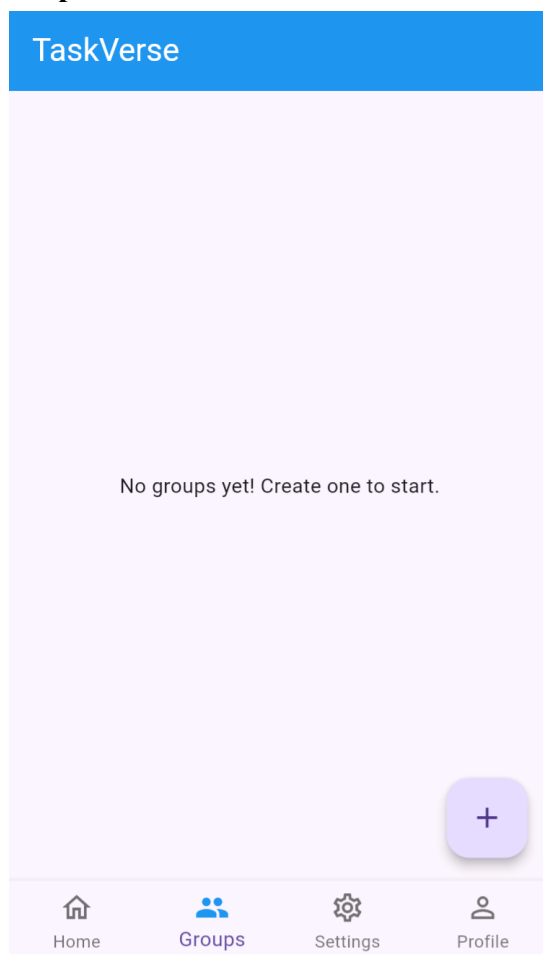
Experiment No 3			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

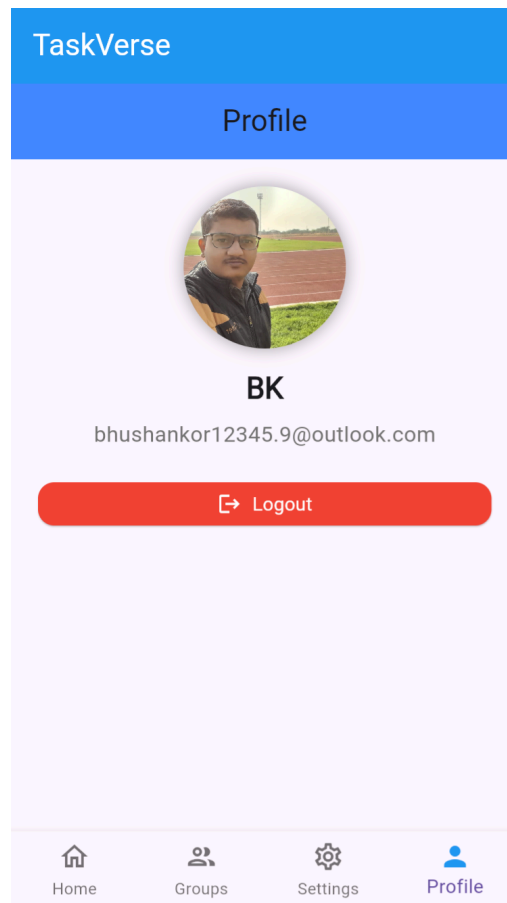
Aim: To include icons, images, fonts in Flutter app.

GitHub Link: <https://github.com/BKCODE2003/ToDo>

Description: This experiment focuses on enhancing the visual appeal and user experience of a Flutter application by including icons, images, and custom fonts. You will learn how to use built-in icons with the Icons class, add local and network images using the Image widget, and customize text appearance by integrating external font files. This setup allows for greater design flexibility and helps in creating more personalized and visually rich applications. Proper asset management and configuration in the pubspec.yaml file will also be covered as part of this task.

Output:





Conclusion: By successfully including icons, images, and custom fonts in a Flutter application, you have learned how to enhance the visual design and branding of your app. Understanding how to manage assets and configure them in the pubspec.yaml file is essential for creating a polished and engaging user interface. This knowledge enables you to build more dynamic and visually appealing applications tailored to user preferences and design standards.

Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

Experiment No 4			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

Aim: To create an interactive Form using a form widget.

GitHub Link: <https://github.com/BKCODE2003/ToDo>

Description: This experiment focuses on creating an interactive form in a Flutter application using the Form widget. You will learn how to collect and validate user input through commonly used form fields such as TextFormField, DropdownButtonFormField, and checkboxes. The experiment also covers implementing validation logic, managing form state with GlobalKey, and handling form submission. Creating interactive forms is essential for building apps that require user input, such as login screens, registrations, and feedback forms.

Output:

Create Account

Username
Bhushan

Email
bhushankor12345.9@gmail.com

Password
.....

Confirm Password
Abcd@12#\$56

Sign Up

Already have an account? Login

Welcome, Innovative Spirit

Email
bhushankor12345.9@gmail.com

Password
.....


Login


Don't have an account? Sign Up


← Add Task


Title


Description


Due Date  Select a date

Due Time  Select a time

Targeted Time  0 hrs 30 min

Use Default Reminders (1 hr & 1 day before) 

Add Custom Reminder 

Mark as Important 

Save Task

← Add Task

Title

Test 1

Description

Just Test

Due Date

2025-04

Due Time

11:51 P

Targeted Time

0 hrs 30 min

Targeted Time

30 min ▼

ne

0 h

1 h

2 h

3 h

4 h

5 h

6 h

7 h

8 h

9 h

10 h

11 h

Conclusion: By creating an interactive form using the Form widget in Flutter, we have learned how to efficiently collect, validate, and manage user input within a mobile application. Understanding form fields, validation logic, and state management enables you to build robust and user-friendly interfaces for features like registration, login, and data entry. This forms an essential part of developing interactive and responsive Flutter applications.

