Yash Rahate D15B - ~~48~~ 44

MPL 01: Selection of title for (MAD)

SDS | Page No. | Date

Aim: To search unique problem statement/title for end to end mini project (MAD)

Title: Senior Citizen Assistance App.

Problemstatement :-
Accessible app for Senior citizen with medication reminders, health records, caregiver connectivity, SOS and multilingual support.

Summary:
Senior citizen Assistance app is a Flutter-based mobile application designed to assist senior citizens with their daily health and well-being. The app focuses on accessibility, user-friendly, and remote connectivity, enabling seniors and their caregivers to stay connected and maintain a healthy lifestyle.

Key Features :-
① Medication Reminders
→ Notify seniors about their daily medicine schedules with sound, vibration or visual alerts.
→ Track misses or delayed doses.

② Health Record Management:
→ Digitally store medical records, prescriptions, and reports.
→ Share health information with caregivers or doctors securely.

KHYATI

SDS | Page No.
Date / /

③ Caregiver connectivity :
→ Allow caregivers or family members to monitor the senior's health activities remotely

④ Emergency (SOS) Assistance :
→ One-tap SOS button to alert emergency contact with the senior's live location.
→ Auto dial emergency services.

⑤ Accessibility and Language Support.
→ Large fonts, high contrast, simple navigation for ease of use, voice commands for tech-challenged users.
→ Multiple regional language support.

Conclusion:
This app bridges the gap between technology and senior-friendly accessibility, making life safer and more convenient for older adults and their families.

PRACTICAL 2

Aim: To design flutter ui by including common widgets.

Theory:

Introduction to Flutter
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It uses the Dart programming language and provides a fast development experience with features like Hot Reload and customizable widgets.

Flutter follows a widget-based architecture, where everything in the UI is represented as a widget. It includes two main types of widgets:

1. StatelessWidget – These widgets do not change once they are built.
2. StatefulWidget – These widgets maintain state and can change dynamically.

Flutter provides a rich set of pre-built widgets, such as Scaffold, AppBar, ListTile, ElevatedButton, BottomNavigationBar, and many more, allowing developers to create highly interactive applications.

Objective
The objective of this experiment is to design and implement a Home Screen for the Senior Assistance App using Flutter. This screen serves as the main interface where users can view their tasks, reminders, and navigate through the application.

Implementation
The code consists of a main.dart file that implements the HomeScreen. Below are the key components of the implementation:

1. MaterialApp and Theme Customization
   - We use MaterialApp as the root widget and set debugShowCheckedModeBanner: false to remove the debug banner.
   - The application follows a dark theme, where the background color is set to black, and the primary color is red.

2. AppBar (Top Navigation Bar)
   - The AppBar is customized with a black background and white text.
   - An action button (mail icon) is added with a red notification indicator to represent unread messages.

3. Displaying Medication Reminders
   - The body contains a Column with a ListTile wrapped inside a Card, representing a medication reminder.

- The reminder includes:
  - Time (20:00)
  - Person's Name (Yash)
  - Medication Information (1 pill)
  - A status icon (gray circle)

4. Floating Action Button
   - At the bottom, an ElevatedButton.icon is added to allow users to create a one-time medication entry.
   - The button is styled with a brown background, white text, and rounded corners.

5. Bottom Navigation Bar
   - A BottomNavigationBar is implemented for easy navigation between different sections of the app.
   - The navigation bar contains four items:
     - Today (Current tasks)
     - Progress (Health progress)
     - Support (Help & Assistance)
     - Treatment (Medical-related features)
   - The selected item is highlighted in red, while unselected items remain gray.

CODE

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomeScreen(),
      theme: ThemeData.dark().copyWith(
        scaffoldBackgroundColor: Colors.black,
        primaryColor: Colors.red,
      ),
    );
  }
}
```

```
class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.black,
        elevation: 0,
        title: Text(
          'Today',
          style: TextStyle(color: Colors.white, fontSize: 20),
        ),
        actions: [
          IconButton(
            icon: Stack(
              children: [
                Icon(Icons.mail, color: Colors.white),
                Positioned(
                  top: 0,
                  right: 0,
                  child: CircleAvatar(
                    radius: 5,
                    backgroundColor: Colors.red,
                  ),
                ),
              ],
            ),
            onPressed: () {},
          )
        ],
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            SizedBox(height: 20),
            Text(
              '20:00',
              style: TextStyle(
                color: Colors.white,
                fontSize: 18,
              ),
            ),
            SizedBox(height: 10),
```

```
Card(
  color: Colors.grey[850],
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(10),
  ),
  child: ListTile(
    leading: Icon(
      Icons.link,
      color: Colors.grey,
    ),
    title: Text(
      'Yash',
      style: TextStyle(
        color: Colors.white,
        fontSize: 16,
      ),
    ),
    subtitle: Text(
      '1 pill(s)',
      style: TextStyle(
        color: Colors.grey,
        fontSize: 14,
      ),
    ),
    trailing: CircleAvatar(
      radius: 12,
      backgroundColor: Colors.transparent,
      child: Icon(
        Icons.circle,
        color: Colors.grey,
      ),
    ),
  ),
),
Spacer(),
Align(
  alignment: Alignment.bottomCenter,
  child: ElevatedButton.icon(
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.brown,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(20),
      ),
      padding: EdgeInsets.symmetric(horizontal: 20, vertical: 10),
```

```
                ),
                onPressed: () {},
                icon: Icon(Icons.add, color: Colors.white),
                label: Text(
                  'One-time entry',
                  style: TextStyle(fontSize: 16, color: Colors.white),
                ),
              ),
            )
          ],
        ),
      ),
      bottomNavigationBar: BottomNavigationBar(
        backgroundColor: Colors.black,
        selectedItemColor: Colors.red,
        unselectedItemColor: Colors.grey,
        items: [
          BottomNavigationBarItem(
            icon: Icon(Icons.list),
            label: 'Today',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.bar_chart),
            label: 'Progress',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.support),
            label: 'Support',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.medical_services),
            label: 'Treatment',
          ),
        ],
      ),
    );
  }
}
```

Screenshot:

Conclusion

In this experiment, we implemented the Home Screen for the Senior Assistance App with a dark theme, a medication reminder card, a floating action button, and a bottom navigation bar for easy navigation. During implementation, we initially faced issues with widget alignment and styling inconsistencies, which we resolved by adjusting padding, using SizedBox for spacing, and refining the Card and ListTile properties for a better UI experience.

PRACTICAL 3

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

Theory:

Introduction
In Flutter, icons, images, and fonts enhance the app's user interface (UI) and improve usability. Custom icons and images make the app visually appealing, while fonts improve readability and branding.

Concepts Used
1. Icons: Flutter provides built-in Icons and allows the use of custom icon sets.
2. Images: Images can be loaded from assets, network, or memory.
3. Fonts: Custom fonts are added through pubspec.yaml to enhance typography.

Implementation in Our App
In our Flutter app, we have:
- Used built-in icons (Icons.mail, Icons.list, etc.) for navigation and actions.
- Added a profile image (assets/user_profile.png) to display user details.
- Integrated a custom font (Roboto) to enhance text appearance.
- Converted HomeScreen into a StatefulWidget to handle UI updates dynamically.

CODE (main.dart)

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomeScreen(),
      theme: ThemeData.dark().copyWith(
        scaffoldBackgroundColor: Colors.black,
        primaryColor: Colors.red,
        textTheme: TextTheme(
          bodyText1: TextStyle(fontFamily: 'Roboto', color: Colors.white),
          bodyText2: TextStyle(fontFamily: 'Roboto', color: Colors.grey),
```

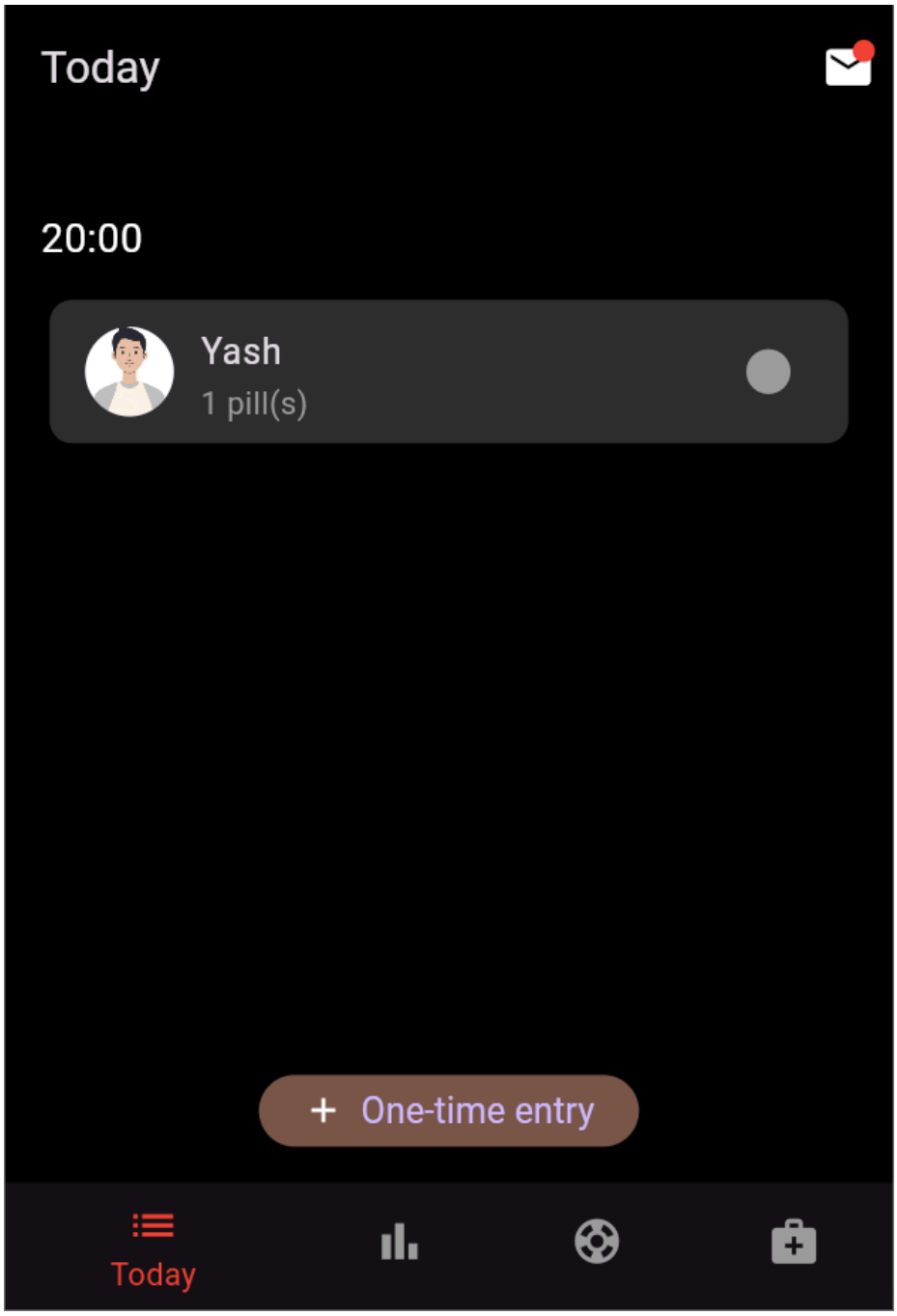
```
        ),
      ),
    );
  }
}

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  int _selectedIndex = 0;

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.black,
        elevation: 0,
        title: Text(
          'Today',
          style: TextStyle(fontSize: 20, fontFamily: 'Roboto'),
        ),
        actions: [
          IconButton(
            icon: Stack(
              children: [
                Icon(Icons.mail, color: Colors.white),
                Positioned(
                  top: 0,
                  right: 0,
                  child: CircleAvatar(
                    radius: 5,
                    backgroundColor: Colors.red,
                  ),
                ),
              ],
```

```
        ),
        onPressed: () {},
      )
    ],
  ),
  body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        SizedBox(height: 20),
        Text(
          '20:00',
          style: TextStyle(
            color: Colors.white,
            fontSize: 18,
            fontFamily: 'Roboto',
          ),
        ),
        SizedBox(height: 10),
        Card(
          color: Colors.grey[850],
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ),
          child: ListTile(
            leading: CircleAvatar(
              backgroundImage: AssetImage('assets/user_profile.png'),
            ),
            title: Text(
              'Yash',
              style: TextStyle(fontSize: 16, fontFamily: 'Roboto'),
            ),
            subtitle: Text(
              '1 pill(s)',
              style: TextStyle(fontSize: 14, color: Colors.grey),
            ),
            trailing: Icon(Icons.circle, color: Colors.grey),
          ),
        ),
        Spacer(),
        Align(
          alignment: Alignment.bottomCenter,
          child: ElevatedButton.icon(
```

```
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.brown,
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(20),
              ),
              padding: EdgeInsets.symmetric(horizontal: 20, vertical: 10),
            ),
            onPressed: () {},
            icon: Icon(Icons.add, color: Colors.white),
            label: Text(
              'One-time entry',
              style: TextStyle(fontSize: 16, fontFamily: 'Roboto'),
            ),
          ),
        )
      ],
    ),
  ),
  bottomNavigationBar: BottomNavigationBar(
    backgroundColor: Colors.black,
    selectedItemColor: Colors.red,
    unselectedItemColor: Colors.grey,
    currentIndex: _selectedIndex,
    onTap: _onItemTapped,
    items: [
      BottomNavigationBarItem(
        icon: Icon(Icons.list),
        label: 'Today',
      ),
      BottomNavigationBarItem(
        icon: Icon(Icons.bar_chart),
        label: 'Progress',
      ),
      BottomNavigationBarItem(
        icon: Icon(Icons.support),
        label: 'Support',
      ),
      BottomNavigationBarItem(
        icon: Icon(Icons.medical_services),
        label: 'Treatment',
      ),
    ],
  ),
);
```

```
  }
}
```

Screenshot:



Conclusion

We successfully added icons, images, and fonts in our app. Initially, we faced issues loading custom fonts and assets due to incorrect pubspec.yaml configuration, which we resolved by properly defining asset paths and font families.

PRACTICAL 4

Aim: To create an interactive Form using form widget .

Theory:
The `Form` widget in Flutter is a fundamental building block for creating forms and handling user input. In the code provided, the `Form` widget is used to manage the state and validation of the input fields. Here's a detailed breakdown of how it works in this context:

Why Use the Form Widget**?**

- **State Management**: It simplifies managing multiple input fields by grouping them under one state.
- **Validation**: Provides a structured way to validate user inputs.
- **Scalability**: You can add more fields easily without affecting the overall structure.
- **Code Organization**: Keeps validation logic separate from the rest of the UI code.