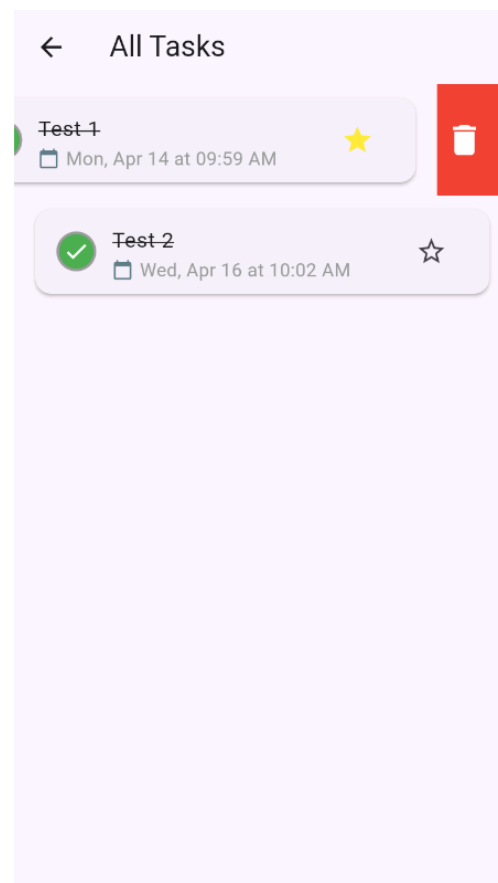
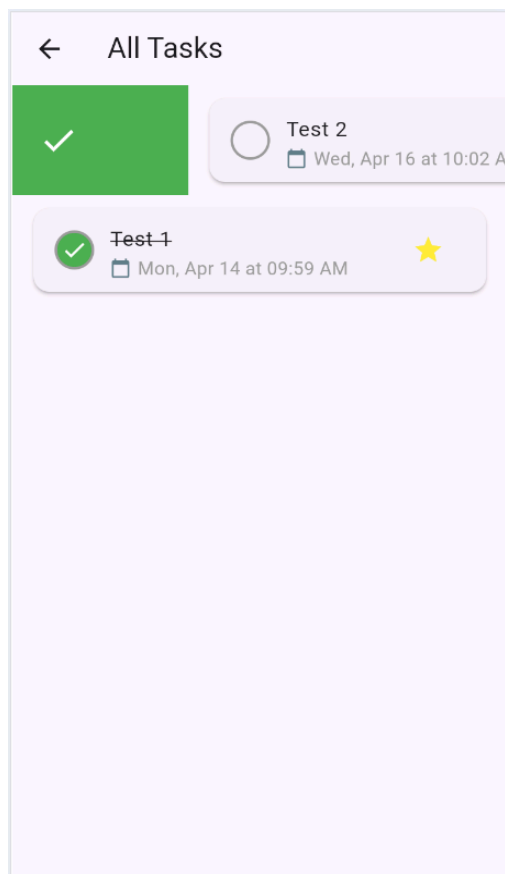
Experiment No 5			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

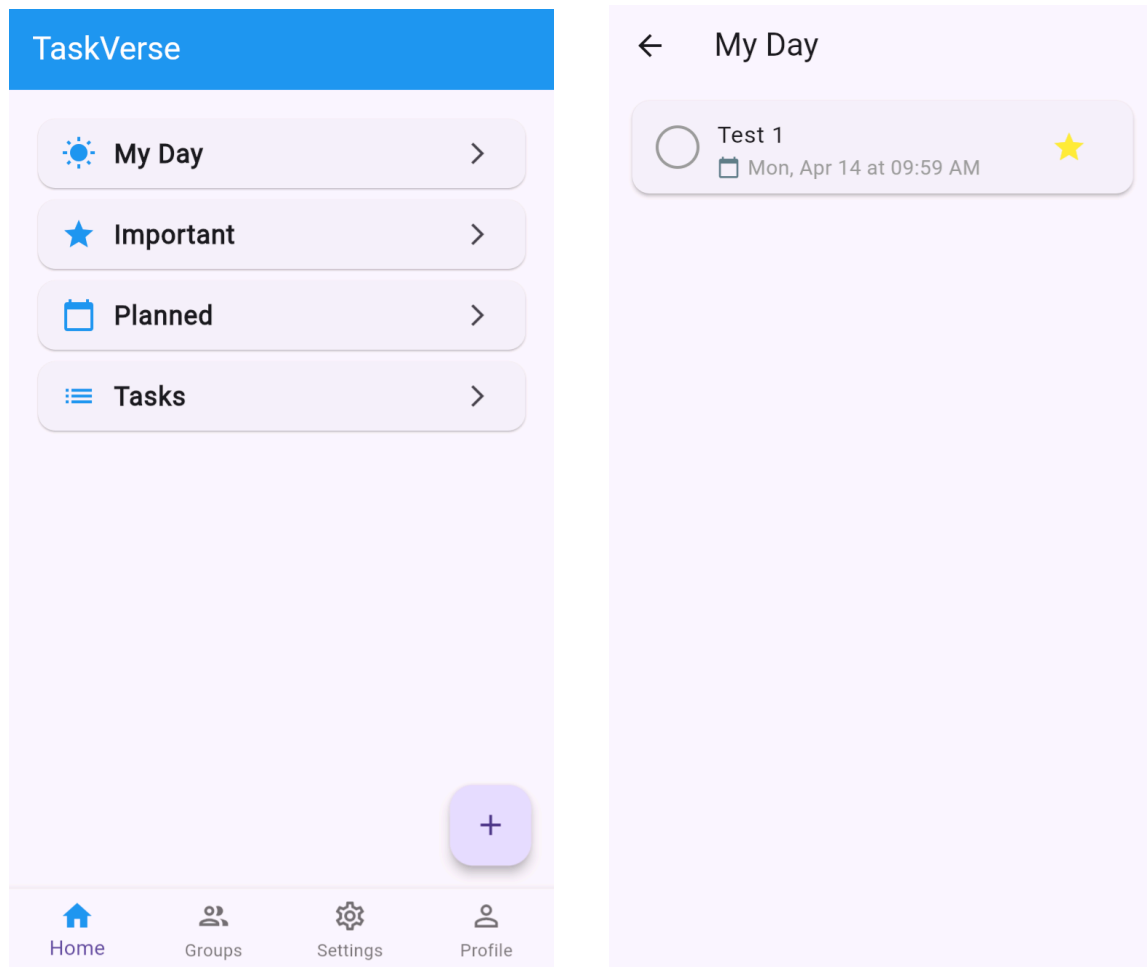
Aim: To apply navigation, routing and gestures in Flutter App.