CODE
Folder Structure

```
lib/
├── screens/
│   ├── home_screen.dart
│   ├── login_screen.dart
│   ├── features/
│   │   ├── medication_reminders_screen.dart
│   │   ├── add_medication_reminder_screen.dart
```

medication_reminders_screen.dart:

```
import 'package:flutter/material.dart';
import 'add_medication_reminder_screen.dart';

class MedicationRemindersScreen extends StatefulWidget {
  const MedicationRemindersScreen({super.key});

  @override
  State<MedicationRemindersScreen> createState() => _MedicationRemindersScreenState();
}

class _MedicationRemindersScreenState extends State<MedicationRemindersScreen> {
  @override
```

```
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(
            Icons.medication,
            size: 80,
            color: Colors.blue.shade700,
          ),
          const SizedBox(height: 16),
          const Text(
            'Medication Reminders',
            style: TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 24),
          ElevatedButton.icon(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue.shade700,
              foregroundColor: Colors.white,
              padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 12),
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(30),
              ),
            ),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const AddMedicationReminderScreen()),
              );
            },
            icon: const Icon(Icons.add),
            label: const Text('Add Medication Reminder', style: TextStyle(fontSize: 16)),
          ),
        ],
      ),
    );
  }
}
```

add_medication_reminder_screen.dart

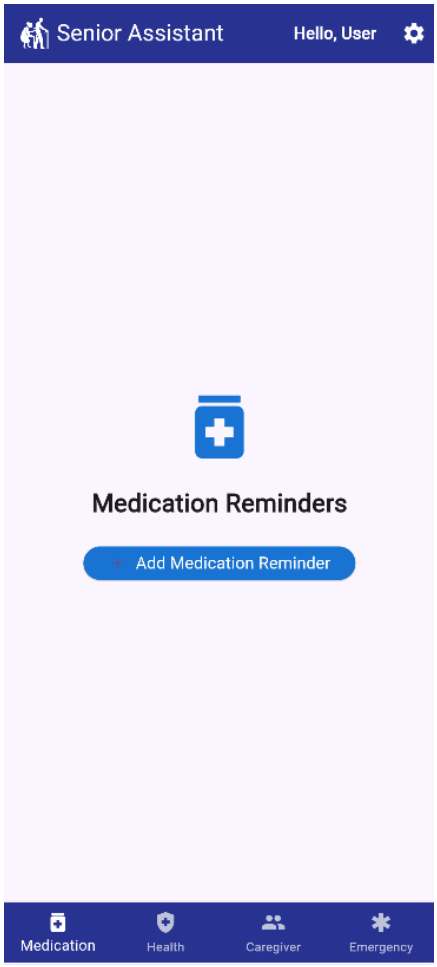
```
import 'package:flutter/material.dart';

class AddMedicationReminderScreen extends StatefulWidget {
  const AddMedicationReminderScreen({super.key});

  @override
  State<AddMedicationReminderScreen> createState() =>
_AddMedicationReminderScreenState();
}

class _AddMedicationReminderScreenState extends State<AddMedicationReminderScreen> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _medicineNameController = TextEditingController();
  final TextEditingController _dosageController = TextEditingController();
  final TextEditingController _timeController = TextEditingController();
  TimeOfDay? _selectedTime;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Add Medication Reminder'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: ListView(
            children: [
              // Medicine Name Field
              TextFormField(
                controller: _medicineNameController,
                decoration: const InputDecoration(
                  labelText: 'Medicine Name',
                  border: OutlineInputBorder(),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter the medicine name';
                  }
                  return null;
                },
              ),
```

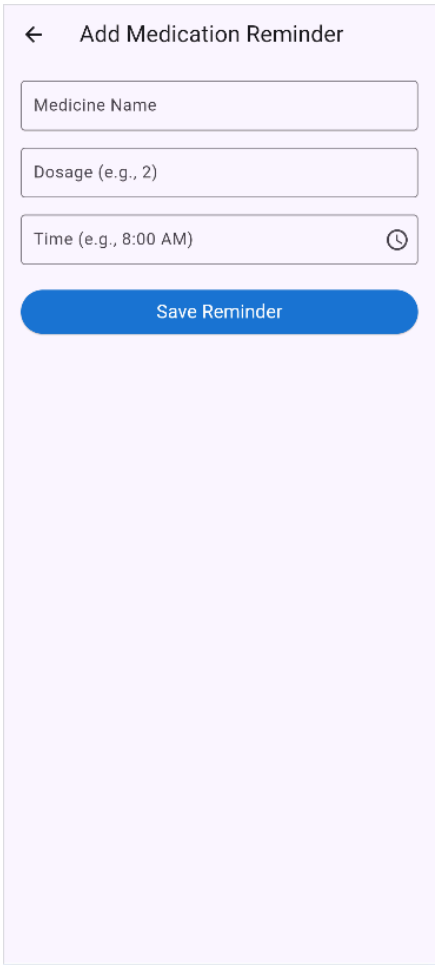
```
const SizedBox(height: 16),

// Dosage Field (Number Input)
TextFormField(
  controller: _dosageController,
  keyboardType: TextInputType.number,
  decoration: const InputDecoration(
    labelText: 'Dosage (e.g., 2)',
    border: OutlineInputBorder(),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter the dosage';
    }
    if (int.tryParse(value) == null) {
      return 'Please enter a valid number';
    }
    return null;
  },
),
const SizedBox(height: 16),

// Time Field (Time Picker)
TextFormField(
  controller: _timeController,
  readOnly: true,
  decoration: const InputDecoration(
    labelText: 'Time (e.g., 8:00 AM)',
    border: OutlineInputBorder(),
    suffixIcon: Icon(Icons.access_time),
  ),
  onTap: () async {
    // Show Time Picker
    TimeOfDay? pickedTime = await showTimePicker(
      context: context,
      initialTime: TimeOfDay.now(),
    );

    if (pickedTime != null) {
      setState(() {
        _selectedTime = pickedTime;
        _timeController.text = pickedTime.format(context);
      });
    }
```

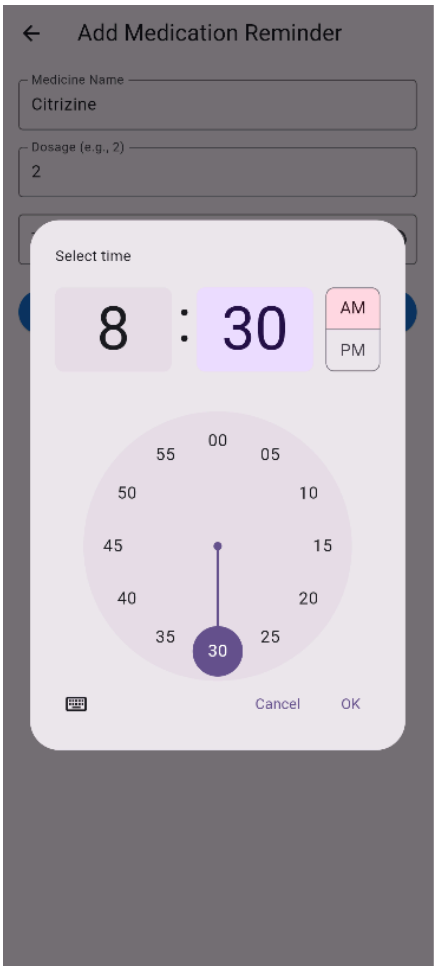
```
            },
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please select a time';
              }
              return null;
            },
          ),
          const SizedBox(height: 24),

          // Save Reminder Button
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue.shade700,
              foregroundColor: Colors.white,
              padding: const EdgeInsets.symmetric(vertical: 16),
            ),
            onPressed: () {
              if (_formKey.currentState!.validate()) {
                // Process the input data
                String medicineName = _medicineNameController.text;
                int dosage = int.parse(_dosageController.text);
                String time = _timeController.text;

                // Show confirmation message
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(
                    content: Text(
                      'Medication Reminder Added:\nMedicine: $medicineName\nDosage:
$dosage\nTime: $time',
                    ),
                  ),
                );

                // Clear form after submission
                _medicineNameController.clear();
                _dosageController.clear();
                _timeController.clear();
              }
            },
            child: const Text('Save Reminder', style: TextStyle(fontSize: 18)),
          ),
        ],
      ),
```

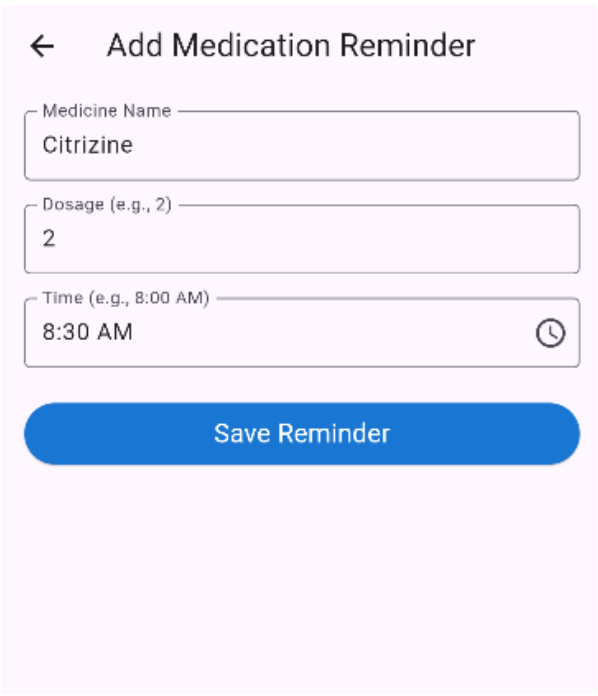
```
        ),
      ),
    );
  }
}
```

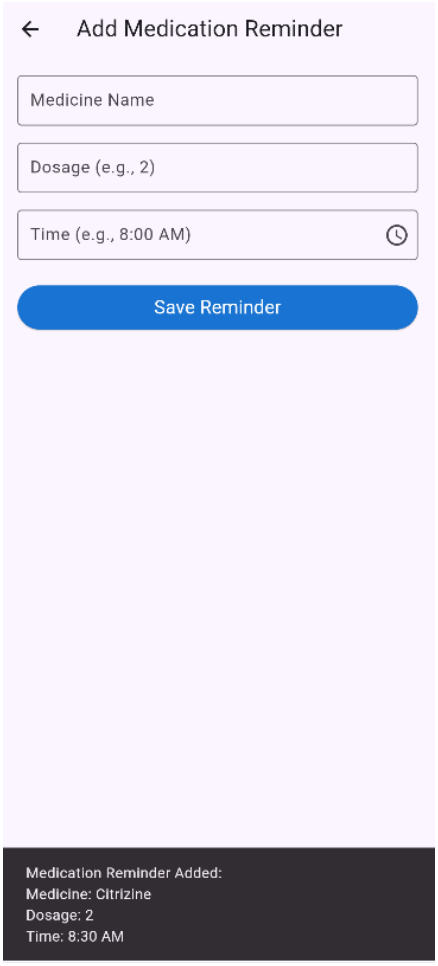
Screenshot:


Senior Assistant
Hello, User
Medication Reminders
Add Medication Reminder
Medication
Health
Caregiver
Emergency


Add Medication Reminder
Medicine Name
Dosage (e.g., 2)
Time (e.g., 8:00 AM)
Save Reminder


Add Medication Reminder
Medicine Name
Citrizine
Dosage (e.g., 2)
2
Select time
8
:
30
AM
PM
00
05
10
15
20
25
30
35
40
45
50
55
Cancel
OK


Add Medication Reminder
Medicine Name
Citrizine
Dosage (e.g., 2)
2
Time (e.g., 8:00 AM)
8:30 AM
Save Reminder



Conclusion

During the implementation, I encountered errors like incorrect type conversions (e.g., parsing `String` to `int` for dosage) and validation failures when fields were left empty. These issues were resolved by adding robust validation logic for each field and displaying clear error messages. Proper type handling and clearing the controllers after submission ensured smooth functionality and improved user experience.

PRACTICAL 5

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

1. Navigation in Flutter

Navigation in Flutter is the process of **switching between screens (routes)** within an app. There are two main ways to implement navigation:

1. **Using `Navigator.push()` and `Navigator.pop()`** (for simple navigation)
2. **Using Named Routes (`Navigator.pushNamed()`)** (for better route management)

Eg:
Navigator.pushNamed(context, '/add_medications');

2. Routing in Flutter

Routing defines how screens are structured and navigated. In our code, we used **named routes** to manage screen transitions.

Defining Routes in `main.dart`:

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Senior Care Assistant',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      initialRoute: '/login',
      routes: {
        '/': (context) => const HomeScreen(),
        '/login': (context) => const LoginScreen(),
        '/signup': (context) => const SignupScreen(),
        '/reset_password': (context) => const ResetPassword(),
        '/settings': (context) => const SettingsScreen(),
        '/search_users': (context) => const UserSearchScreen(),
        '/add_medications': (context) => const AddMedicationReminderScreen() ,
      },
    ); }}
```

3. Gesture Detection in Flutter

Flutter provides the `GestureDetector` widget to capture user gestures like **tap, swipe, and long press**.

Gestures Implemented in the Code:

```
GestureDetector(
  onTap: () {
    Navigator.pushNamed(context, '/add_medications');
  },
  onLongPress: () {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Long Press Detected!")),
    );
  },
  onHorizontalDragEnd: (details) {
    if (details.primaryVelocity! < 0) {
      Navigator.pushNamed(context, '/add_medications');
    }
  },
  child: Icon(Icons.medication, size: 80, color: Colors.blue.shade700),
);
```

CODE

Folder Structure

```
lib/
├── reuseable_widgets/
│   └── reuseable_widgets.dart
├── screens/
│   ├── features/
│   │   ├── add_medication_reminder_screen.dart
│   │   ├── caregiver_connection_screen.dart
│   │   ├── emergency_assistance_screen.dart
│   │   ├── health_records_screen.dart
│   │   └── medication_reminders_screen.dart
│   ├── settings/
│   │   ├── font_size_settings.dart
│   │   ├── language_settings.dart
│   │   └── settings_screen.dart
│   ├── home_screen.dart
│   ├── login_screen.dart
│   ├── reset_password.dart
│   └── signup_screen.dart
├── util/
│   └── color_util.dart
├── widgets/
│   └── custom_text_field.dart
├── firebase_options.dart
└── main.dart
```

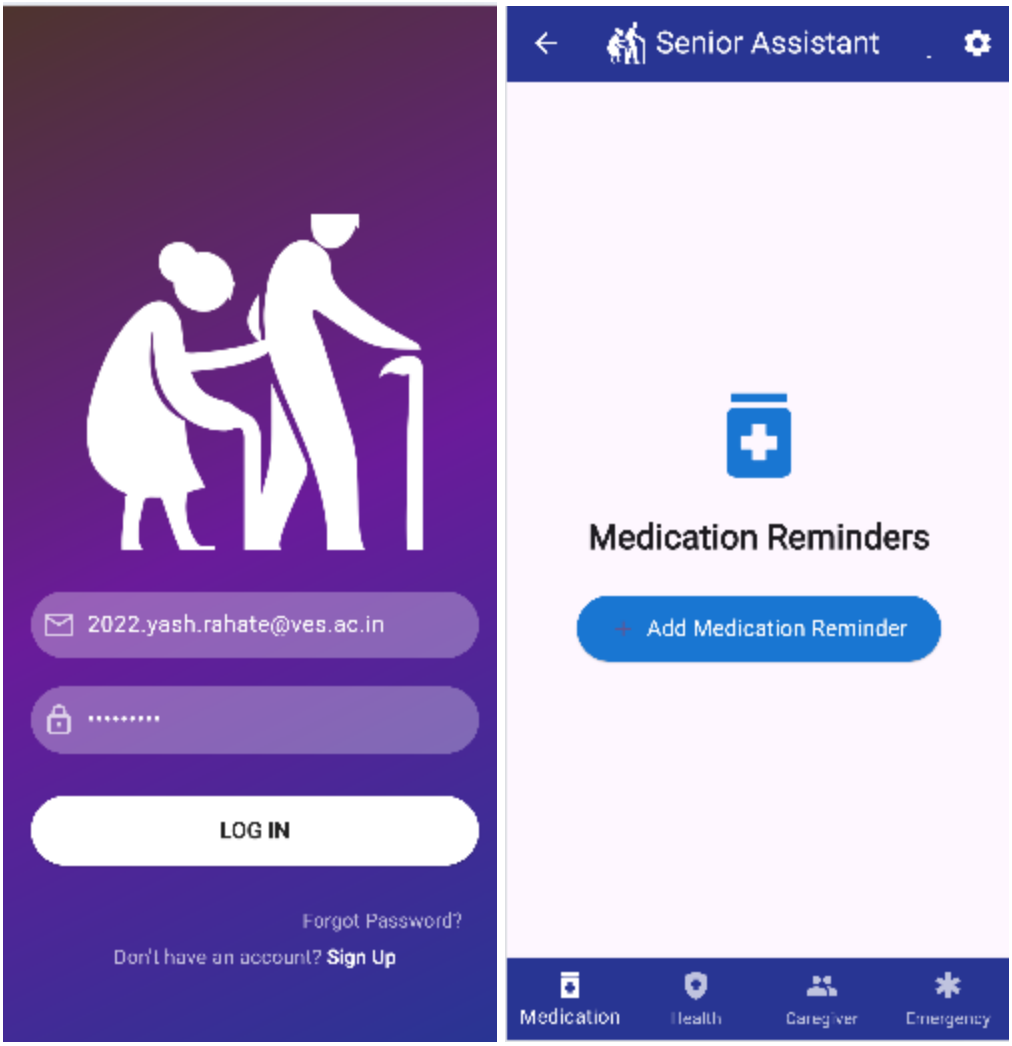
Navigation:

lib/screens/login_screen.dart:

```
signInSignUpButton(context, true, () {
                  if (_formKey.currentState!.validate()) {
                    _loginUser(context);
                  }
                }),

signUpOption(),
```

```
Future<void> _loginUser(BuildContext context) async {
   try {
     // Show a loading indicator
     showDialog(
       context: context,
       barrierDismissible: false,
       builder: (context) => Center(child: CircularProgressIndicator()),
     );

     // Attempt to log in
     UserCredential userCredential = await FirebaseAuth.instance
         .signInWithEmailAndPassword(
       email: _emailTextController.text.trim(),
       password: _passwordTextController.text.trim(),
     );

     print("Logged in successfully: ${userCredential.user!.email}");

     // Navigate to HomeScreen after successful login
     Navigator.pop(context); // Close the loading indicator
     Navigator.pushReplacement(
       context,
       MaterialPageRoute(builder: (context) => HomeScreen()),
     );
   } catch (error) {
     Navigator.pop(context); // Close the loading indicator
     _showErrorDialog(context, error.toString());
   }
 }
```

```
Row signUpOption() {
   return Row(
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
       const Text("Don't have an account?",
           style: TextStyle(color: Colors.white70)),
       GestureDetector(
         onTap: () {
           Navigator.push(context,
               MaterialPageRoute(builder: (context) => SignupScreen()));
         },
         child: const Text(
           " Sign Up",
           style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold),
         ),
       )
     ],
   );
 }
```



Routes

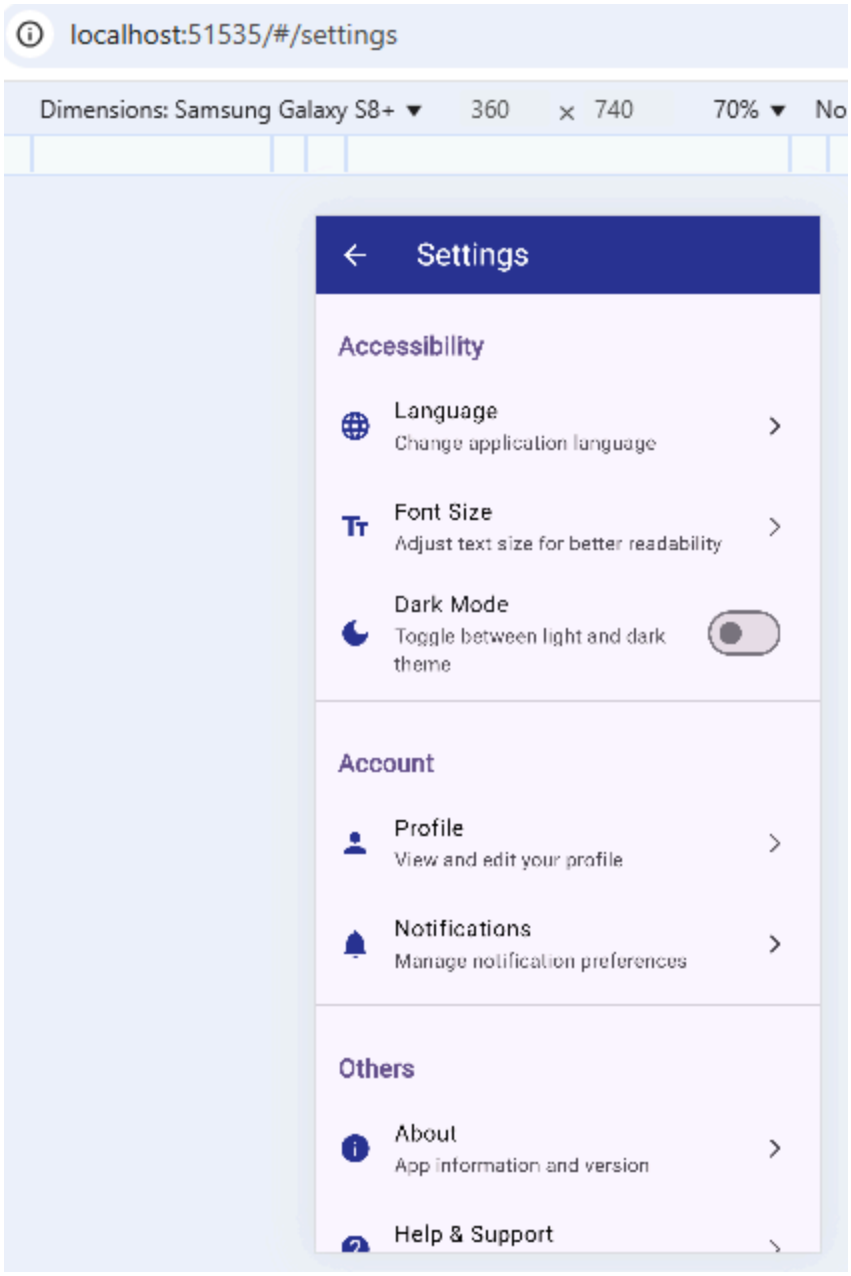
lib/main.dart:

```
import 'package:sca/screens/features/add_medication_reminder_screen.dart';
import 'package:sca/screens/login_screen.dart';
import 'package:sca/screens/home_screen.dart';
import 'package:sca/screens/signup_screen.dart';
import 'package:sca/screens/reset_password.dart';
import 'package:sca/screens/features/user_search_screen.dart';
import 'package:sca/screens/settings/settings_screen.dart';
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
```

```
      title: 'Senior Care Assistant',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      initialRoute: '/login',
      routes: {
        '/': (context) => const HomeScreen(),
        '/login': (context) => const LoginScreen(),
        '/signup': (context) => const SignupScreen(),
        '/reset_password': (context) => const ResetPassword(),
        '/settings': (context) => const SettingsScreen(),
        '/search_users': (context) => const UserSearchScreen(),
        '/add_medications': (context) => const AddMedicationReminderScreen() ,
      },
    );
  }
}
```

lib/screens/features/medication_reminders_screen.dart

```
class _MedicationRemindersScreenState extends State<MedicationRemindersScreen> {
  void _navigateToAddMedications() {
    Navigator.pushNamed(context, '/add_medications');
  }
```



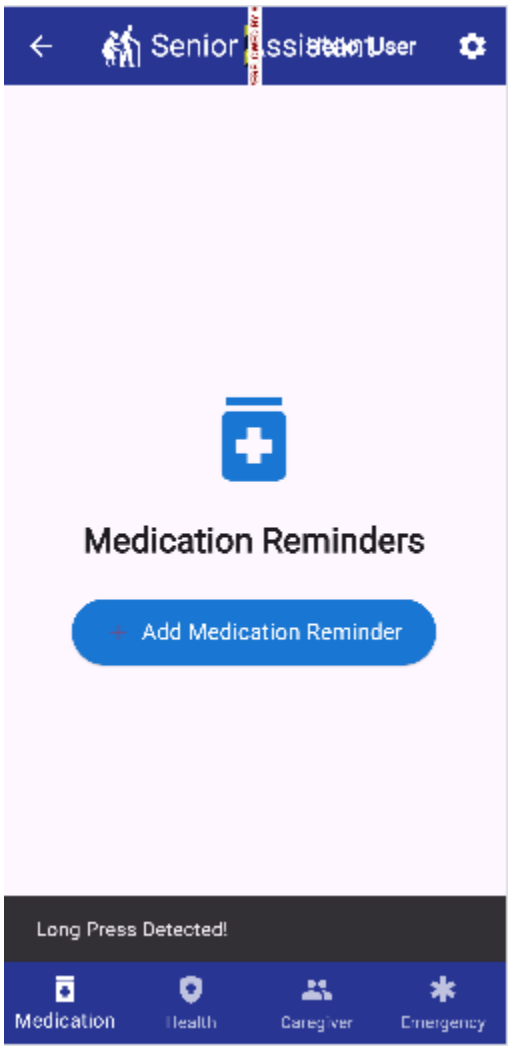
Gesture

lib/screens/features/medication_reminders_screen.dart:

```
import 'package:flutter/material.dart';

class MedicationRemindersScreen extends StatefulWidget {
  const MedicationRemindersScreen({super.key});

  @override
  State<MedicationRemindersScreen> createState() => _MedicationRemindersScreenState();
}
```

```
class _MedicationRemindersScreenState extends State<MedicationRemindersScreen> {
  void _navigateToAddMedications() {
    Navigator.pushNamed(context, '/add_medications');
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: GestureDetector(
        onTap: _navigateToAddMedications, // Tap to navigate
        onLongPress: () {
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text("Long Press Detected!")),
          );
        },
        onHorizontalDragEnd: (details) {
          if (details.primaryVelocity! < 0) {
            _navigateToAddMedications(); // Swipe Left to Navigate
          }
        },
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Icon(
              Icons.medication,
              size: 80,
              color: Colors.blue.shade700,
            ),
            const SizedBox(height: 16),
            const Text(
              'Medication Reminders',
              style: TextStyle(
                fontSize: 24,
                fontWeight: FontWeight.bold,
              ),
            ),
            const SizedBox(height: 24),
            ElevatedButton.icon(
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.blue.shade700,
                foregroundColor: Colors.white,
                padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 12),
                shape: RoundedRectangleBorder(
```

```
            borderRadius: BorderRadius.circular(30),
          ),
        ),
        onPressed: _navigateToAddMedications,
        icon: const Icon(Icons.add),
        label: const Text('Add Medication Reminder', style: TextStyle(fontSize: 16)),
      ),
    ],
  ),
 ),
); }}
```



Conclusion

In this experiment, we successfully applied **navigation, routing, and gestures** in a Flutter app. The following objectives were achieved:

Implemented Named Routing for better screen management.
Added Gesture-Based Navigation (tap, swipe, and long press).
Ensured a Smooth User Experience by making icons and buttons interactive.

PRACTICAL 6

Aim: To Connect Flutter UI with FireBase database.

Theory:

In modern mobile applications, user authentication and data management play a crucial role. Firebase, a backend-as-a-service platform by Google, simplifies authentication and database management for Flutter applications. Firebase Authentication allows secure sign-in using different methods such as email/password, phone authentication, and third-party providers (Google, Facebook, etc.). In this experiment, we integrate a Flutter UI with Firebase to enable user signup, login, password reset, and data storage in Firestore.

Implementation in Our Code

1. User Signup
    - Users enter email, password, and username.
    - Email format and password length are validated.
    - Firebase creates an account and stores user details in Firestore.
    - A verification email is sent to confirm authenticity.

2. Email Verification
    - Users must verify their email before logging in.
    - The app restricts access until verification is complete.

3. User Login
    - Users log in with email and password.
    - Firebase checks credentials and verification status before granting access.

4. Password Reset
    - Users can request a password reset email if they forget their password.

5. Form Validation
    - Ensures correct email format, password length (min 8 chars), and non-empty fields.

CODE

Folder Structure

```
lib/
├── reuseable_widgets/
│   └── reuseable_widgets.dart
├── screens/
│   ├── features/
│   │   ├── add_medication_reminder_screen.dart
│   │   ├── caregiver_connection_screen.dart
│   │   ├── emergency_assistance_screen.dart
│   │   ├── health_records_screen.dart
│   │   └── medication_reminders_screen.dart
│   ├── settings/
│   │   ├── font_size_settings.dart
│   │   ├── language_settings.dart
│   │   └── settings_screen.dart
│   ├── home_screen.dart
│   ├── login_screen.dart
│   ├── reset_password.dart
│   └── signup_screen.dart
├── util/
│   └── color_util.dart
├── widgets/
│   └── custom_text_field.dart
├── firebase_options.dart
└── main.dart
```

signup_screen.dart:

```
void _signupUser() {
  if (_formKey.currentState?.validate() ?? false) {
    setState(() {
      _isLoading = true;
    });

    FirebaseAuth.instance
        .createUserWithEmailAndPassword(
      email: _emailTextController.text.trim(),
      password: _passwordTextController.text.trim(),
    )
        .then((value) async {
      User? user = value.user;

      if (user != null) {
        // Save user details to Firestore
        await FirebaseFirestore.instance.collection('users').doc(user.uid).set({
          'username': _userNameTextController.text.trim(),
          'email': _emailTextController.text.trim(),
        });
```

```
        await user.sendEmailVerification(); // Ensure email verification
      }

      setState(() {
        _isLoading = false;
      });

      print("Created new account");

      // Navigate to HomeScreen
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => HomeScreen()),
      );
    }).catchError((error) {
      setState(() {
        _isLoading = false;
      });
      print("Error: ${error.toString()}");
      _showErrorDialog("Failed to create an account. Please try again.");
    });
  }
}
```

login_screen.dart:

```
 Future<void> _loginUser(BuildContext context) async {
    try {
      showDialog(
        context: context,
        barrierDismissible: false,
        builder: (context) => Center(child: CircularProgressIndicator()),
      );

      // Attempt to log in
      UserCredential userCredential =
          await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: _emailTextController.text.trim(),
        password: _passwordTextController.text.trim(),
      );

      User? user = userCredential.user;
```

```
      // Check if the email is verified
      if (user != null && !user.emailVerified) {
        await user.sendEmailVerification(); // Send verification email
        Navigator.pop(context);
        _showErrorDialog(
          context,
          "Please verify your email. A verification link has been sent to your email.",
        );
        return;
      }

      if (user != null) {
        print("Logged in successfully: ${user.email}");
      }