GitHub Link: <https://github.com/BKCODE2003/ToDo>

Description: This experiment focuses on implementing navigation, routing, and gesture handling in a Flutter application to enhance user interaction and app flow. You will learn how to navigate between different screens using Navigator, define named routes, and pass data between pages. Additionally, gesture detectors such as GestureDetector and InkWell will be used to respond to user interactions like taps, swipes, and long presses. Mastering these concepts is crucial for building multi-screen applications with smooth transitions and interactive user experiences.

Output:





Conclusion: By applying navigation, routing, and gesture handling in a Flutter application, we have learned how to create smooth transitions between screens and build interactive user experiences. Understanding how to use Navigator, manage routes, and respond to gestures is essential for developing dynamic, multi-screen applications. These features play a key role in improving app usability and enabling intuitive user interactions.

Experiment No 6			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

Aim: To Connect Flutter UI with fireBase database.

GitHub Link: <https://github.com/BKCODE2003/ToDo>

Description: This experiment focuses on connecting a Flutter application's user interface (UI) with a Firebase database to enable real-time data storage and retrieval. You will learn how to integrate Firebase into a Flutter project, configure the required dependencies, and perform basic operations such as adding, reading, updating, and deleting data (CRUD). By connecting the UI with Firebase, the app becomes dynamic and data-driven, allowing for seamless interaction between the user interface and the backend. This is essential for building scalable and responsive applications.

Output:

← Add Task

Title

Description

Due Date

Select a date

Due Time

Select a time

Targeted Time

0 hrs 30 min

Use Default Reminders (1 hr & 1 day before)

Add Custom Reminder

Mark as Important

Save Task

← Task Details

Title

Test 1

Description

Just Test

Due Date

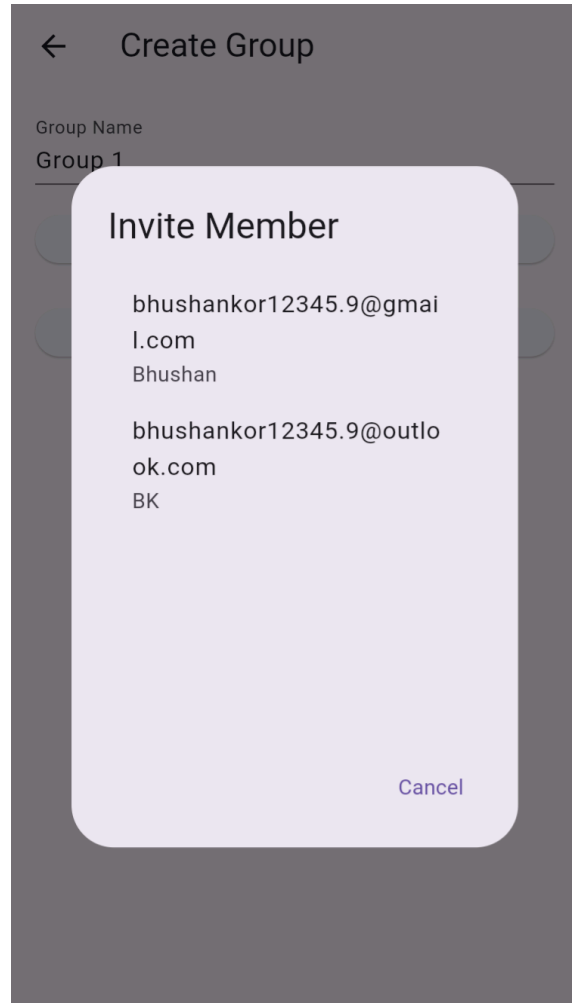
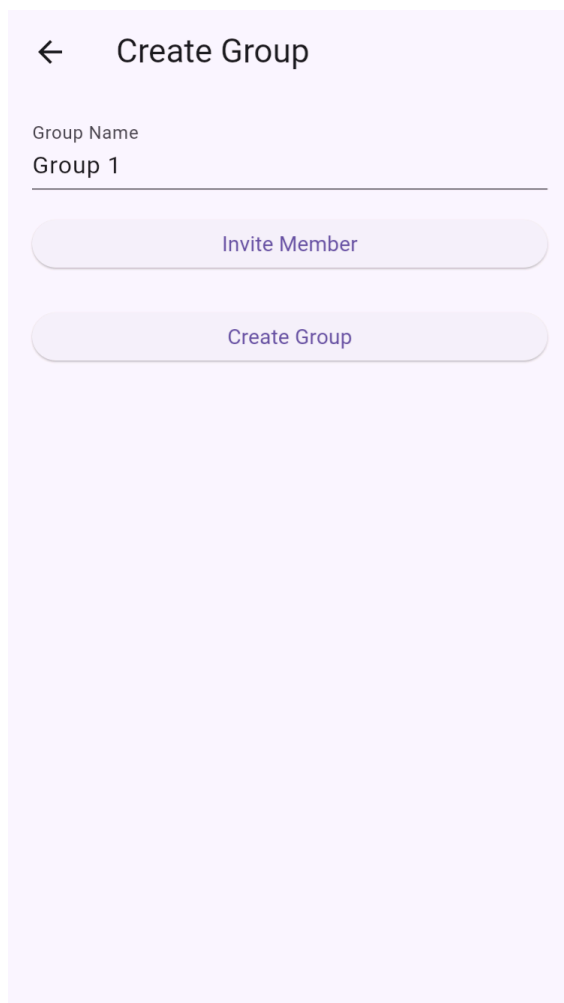
Mon, Apr 14

Due Time

9:59 AM

Mark as Important

Update Task



Conclusion: By successfully connecting the Flutter UI with the Firebase database, you have learned how to create dynamic and data-driven applications. This integration allows real-time data synchronization between the app and the backend, enabling functionalities such as data storage, retrieval, and updates. Understanding how to connect Flutter with Firebase is essential for developing modern applications that require persistent data, user management, and seamless cloud connectivity.