      // Navigate to HomeScreen after successful login
      Navigator.pop(context);
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => HomeScreen()),
      );
    } catch (error) {
      Navigator.pop(context);
      _showErrorDialog(context, error.toString());
    }
  }
```

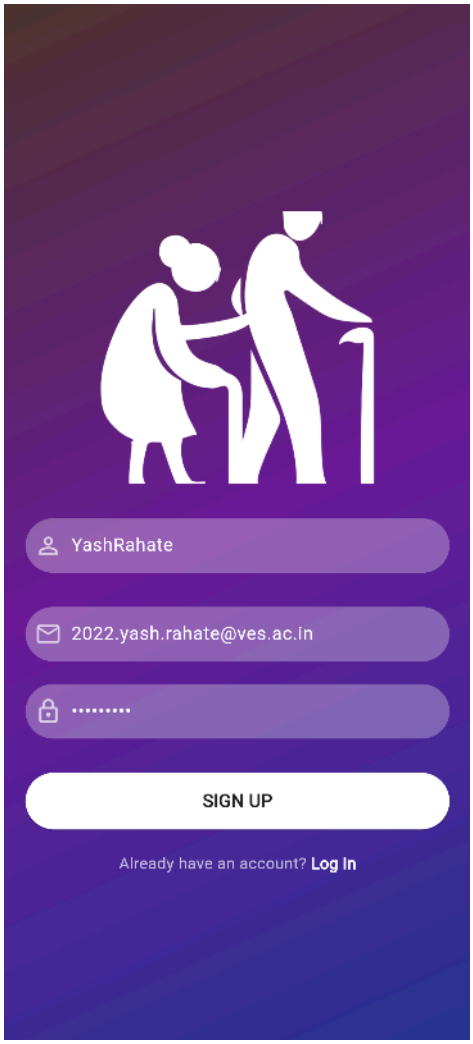
reset_password.dart:

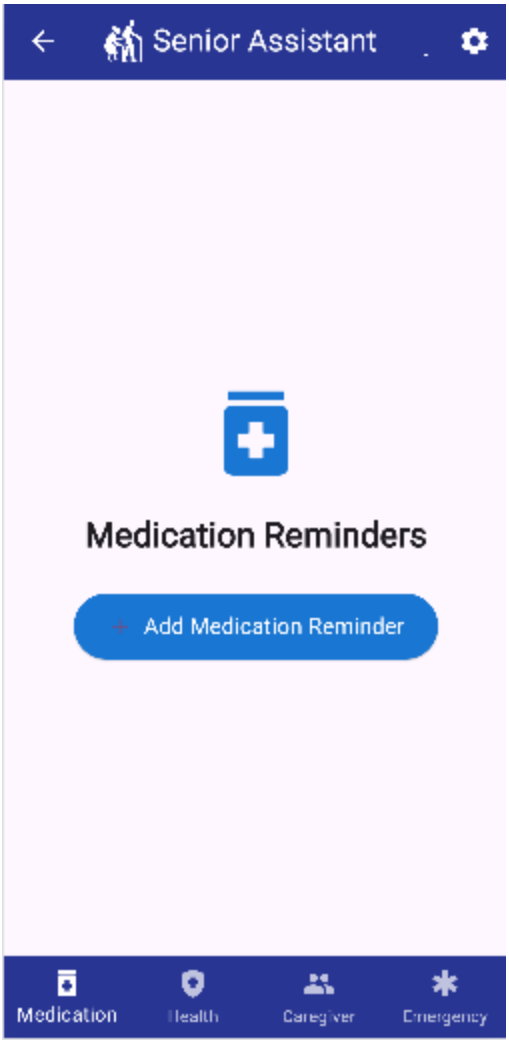
```
 void _resetPassword() {
    final email = _emailTextController.text.trim();

    setState(() {
      _isLoading = true;
    });

    FirebaseAuth.instance
        .sendPasswordResetEmail(email: email)
        .then((value) {
      setState(() {
        _isLoading = false;
      });
      _showSnackbar("Password reset email sent successfully!");
      Navigator.of(context).pop();
```

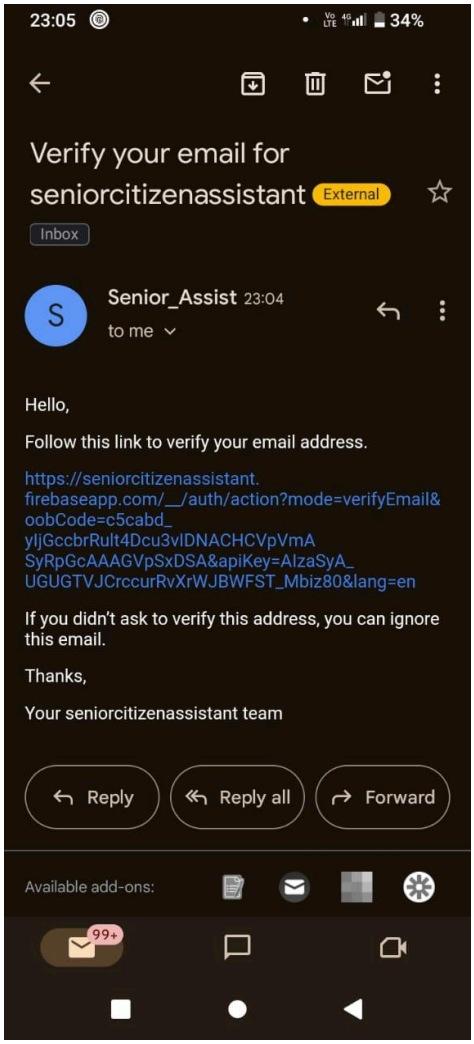
```
    }).catchError((error) {
      setState(() {
        _isLoading = false;
      });
      _showSnackbar("Error: ${error.message}");
    });
  }
```

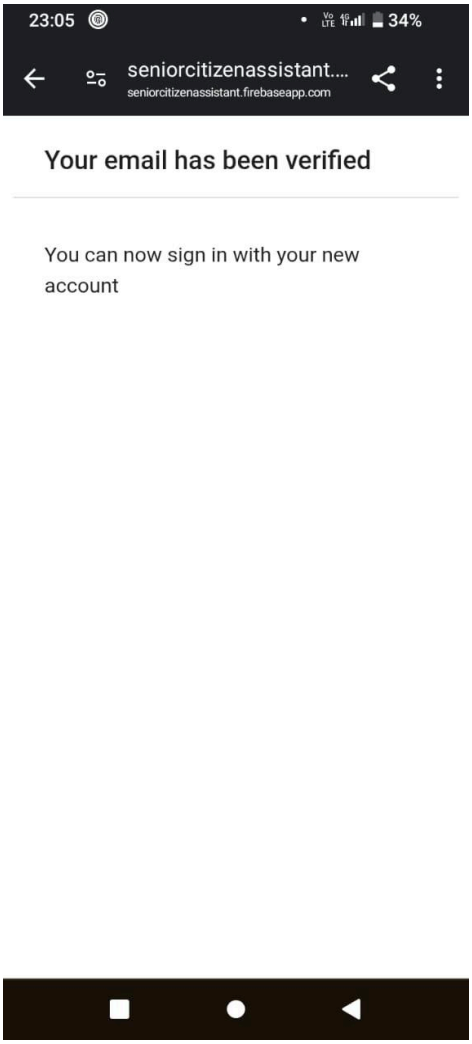
VALIDATION code:

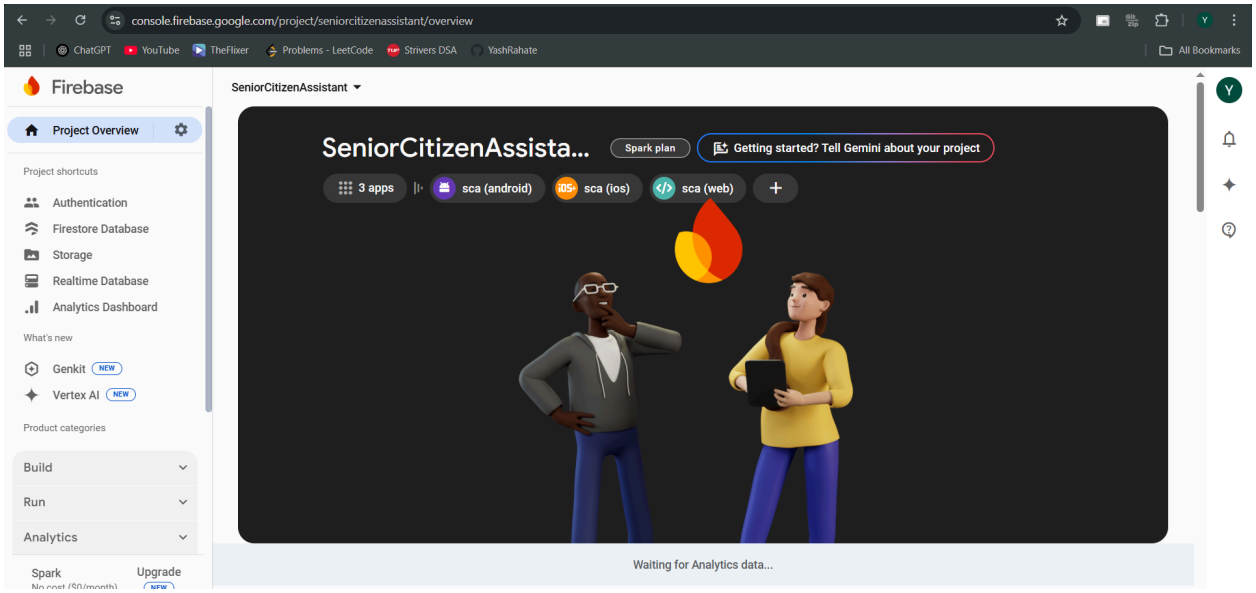
```
(value) {
                    if (value == null || value.isEmpty) {
                      return "Please enter your email";
                    }
                    if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
                      return "Enter a valid email address";
                    }
                    return null;
                  },

(value) {
                    if (value == null || value.isEmpty) {
                      return 'Password is required';
                    }
                    if (value.length < 8) {
                      return 'Password must be at least 8 characters long';
                    }
                    return null;
                  },
```

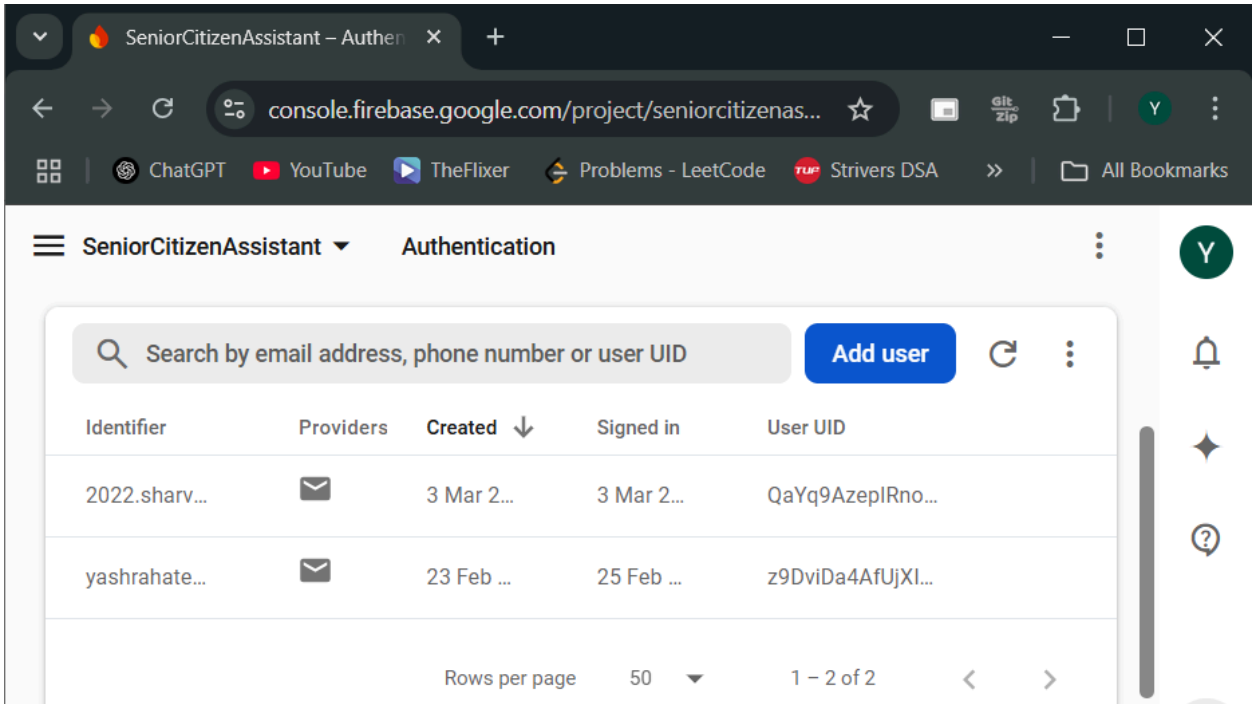

YashRahate
2022.yash.rahate@ves.ac.in
SIGN UP
Already have an account? Log In

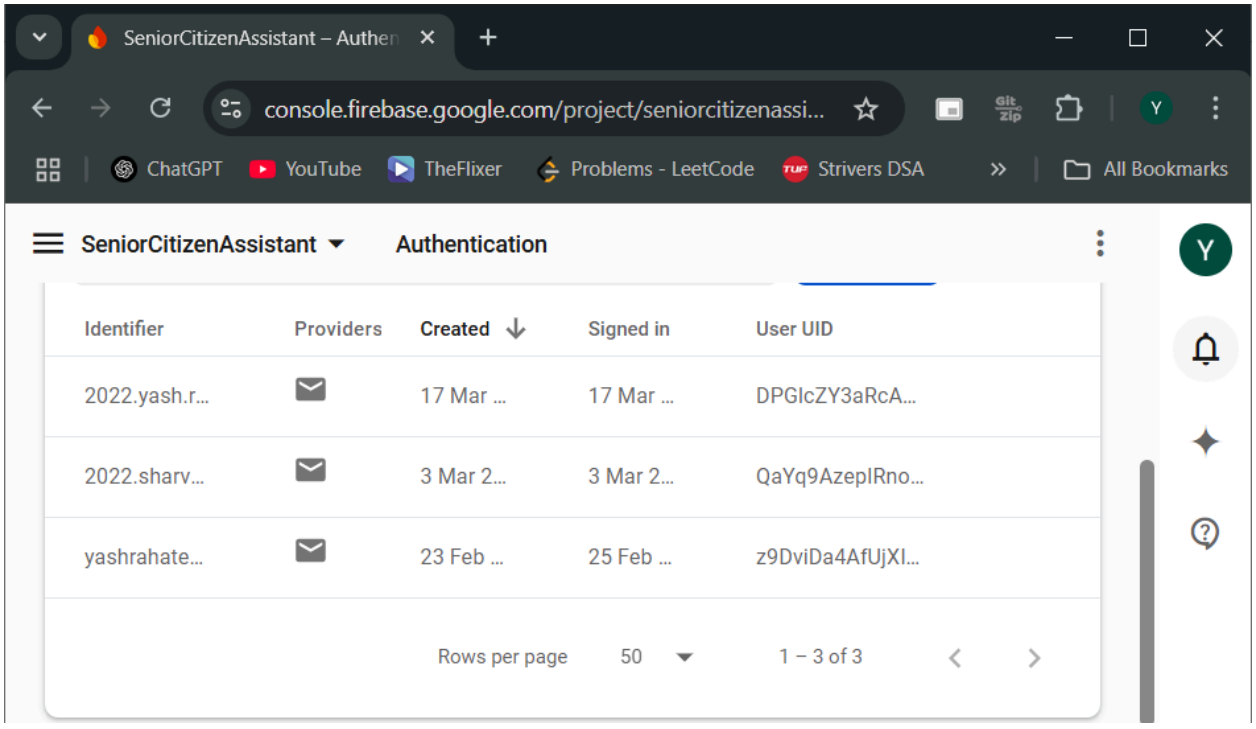

Senior Assistant
Medication Reminders
Add Medication Reminder
Medication
Health
Caregiver
Emergency


23:05
34%
Verify your email for seniorcitizenassistant
External
Inbox
Senior_Assist 23:04
to me
Hello,
Follow this link to verify your email address.
https://seniorcitizenassistant.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=c5cabd_yIjGccbrRult4Dcu3vIDNACHCVpVmASyRpGcAAAGVpSxDSA&apiKey=AIzaSyA_UGUGTVJCrccurRvXrWJBWFST_Mbiz80&lang=en
If you didn't ask to verify this address, you can ignore this email.
Thanks,
Your seniorcitizenassistant team
Reply
Reply all
Forward
Available add-ons:


23:05
34%
seniorcitizenassistant....
seniorcitizenassistant.firebaseapp.com
Your email has been verified
You can now sign in with your new account


console.firebase.google.com/project/seniorcitizenassistant/overview
Firebase
SeniorCitizenAssistant
Project Overview
Project shortcuts
Authentication
Firestore Database
Storage
Realtime Database
Analytics Dashboard
What's new
Genkit
Vertex AI
Product categories
Build
Run
Analytics
SeniorCitizenAssista...
Spark plan
Getting started? Tell Gemini about your project
3 apps
sca (android)
sca (ios)
sca (web)
Waiting for Analytics data...





Conclusion: We successfully integrated Firebase Authentication with Flutter, implementing signup, login, email verification, password reset, and Firestore data storage. During development, we faced issues such as username not fetching from Firestore and Firestore offline errors, which we resolved by forcing online data retrieval, handling cache properly, and ensuring Firestore writes before navigation.

Practical 7

Name: Yash Rahate
Class: D15B
Roll No.: 44

Aim:

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

Theory:

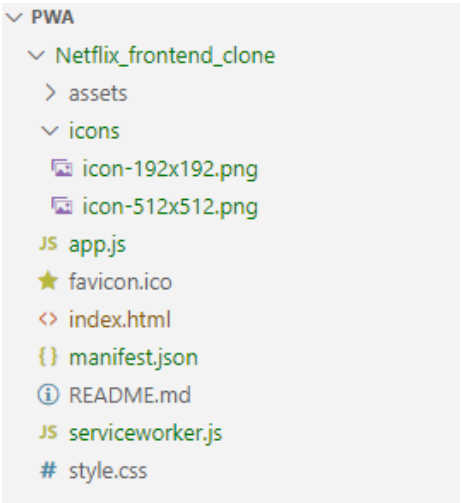
A **Progressive Web App (PWA)** enhances a regular web application by making it installable and capable of running offline like a native app. One of the key components that make a web app a PWA is the **Web App Manifest** file — a simple JSON file that provides essential metadata about the app.
In this experiment, we created a manifest.json file for Netflix clone web application. This file includes details like:

- **App name and short name** for display
- **Start URL**, **display mode** (standalone), **background color**, and **theme color**
- A **description** for the app
- A list of **icons** in required formats (PNG, 192x192 and 512x512, with purpose: "any")
- **Screenshots** of the app to showcase its interface

These details allow modern browsers to recognize the web app as a PWA and prompt users with an **“Add to Home Screen”** option.
We also fixed common validation issues such as:

- Ensuring icons are of minimum size 144x144
- Using proper image formats like PNG and WEBP
- Setting the required sizes and purpose attributes for icons

Folder Structure:



Index.html

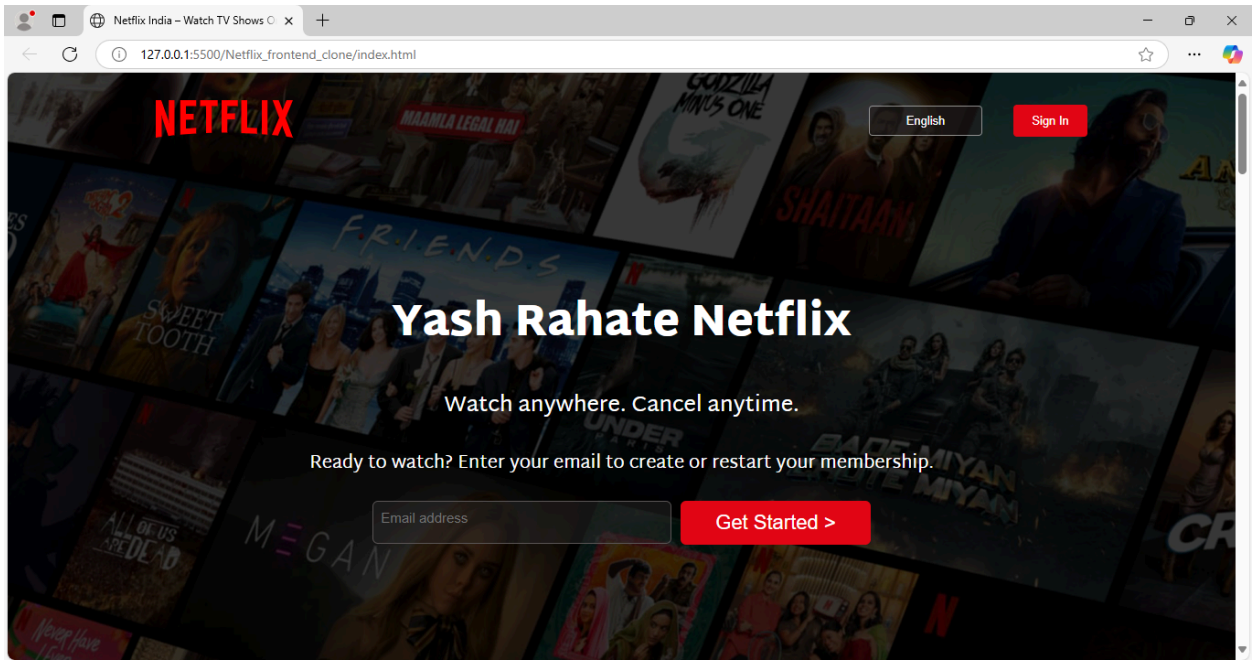
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Netflix India – Watch TV Shows Online, Watch Movies Online</title>
    <link rel="stylesheet" href="style.css" />
    <link rel="manifest" href="manifest.json">
  </head>
  <body>
    <div class="main">
      <nav>
        <span
          ><img  src="assets/images/logo.svg" alt="logo"
        /></span>
        <div>
          <button class="btn">English</button>
          <button class="btn-red-sm">Sign In</button>
        </div>
      </nav>
      <div class="box"></div>
      <div class="hero">
        <span>Yash Rahate Netflix</span>
        <span>Watch anywhere. Cancel anytime.</span>
        <span
          >Ready to watch? Enter your email to create or restart your
          membership.</span
        >
        <div class="herobtn">
          <input type="text" placeholder="Email address" />
          <button class="btn-red">Get Started &gt;</button>
        </div>
      </div>
```

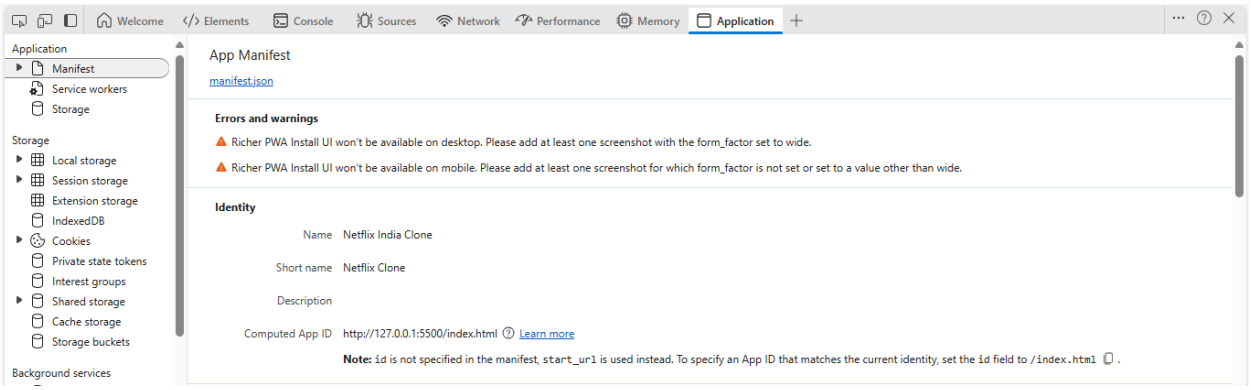
Manifest.json

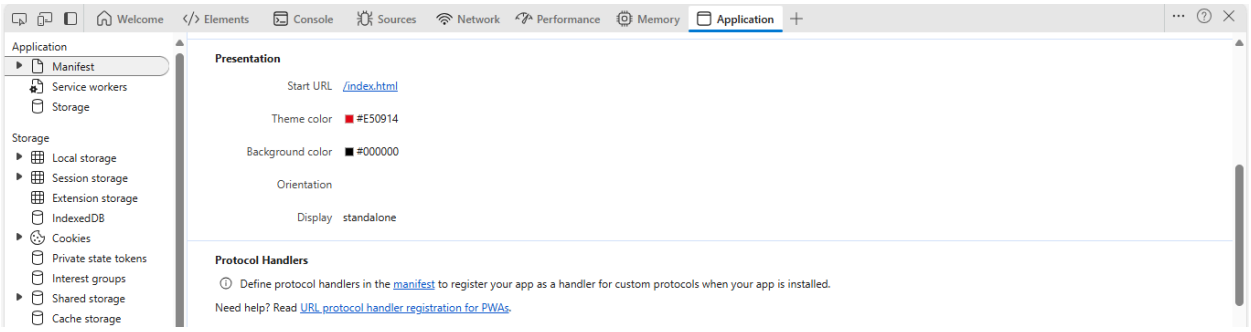
```
{
   "name": "Netflix India Clone",
```

```
  "short_name": "Netflix Clone",
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#000000",
  "theme_color": "#E50914",
  "icons": [
   {
    "src": "icons/icon-192x192.png",
    "sizes": "192x192",
    "type": "image/png"
   },
   {
    "src": "icons/icon-512x512.png",
    "sizes": "512x512",
    "type": "image/png"
   }
  ]
}
```

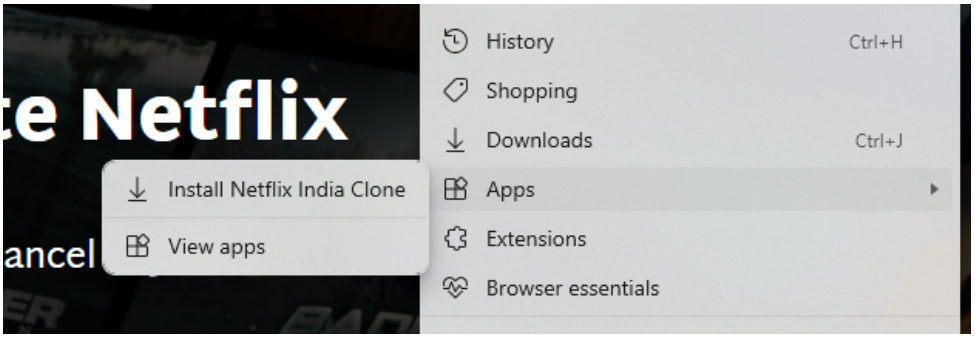
Output:

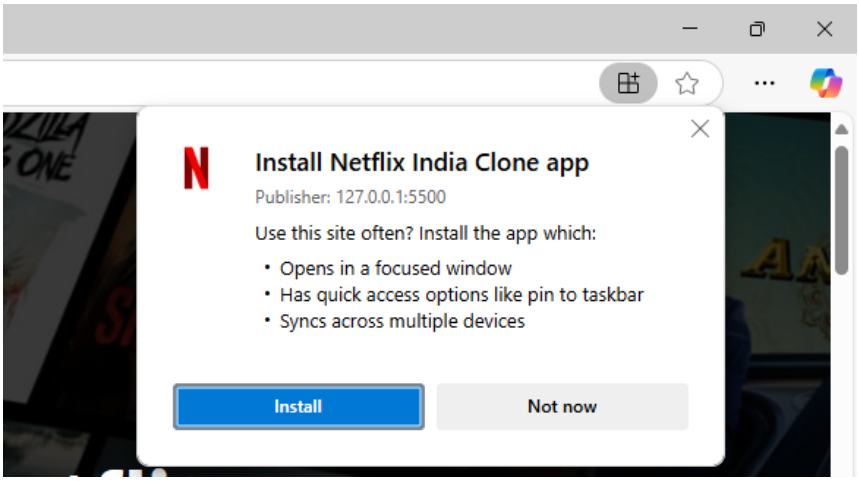


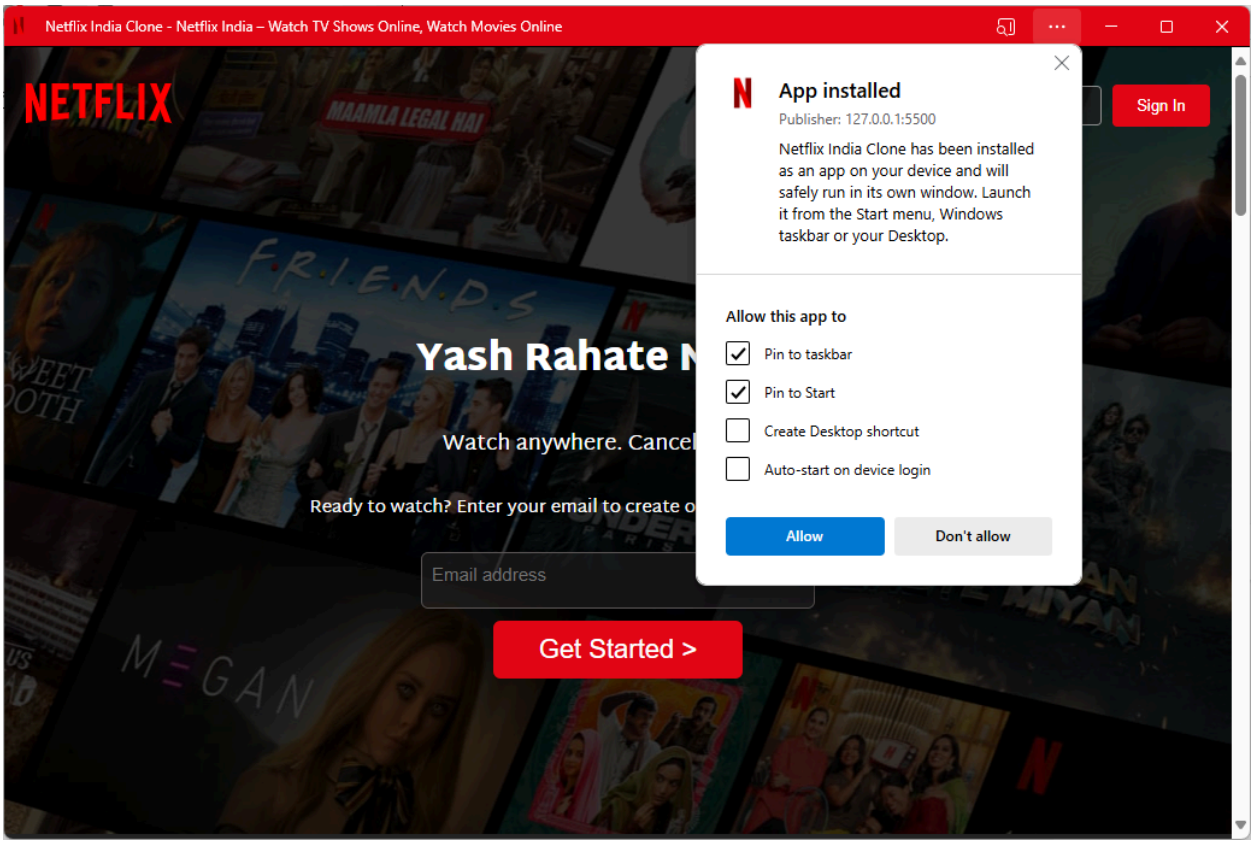

Welcome
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
App Manifest
manifest.json
Errors and warnings
Richer PWA Install UI won't be available on desktop. Please add at least one screenshot with the form_factor set to wide.
Richer PWA Install UI won't be available on mobile. Please add at least one screenshot for which form_factor is not set or set to a value other than wide.
Identity
Name Netflix India Clone
Short name Netflix Clone
Description
Computed App ID http://127.0.0.1:5500/index.html Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /index.html