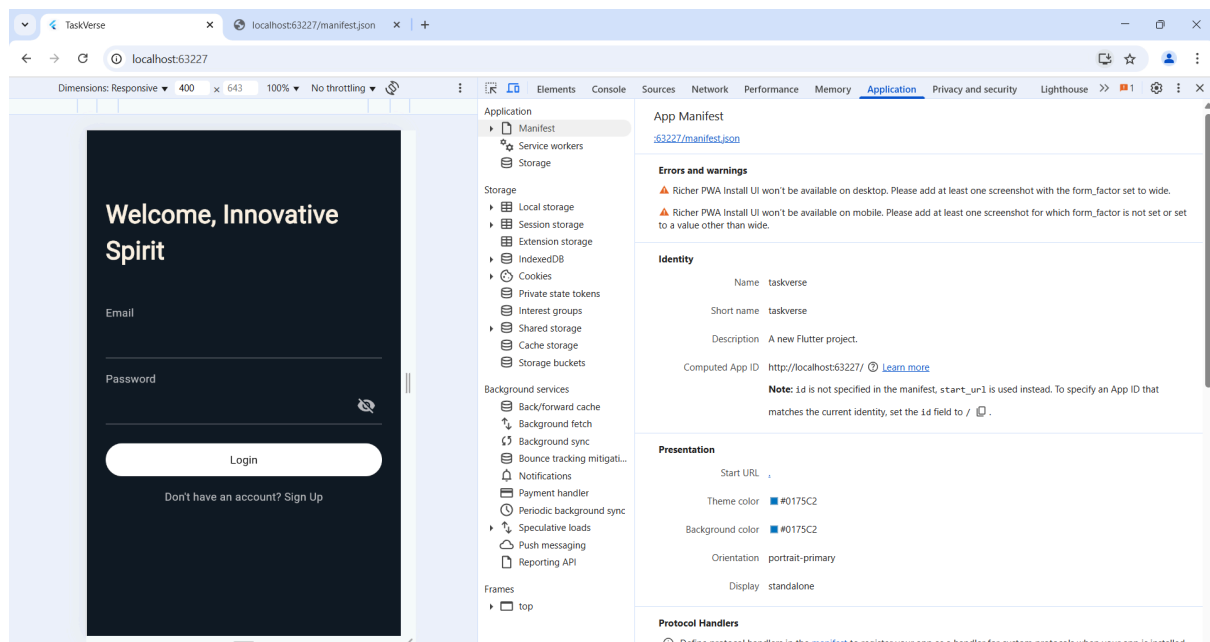
Experiment No 7			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

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

GitHub Link: https://github.com/BKCODE2003/Flutter_PWA

Description: In this experiment, we design and configure a Web App Manifest file for an ToDo Progressive Web App (PWA). The manifest defines key metadata such as the app's name, short name, icon set, start URL, display mode (e.g., standalone or fullscreen), background color, and theme color. This file is linked to the main HTML page, allowing modern browsers to recognize it and offer users the "Add to Home Screen" option. This enhances the app's accessibility and provides a native-like experience, improving user engagement and usability.

Output:



Conclusion: By successfully creating and linking the Web App Manifest file, the browser can identify the ToDo PWA and prompt users with the option to install it on their device's home screen. This improves the app's accessibility, boosts user engagement, and delivers a more seamless, native-like experience across various platforms.

Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

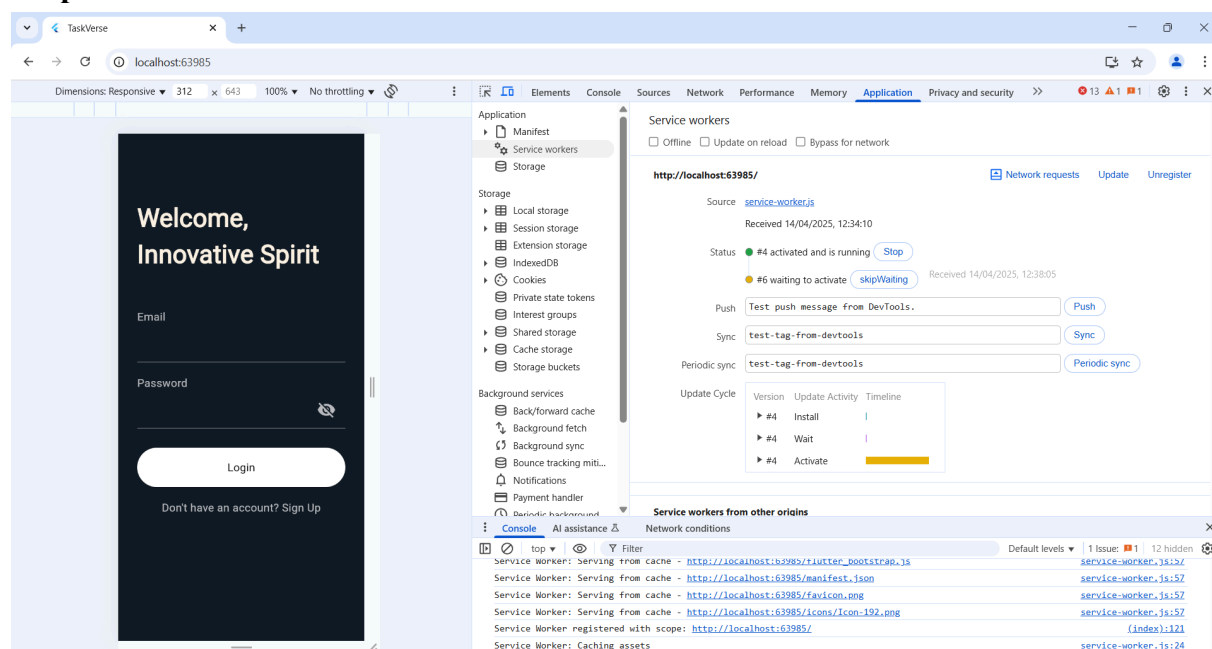
Experiment No 8			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