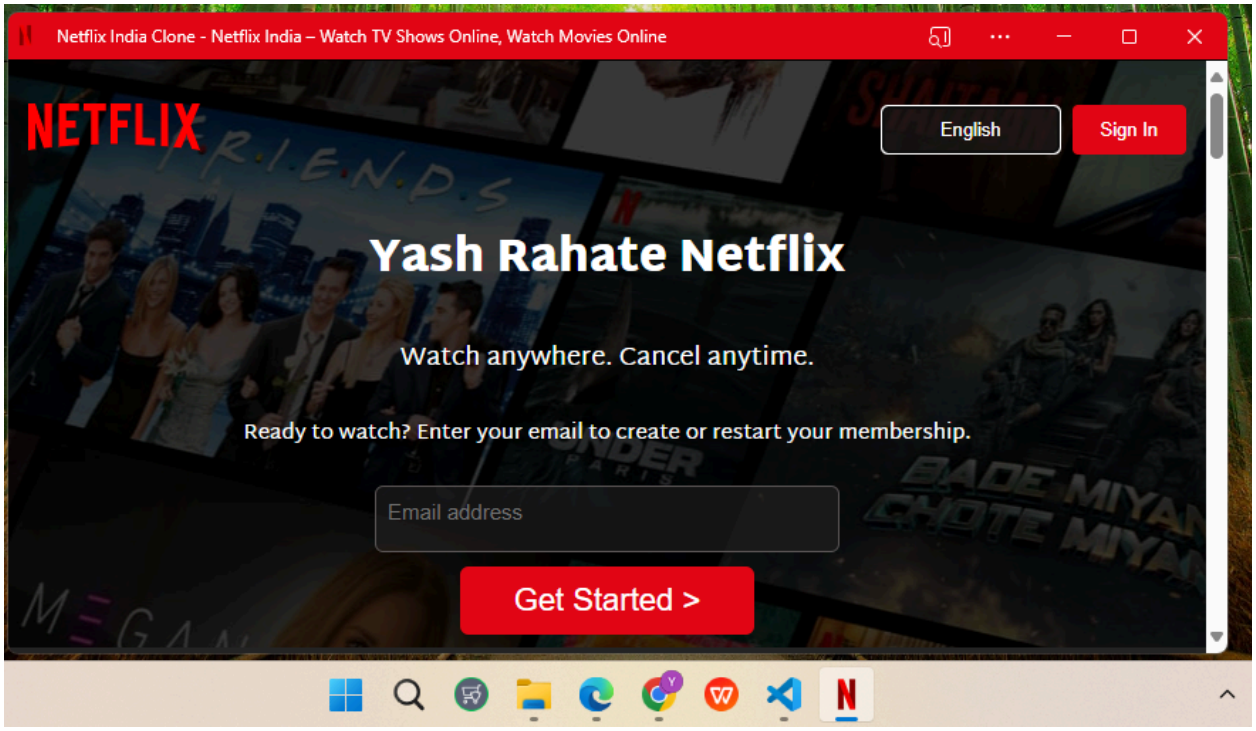

Welcome
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Presentation
Start URL /index.html
Theme color #E50914
Background color #000000
Orientation
Display standalone
Protocol Handlers
Define protocol handlers in the manifest to register your app as a handler for custom protocols when your app is installed.
Need help? Read URL protocol handler registration for PWAs.


Welcome
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitiga...
Notifications
Payment handler
Icons
Show only the minimum safe area for maskable icons
Need help? Read the App Image Generator.
192×192px
image/png
512×512px
image/png
Console
Issues


Netflix
Install Netflix India Clone
View apps
History Ctrl+H
Shopping
Downloads Ctrl+J
Apps
Extensions
Browser essentials


Install Netflix India Clone app
Publisher: 127.0.0.1:5500
Use this site often? Install the app which:
• Opens in a focused window
• Has quick access options like pin to taskbar
• Syncs across multiple devices
Install
Not now


Netflix India Clone - Netflix India – Watch TV Shows Online, Watch Movies Online
NETFLIX
Sign In
App installed
Publisher: 127.0.0.1:5500
Netflix India Clone has been installed as an app on your device and will safely run in its own window. Launch it from the Start menu, Windows taskbar or your Desktop.
Allow this app to
Pin to taskbar
Pin to Start
Create Desktop shortcut
Auto-start on device login
Allow
Don't allow
Watch anywhere. Cancel
Ready to watch? Enter your email to create o
Email address
Get Started >



Conclusion:

By configuring the manifest.json correctly, the Netflix clone web app becomes installable and behaves more like a native mobile application. This improves user engagement and accessibility, especially for mobile users. I faced errors while configuring the icons but is was easily solved by changing the width and height of the images.

Practical 8

Name: Yash Rahate
Class: D15B
Roll No.: 44

Aim:

To code and register a service worker, and complete the install and activation process for a new service worker for the Netflix clone PWA.

Theory:

A Service Worker is a background script that acts as a network proxy, allowing a web app to intercept network requests and serve cached content. It enhances performance and provides offline support.
In our serviceworker.js file, we implemented the following:

- **Install Event:**
  Caches important assets (index.html, CSS, images, manifest) during the installation of the service worker to make them available offline.
- **Activate Event:**
  Deletes old caches that don't match the current version to avoid storage clutter and ensure the latest files are used.
- **Fetch Event:**
  Intercepts all network requests. If the resource is in the cache, it returns the cached version; otherwise, it fetches it from the network.

This setup ensures our eCommerce PWA works smoothly even when offline and improves performance by loading resources from the cache.

Code:

Servicceworker.js

```
const CACHE_NAME = "netflix-clone-cache-v1";
const urlsToCache = [
  "index.html",
  "style.css",
  "app.js",
  "icons/icon-192x192.png",
  "icons/icon-512x512.png"
];

self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
```

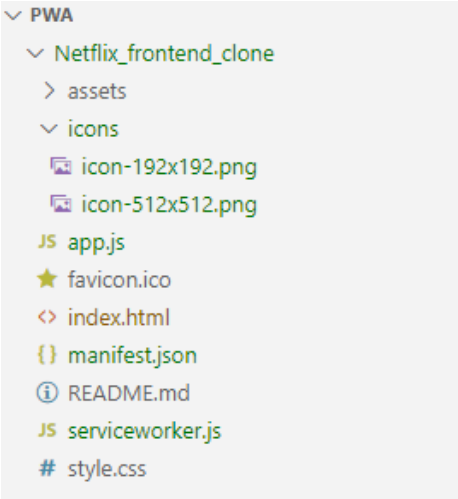
```
      return cache.addAll(urlsToCache);
    }).catch(err => {
      console.error("❌ Caching failed:", err);
    })
  );
});

self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
  return self.clients.claim();
});

self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
      return response || fetch(event.request);
    })
  );
});
```

Index.html

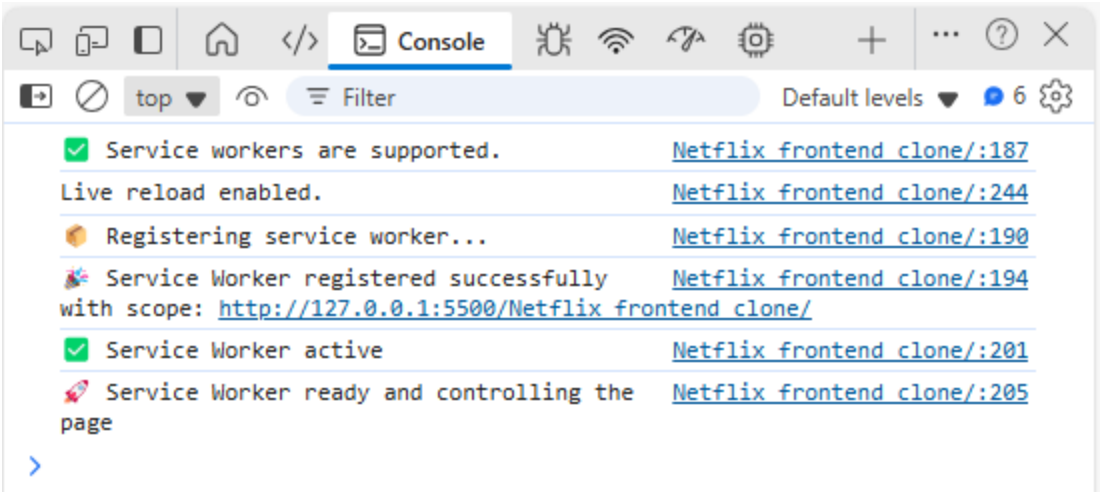
```
<script>
  if ('serviceWorker' in navigator) {
    console.log('✅ Service workers are supported.');

    window.addEventListener('load', () => {
      console.log('📦 Registering service worker...');

      navigator.serviceWorker.register('serviceworker.js')
```

```
      .then(reg => {
        console.log('🎉 Service Worker registered successfully with scope:', reg.scope);

        if (reg.installing) {
          console.log('🔧 Service Worker installing');
        } else if (reg.waiting) {
          console.log('⏸️ Service Worker installed and waiting');
        } else if (reg.active) {
          console.log('✅ Service Worker active');
        }

        navigator.serviceWorker.ready.then(() => {
          console.log('🚀 Service Worker ready and controlling the page');
        });
      })
      .catch(err => {
        console.error('❌ Service Worker registration failed:', err);
      });
  });
 } else {
  console.warn('⚠️ Service workers are not supported in this browser.');
 }
</script>
```

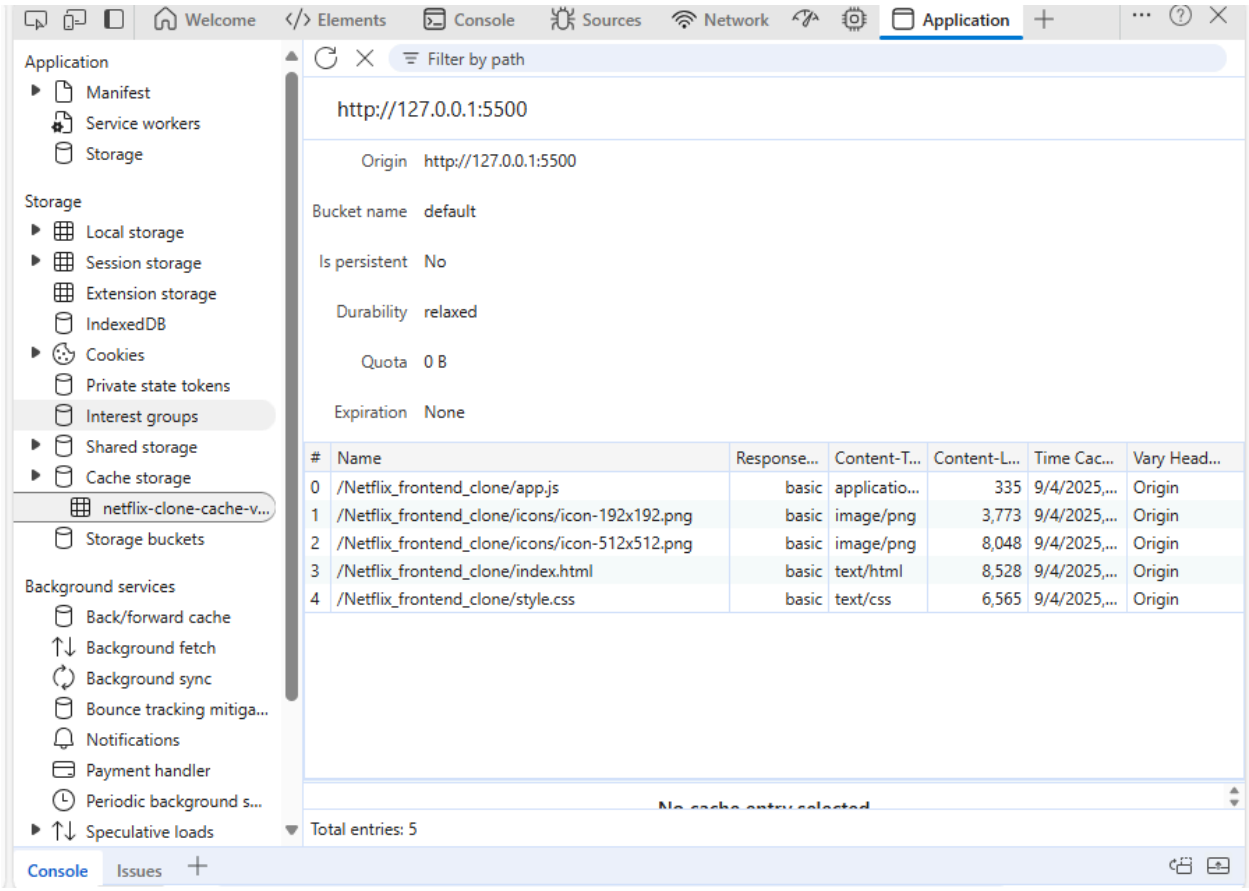
Folder Structure:



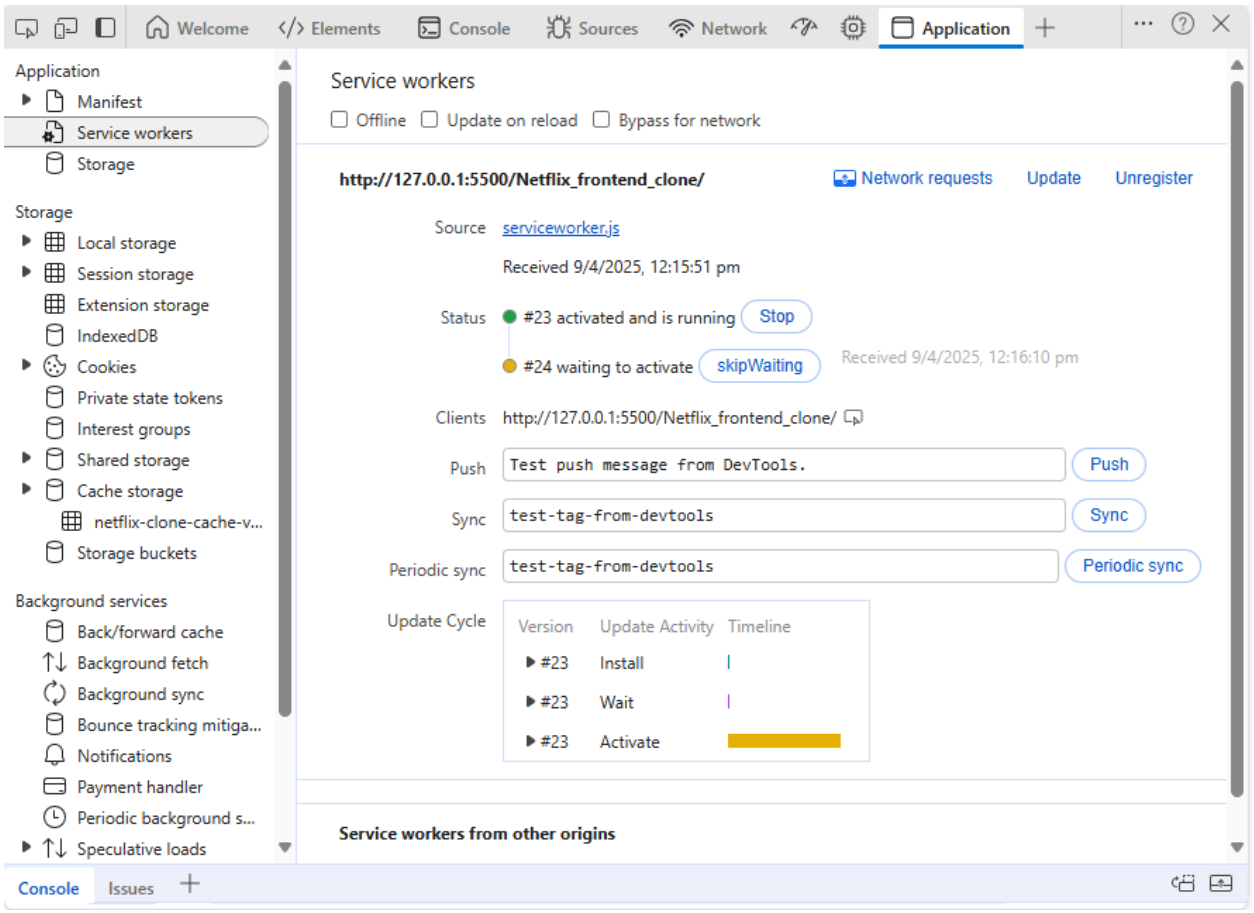
Output:
Console logs



Cache Storage



Service Worker running in the background.