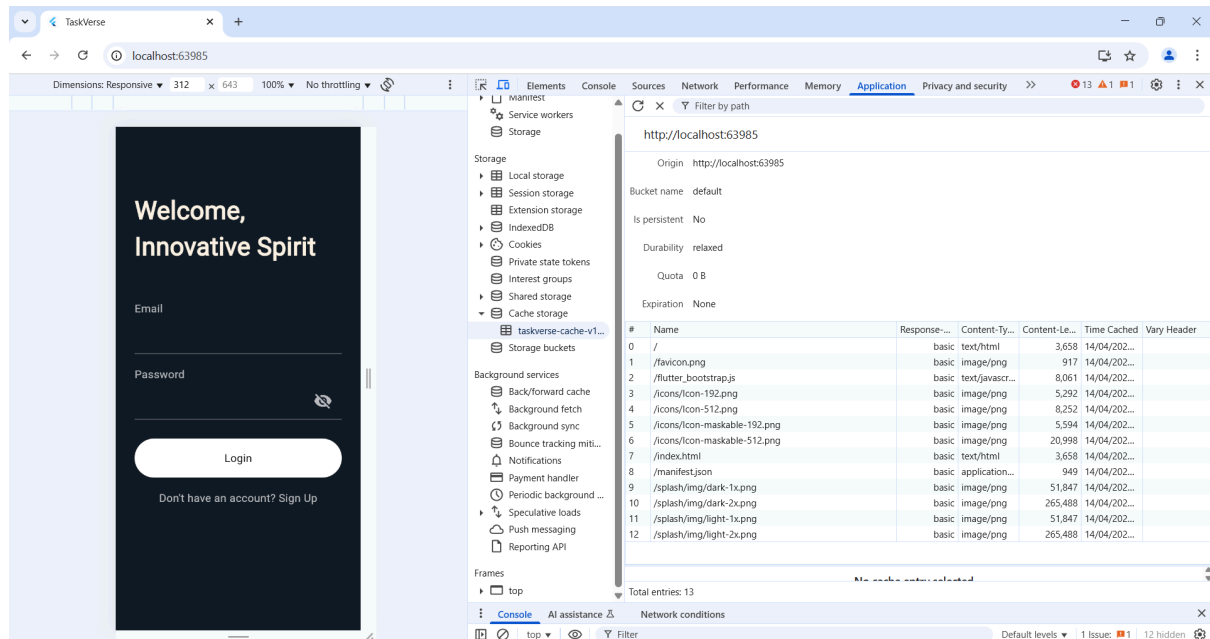
Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the ToDo PWA.

GitHub Link: https://github.com/BKCODE2003/Flutter_PWA

Description: This experiment involves coding and registering a service worker for an ToDo Progressive Web App (PWA), and completing its install and activation lifecycle. A service worker is a background script that enables advanced PWA features such as offline access, background sync, and push notifications. In this task, you will create a service worker JavaScript file, register it in your main HTML or JavaScript entry point, and implement the install and activate events. This setup is essential for enabling caching, improving performance, and delivering a more reliable user experience even with limited or no internet connectivity.

Output:





Conclusion: By successfully coding and registering a service worker, and completing its installation and activation process, we have enabled core Progressive Web App functionalities such as offline support and background processing. This significantly enhances the reliability and performance of the eCommerce PWA, allowing it to function smoothly even in low or no network conditions. Implementing a service worker is a crucial step toward delivering a fast, resilient, and app-like experience to users across different platforms.

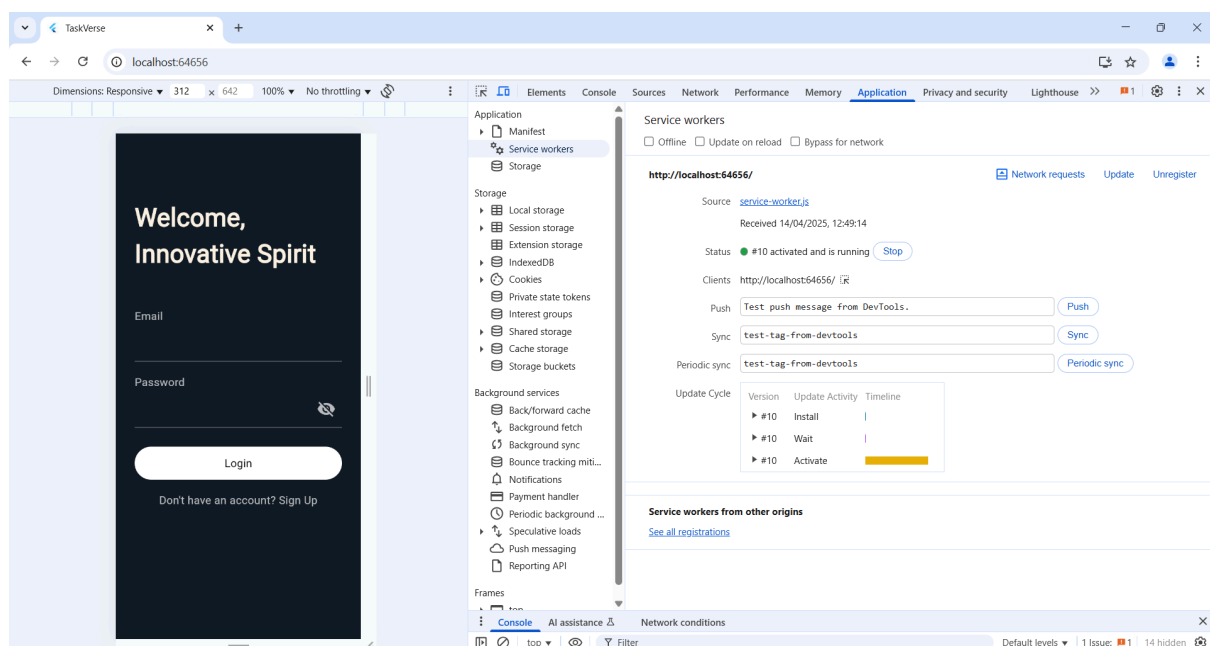
Experiment No 9			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