Conclusion:

Through this experiment, we successfully implemented a Service Worker in our PWA, enabling offline functionality and improving load performance. It cached essential resources during installation, managed old caches on activation, and served content efficiently through fetch interception. This demonstrates how Service Workers play a crucial role in enhancing user experience in Progressive Web Apps. We faced an error when the cache storage was not being visible, so we had to delete the storage and reload the site.

Practical 9

Name: Yash Rahate
Class: D15B
Roll No.: 44

Aim:

To implement Service worker events like fetch, sync and push for Netflix clone PWA.

Theory:

Service workers are powerful scripts that run in the background of Progressive Web Apps (PWAs), enabling features like offline access, background sync, and push notifications. In this experiment, we focused on three key service worker events: fetch, sync, and push.

In this experiment, we enhanced our Netflix clone Progressive Web App (PWA) by implementing advanced Service Worker events — **fetch**, **sync**, and **push**.

- The **install** and **activate** events handle caching of files and removal of old caches.
- The **fetch** event intercepts network requests and serves cached resources for better performance and offline support.
- The **sync** event simulates background synchronization when the app regains connectivity.
- The **push** event listens for push notifications and displays them, improving user engagement.

These events help in making PWAs more reliable, fast, and interactive even in low or no network conditions.

Code:

Index.html

```
<script>
  if ('serviceWorker' in navigator && 'SyncManager' in window) {
    window.addEventListener('load', () => {
      navigator.serviceWorker.register('serviceworker.js')
        .then(reg => {
          console.log("✅ Service Worker registered:", reg.scope);

          // Optional: Register a sync event
          reg.sync.register('sync-tag').then(() => {
            console.log('🔁 Sync registered');
          }).catch(err => {
            console.warn('❌ Sync registration failed:', err);
```

```
        });

      }).catch(err => {
        console.error("❌ Service Worker registration failed:", err);
      });
  });

  // Request notification permission
  Notification.requestPermission().then(permission => {
    if (permission === "granted") {
      console.log("🔔 Notifications allowed");
    }
  });

} else {
  console.warn("⚠️ Service Worker or Background Sync not supported.");
}
</script>
```

Serviceworker.js

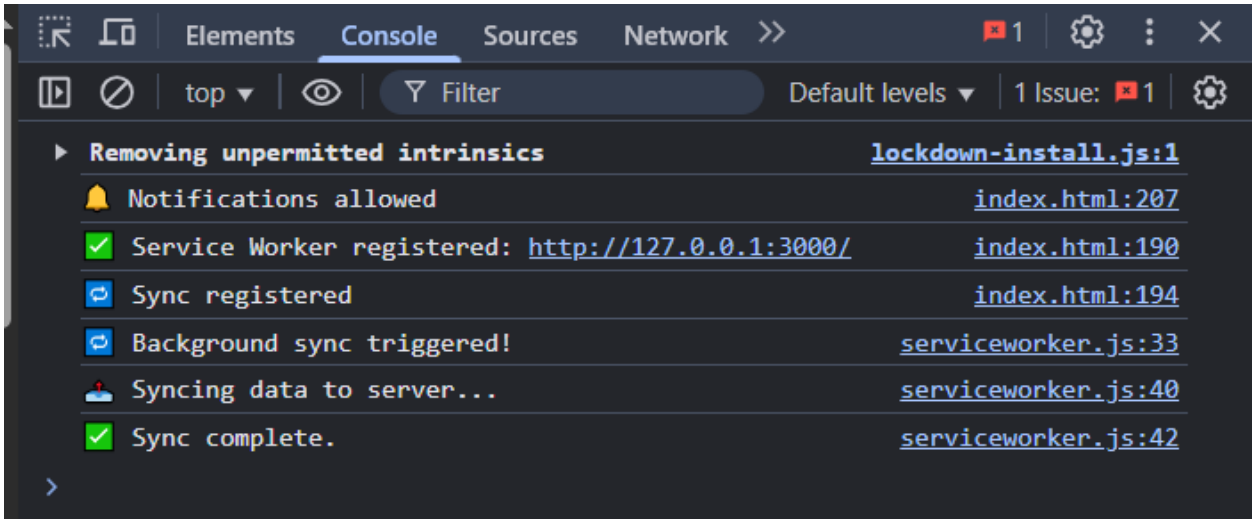
```
const CACHE_NAME = "netflix-clone-cache-v2";
const urlsToCache = [
  "index.html",
  "style.css",
  "app.js",
  "icons/icon-192x192.png",
  "icons/icon-512x512.png"
];

// Install event: cache assets
self.addEventListener("install", (event) => {
  console.log("📦 Installing service worker...");
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => cache.addAll(urlsToCache))
      .catch((err) => console.error("❌ Cache add failed", err))
  );
```

```
});

// Fetch event: serve from cache or network
self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches.match(event.request)
      .then((response) => {
        return response || fetch(event.request);
      })
  );
});

// Sync event: background sync logic (mocked)
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-tag") {
    console.log("🔁 Background sync triggered!");
    event.waitUntil(syncDataWithServer());
  }
});

async function syncDataWithServer() {
  // Simulate sending queued data to server
  console.log("📤 Syncing data to server...");
  await new Promise(resolve => setTimeout(resolve, 1000));
  console.log("✅ Sync complete.");
}

// Push event: show notification
self.addEventListener("push", (event) => {
  const data = event.data?.text() || "Default Push Message";
  const options = {
    body: data,
    icon: "icons/icon-192x192.png",
    badge: "icons/icon-192x192.png"
  };
  event.waitUntil(
    self.registration.showNotification("🔥 Push Notification, Welcome the Yash Rahate Netflix ",
options)
```

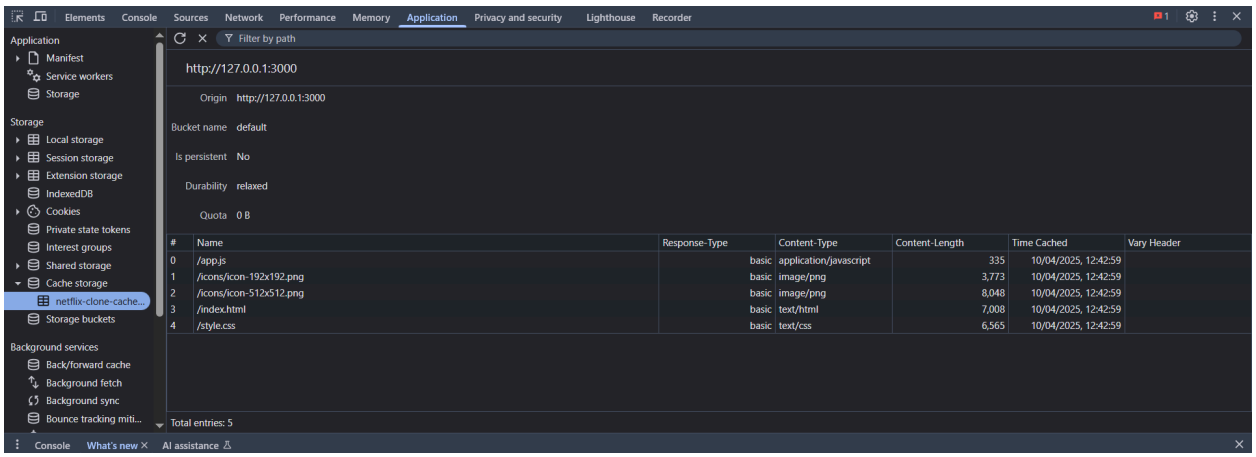
```
  );
});
```

Output:

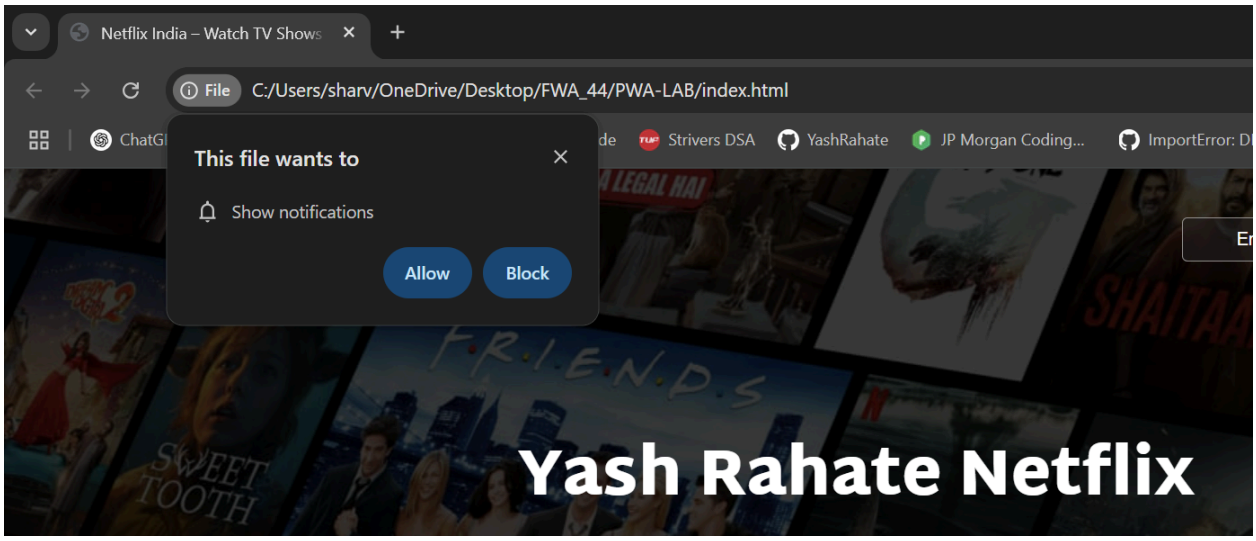
Working of service worker



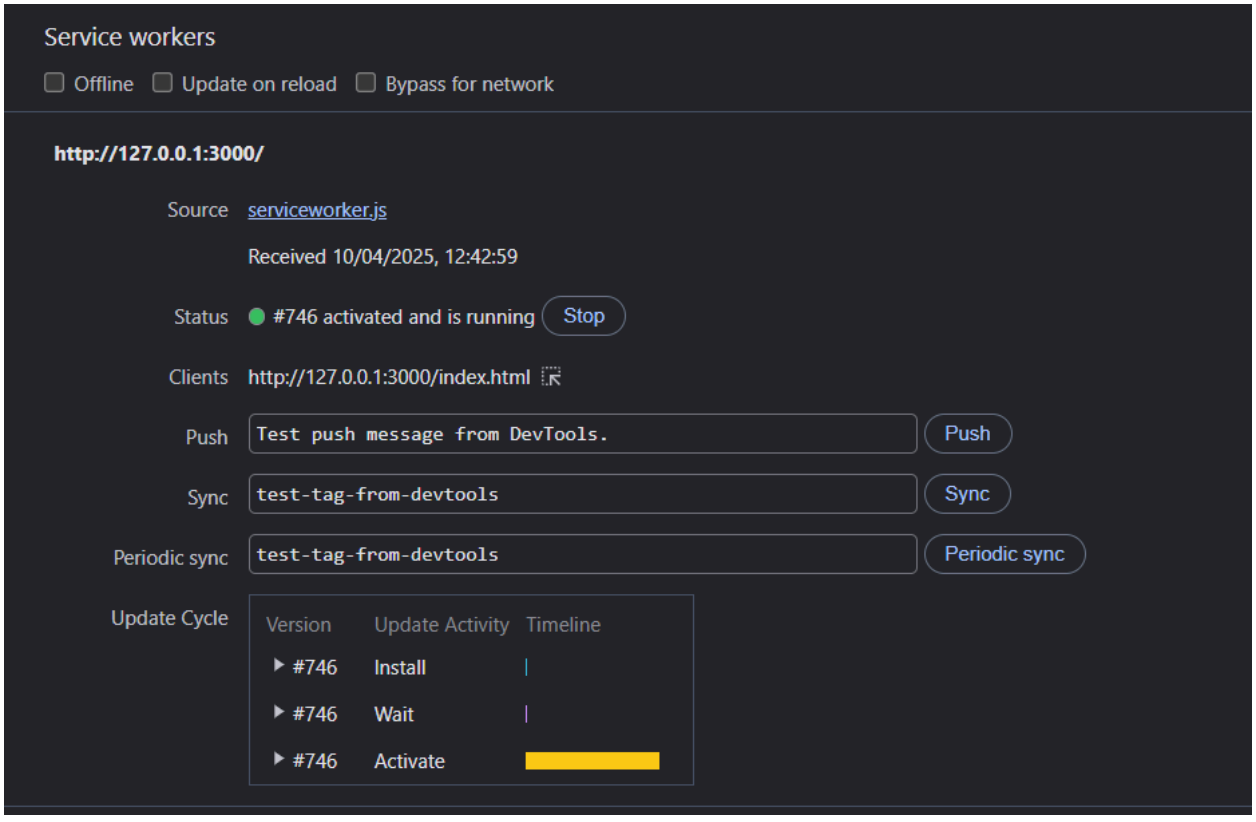
Cache

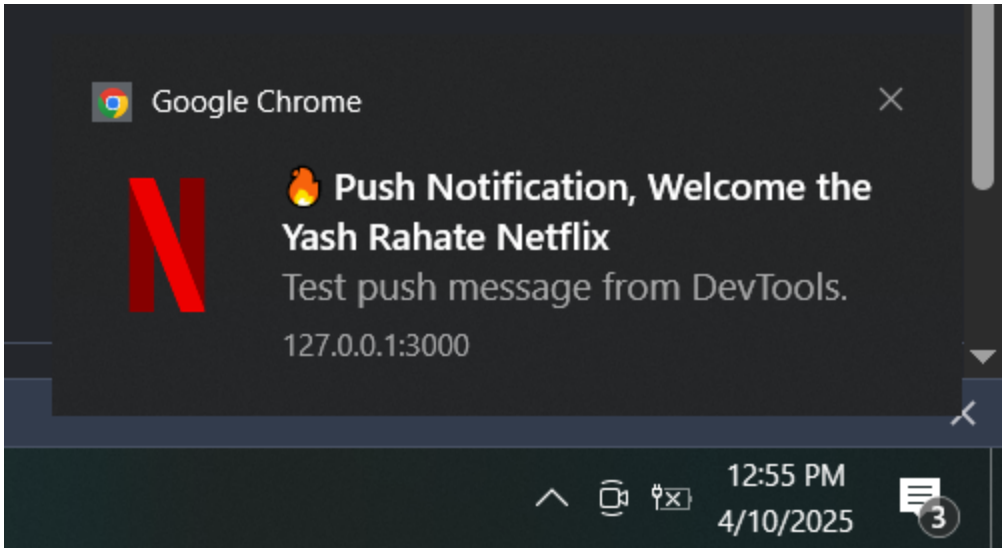


Sending push notification.



Test: Push Notification sent





Conclusion:

By implementing fetch, sync, and push events in the Service Worker, we made our Netflix clone PWA capable of working offline, syncing data in the background, and sending push notifications. This ensures a better user experience and highlights the power of PWAs in building modern web applications.