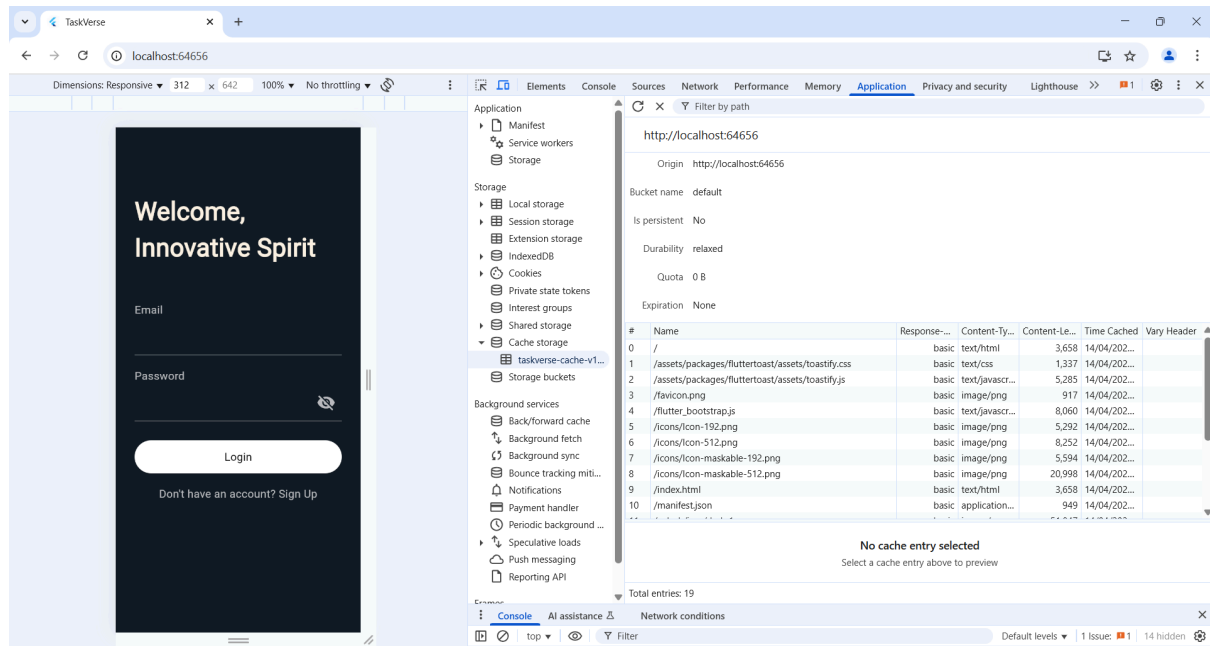
Aim: To implement Service worker events like fetch, sync and push for ToDo PWA.

GitHub Link: https://github.com/BKCODE2003/Flutter_PWA

Description: This experiment focuses on implementing key service worker events—fetch, sync, and push—to enhance the functionality of an ToDo Progressive Web App (PWA). The fetch event is used to intercept network requests and serve cached content, enabling offline access and faster load times. The sync event allows background synchronization of data when connectivity is restored, ensuring smooth user experiences even after offline interactions. The push event enables the app to receive and display notifications, keeping users informed about updates like new offers or order statuses. Implementing these events ensures the PWA is reliable, efficient, and engaging under various network conditions.

Output:





Conclusion: By implementing service worker events such as fetch, sync, and push, we have enhanced the reliability, responsiveness, and interactivity of the eCommerce PWA. These features allow the app to serve cached content offline, sync data in the background, and send real-time notifications to users. Together, they contribute to a seamless and engaging user experience, making the PWA behave more like a native application, even under challenging network conditions.

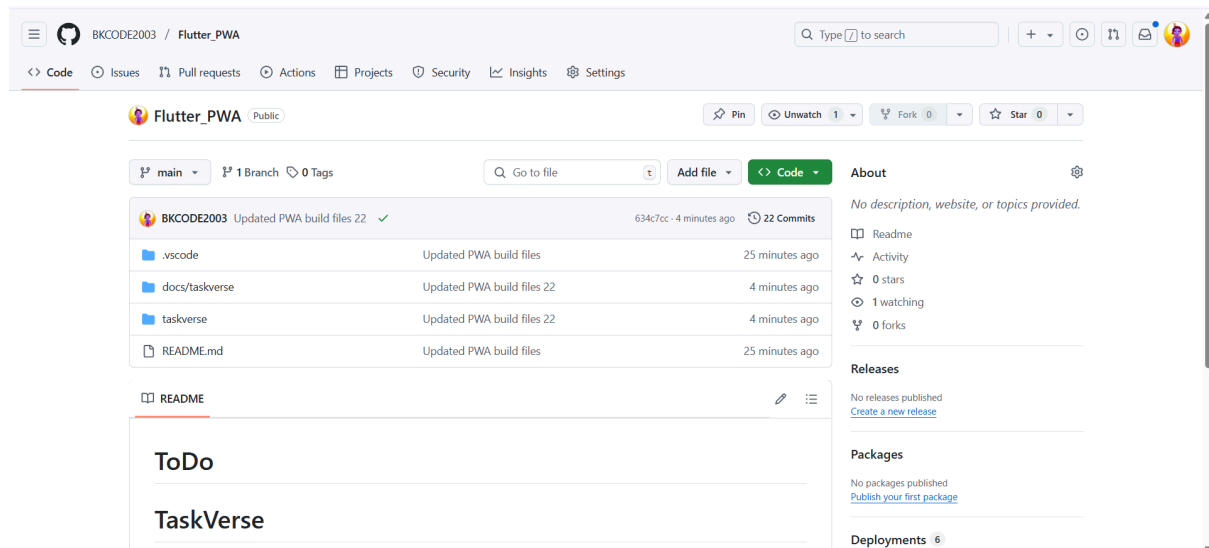
Experiment No 10			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

Aim: To study and implement deployment of ToDo PWA to GitHub Pages.

GitHub Link: https://github.com/BKCODE2003/Flutter_PWA

Description: The objective of this experiment is to explore and carry out the deployment of a ToDo Progressive Web App (PWA) on GitHub Pages. This includes generating a production-ready build using the appropriate commands, configuring essential files such as index.html, manifest.json, and the service worker to ensure correct path references, and setting the base href in index.html. After configuration, the project is pushed to a GitHub repository and deployed via GitHub Pages. This process enables the PWA to be accessible through any browser and installable on a user's home screen.

Output:



The screenshot displays the GitHub interface for the repository 'Flutter_PWA' owned by 'BKCODE2003'. The repository is public and has 0 stars, 1 watcher, and 0 forks. The commit history shows several updates to PWA build files. The README file is visible, containing the text 'ToDo' and 'TaskVerse'. The right sidebar shows the repository's status, including no releases or packages published.

The screenshot shows the GitHub Pages configuration interface. On the left, a sidebar lists various repository settings: General, Access, Collaborators, Moderation options, Code and automation (with sub-items: Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces), Pages (selected), Security, Code Security, and Deploy keys. The main content area is titled 'GitHub Pages' and includes the following sections:

- General:** A message stating 'Your site is live at https://bkcode2003.github.io/Flutter_PWA/' and 'Last deployed by BKCODE2003 4 minutes ago'. A 'Visit site' button is present.
- Build and deployment:**
 - Source:** A dropdown menu set to 'Deploy from a branch'.
 - Branch:** A message stating 'Your GitHub Pages site is currently being built from the /docs folder in the main branch. [Learn more about configuring the publishing source for your site.](#)' Below this, a dropdown menu is set to 'main' and a folder dropdown is set to '/docs', with a 'Save' button.
- Custom domain:** A section with a heading and a link to 'Learn how to add a Jekyll theme to your site.'

Link: https://bkcode2003.github.io/Flutter_PWA/taskverse/

Conclusion: Deploying the ToDo PWA to GitHub Pages makes the application publicly accessible on the internet, offering an easy way to share and test it. This approach ensures a stable and consistent hosting environment. A successful deployment highlights the capability to host and manage modern web applications using version control platforms such as GitHub.

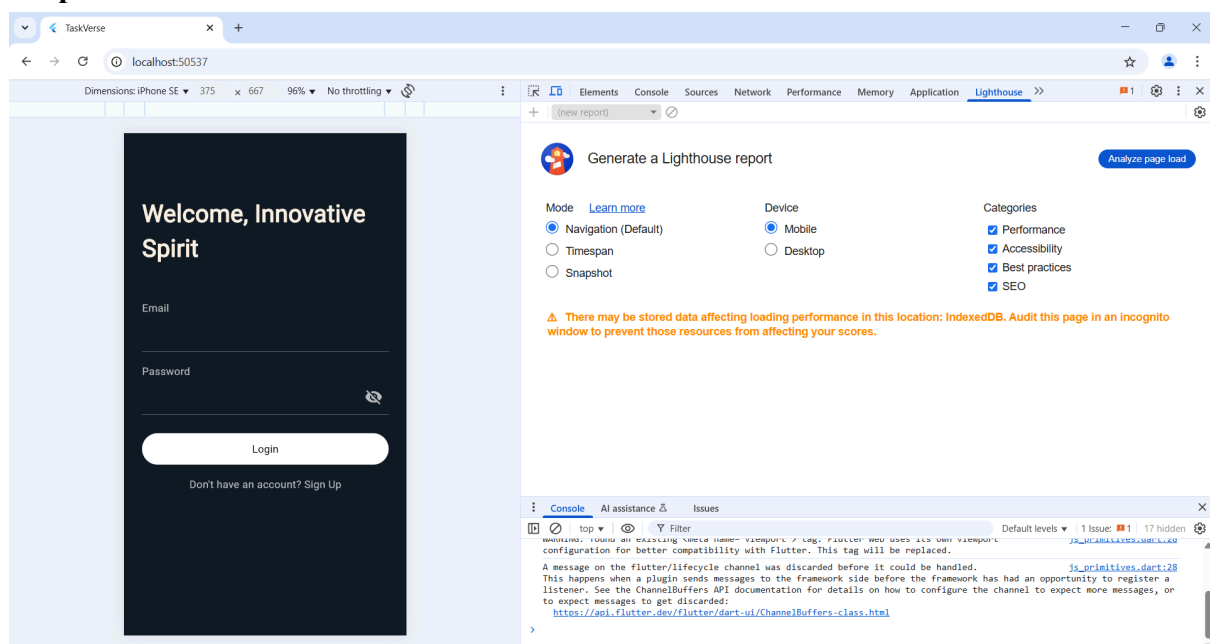
Experiment No 11			
Name		Bhushan Mukund Kor	
Roll No	28	Class	D15C
DOP		DOS	
Sign		Grade	

Aim: To use google Lighthouse PWA Analysis Tool to test the PWA functioning.

GitHub Link: https://github.com/BKCODE2003/Flutter_PWA

Description: The goal of this experiment is to utilize the Google Lighthouse PWA Analysis Tool to assess and validate the functionality of a Progressive Web App (PWA). Lighthouse, integrated into Chrome DevTools, is an automated auditing tool that evaluates critical aspects of web applications, including performance, accessibility, SEO, and PWA standards. In this experiment, a comprehensive Lighthouse report is generated to check if the app meets key PWA requirements such as fast load times, offline support, responsive design, secure HTTPS connection, service worker registration, and correct web app manifest setup. This process aids developers in enhancing their app for a smoother and more consistent user experience across different devices.

Output:



The screenshot displays the Google Lighthouse PWA Analysis Tool interface. The top section shows the page title 'TaskVerse' and the URL 'localhost:50537'. The main content area features a 'Generate a Lighthouse report' button, a 'new report' button, and a 'Learn more' link. The report shows a score of 100 for Performance, 100 for Accessibility, 100 for Best practices, and 100 for SEO. The report also includes a warning about IndexedDB data affecting loading performance.

Dimensions: iPhone SE 375 x 667 96% No throttling

TaskVerse

localhost:50537

Generate a Lighthouse report

new report

Learn more

Mode: Navigation (Default)

Device: Mobile

Categories: Performance, Accessibility, Best practices, SEO

There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.

Console

AI assistance

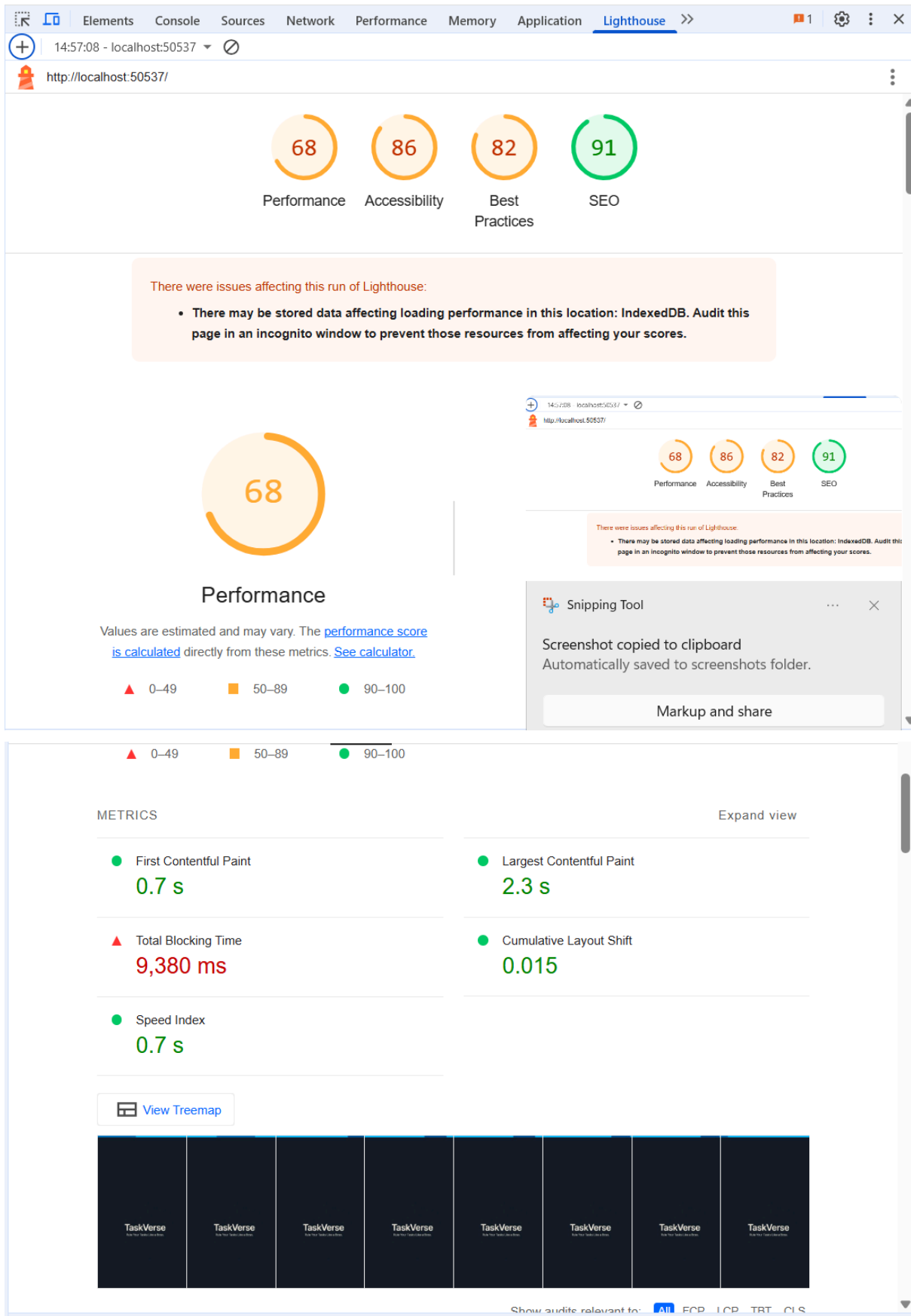
Issues

Warning: Found an existing <meta name= viewport /> tag; flutter web uses its own viewport configuration for better compatibility with Flutter. This tag will be replaced.

A message on the flutter/lifecycle channel was discarded before it could be handled.

This happens when a plugin sends messages to the framework side before the framework has had an opportunity to register a listener. See the ChannelBuffers API documentation for details on how to configure the channel to expect more messages, or to expect messages to get discarded:

https://api.flutter.dev/flutter/dart-ui/ChannelBuffers-class.html



Show audits relevant to: [All](#) [FCP](#) [LCP](#) [TBT](#) [CLS](#)

DIAGNOSTICS

▲

Minimize main-thread work — 27.6 s

▼

▲

Reduce JavaScript execution time — 14.5 s

▼

▲

Serve images in next-gen formats — Potential savings of 250 KiB

▼

▲

Properly size images — Potential savings of 61 KiB

▼

▲

Page prevented back/forward cache restoration — 1 failure reason

▼

!

Enable text compression — Error!

▼

■

Image elements do not have explicit width and height

▼

■

Minify JavaScript — Potential savings of 10,363 KiB

▼

■

Serve static assets with an efficient cache policy — 2 resources found

▼

■

Remove duplicate modules in JavaScript bundles — Potential savings of 124 KiB

▼

Conclusion: Using the Google Lighthouse tool, we thoroughly evaluated the PWA's performance, accessibility, and compliance with web standards. The audit report offered insightful feedback and practical recommendations, enabling us to enhance the overall functionality and user experience of the ToDo PWA. As a result, the app became more dependable and user-friendly across various platforms.