Practical 10

Name: Yash Rahate
Class: D15B
Roll No.: 44

Aim:

To study and implement deployment of Netflix clone PWA to GitHub Pages.

Theory:

GitHub Pages is a free hosting service provided by GitHub to publish static websites directly from a GitHub repository. It supports HTML, CSS, JavaScript, and other front-end files, making it ideal for hosting PWAs.

What we implemented in our code

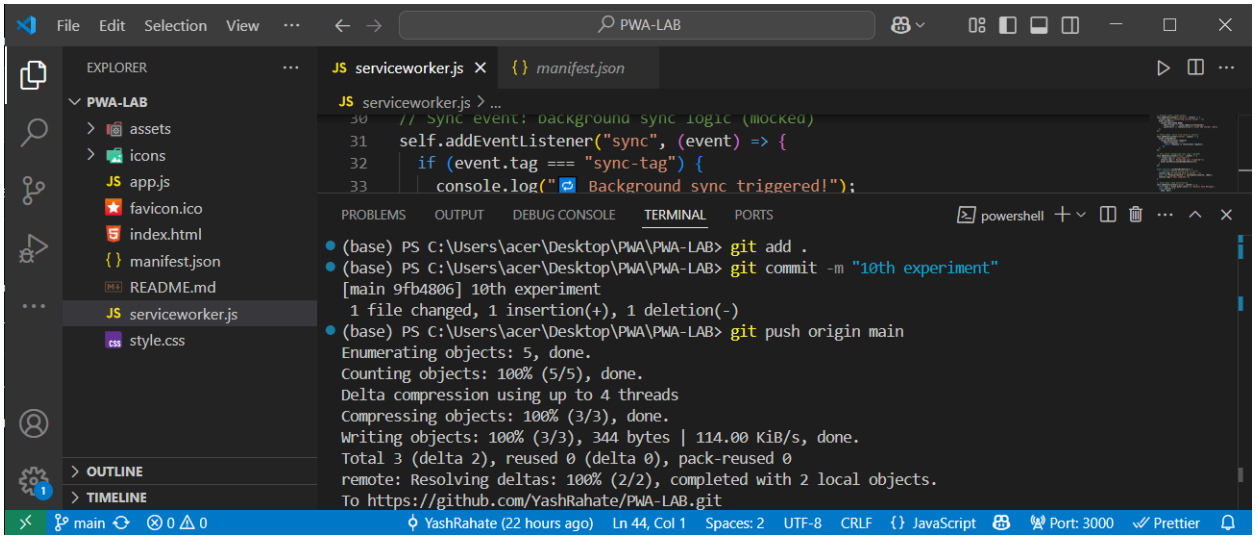
In this experiment, we deployed our Netflix clone PWA project, "Eco Friendly Gadgets Collection", to GitHub Pages using these steps:

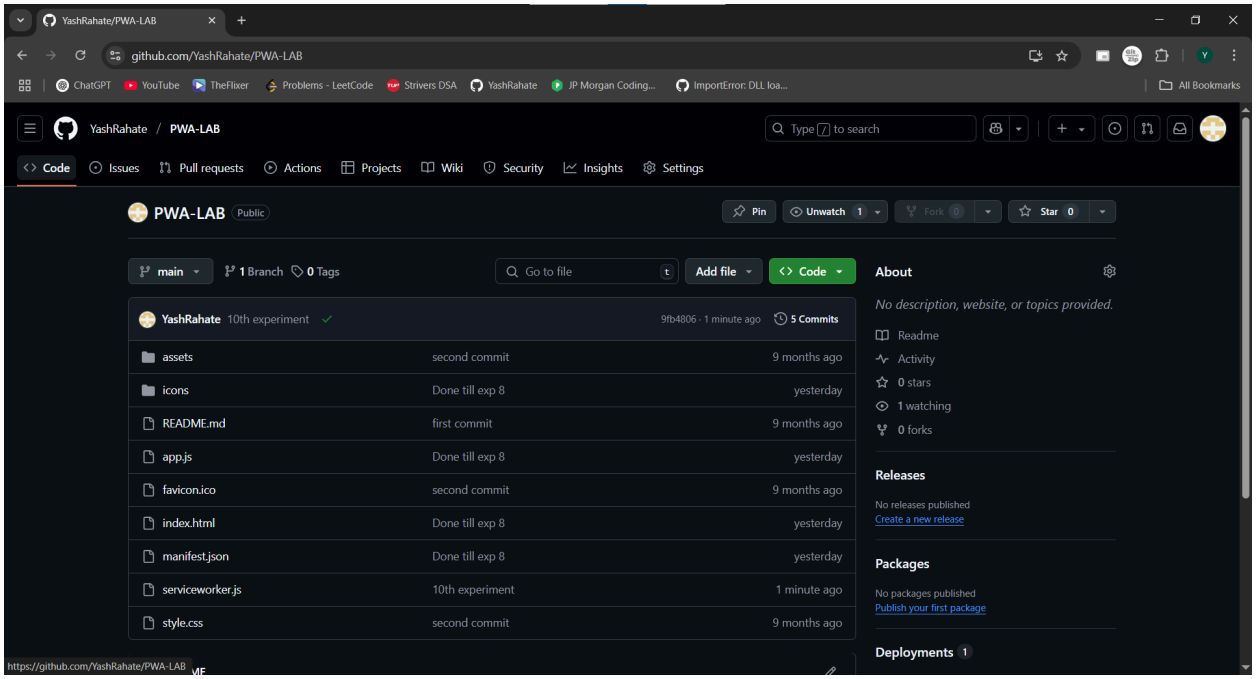
- We ensured all necessary PWA files were in place, such as index.html, manifest.json, serviceworker.js, and offline assets.
- The index.html file used relative paths correctly so resources load properly after deployment.
- We registered the service worker to enable offline access and caching of essential files.
- The manifest.json provided metadata to support "Add to Home Screen" functionality.
- We created a GitHub repository and pushed all project files to it.
- Then, from GitHub Settings → Pages, we selected the main branch and the / (root) folder as the source.
- After saving, GitHub generated a live link through which we accessed the deployed PWA.

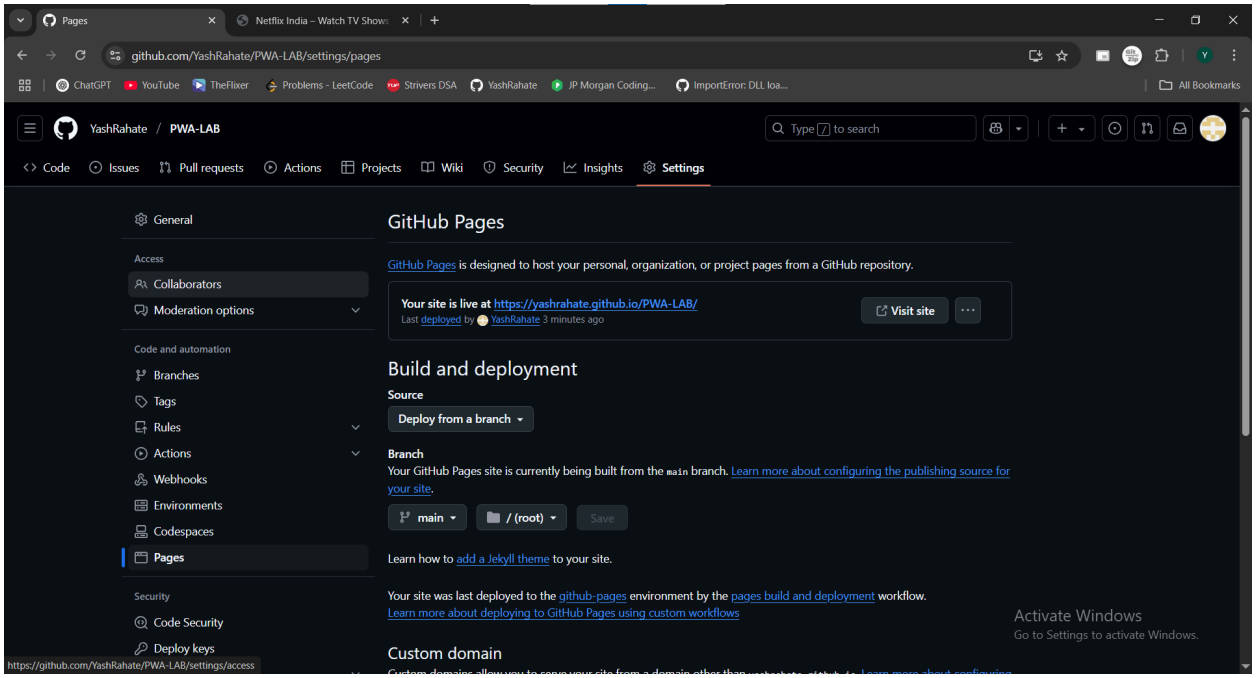
This deployment made our PWA fully functional and accessible online with service worker support, offline capabilities, and installability features.

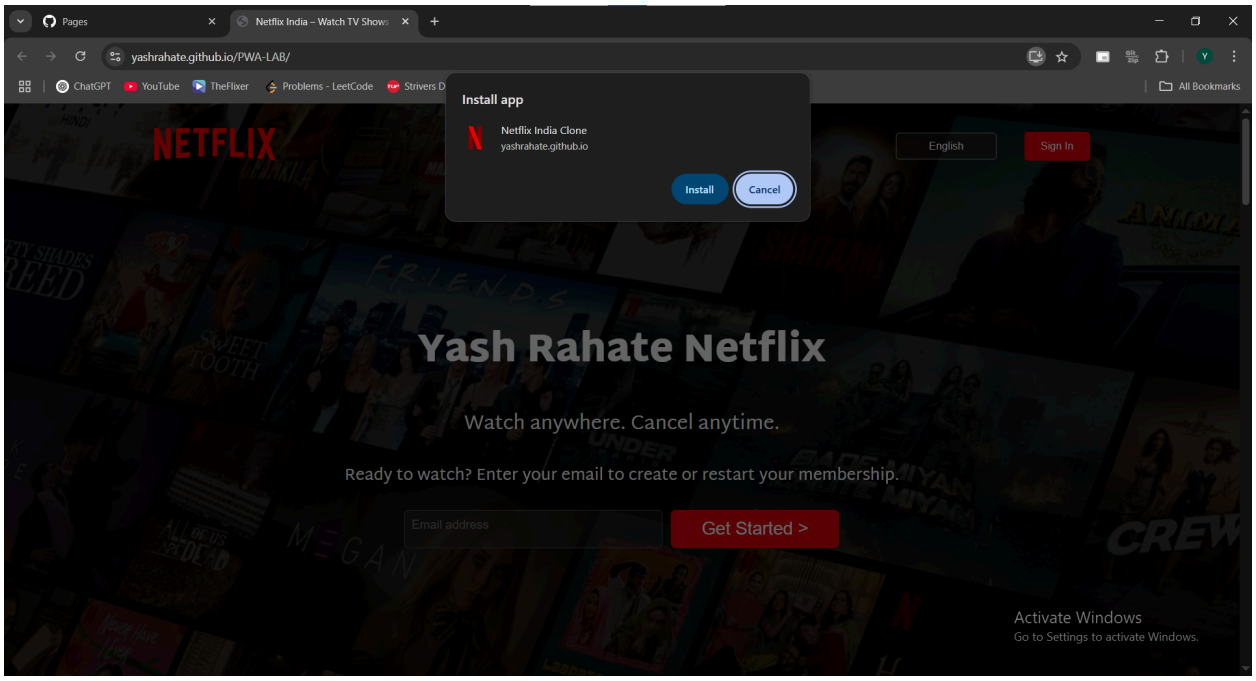
Screenshots:

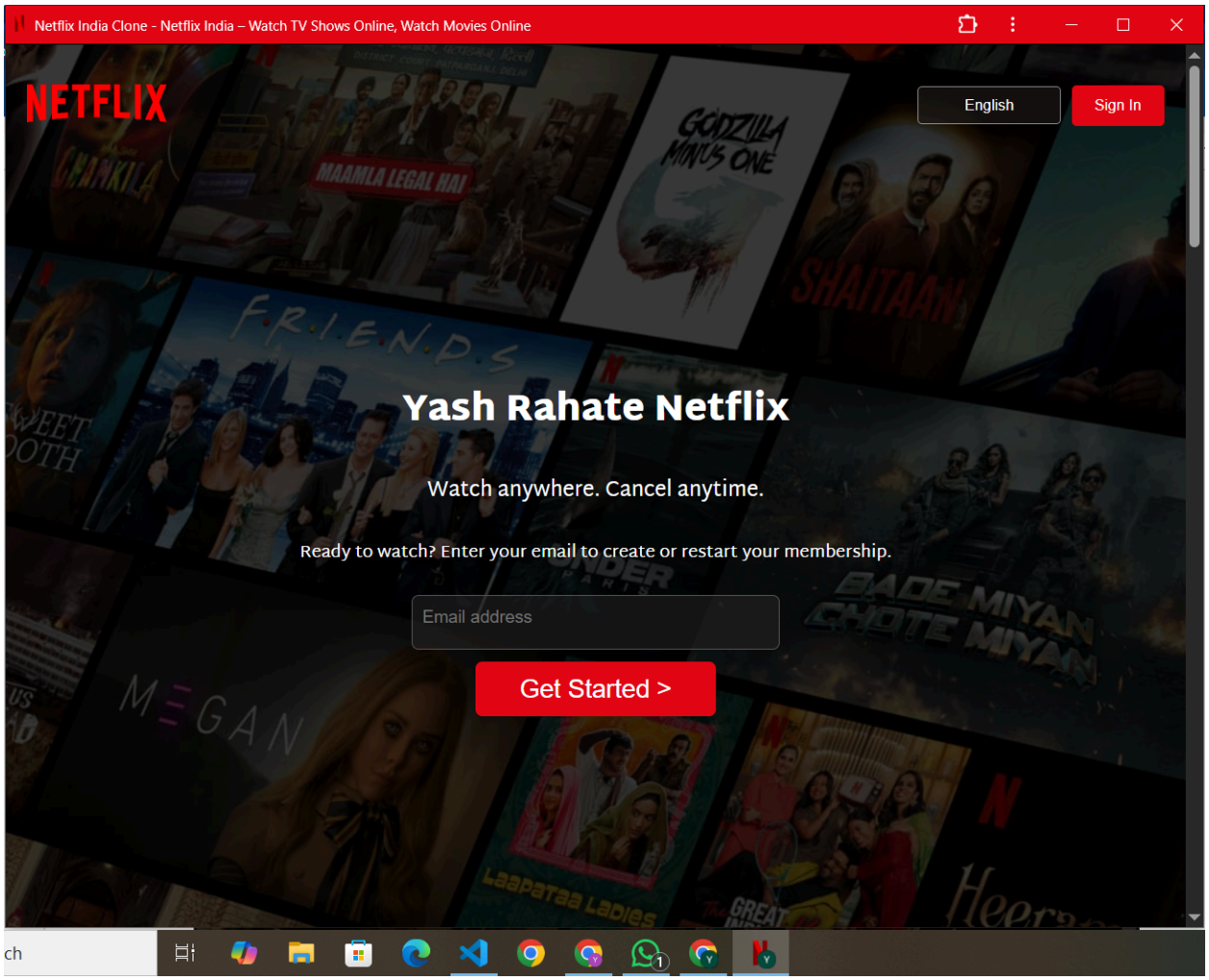
Creation of new repository in public settings.






Pages
Netflix India – Watch TV Shows
github.com/YashRahate/PWA-LAB/settings/pages
YashRahate / PWA-LAB
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code Security
Deploy keys
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://yashrahate.github.io/PWA-LAB/
Visit site
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the main branch.
main
/ (root)
Save
Learn how to add a Jekyll theme to your site.
Your site was last deployed to the github-pages environment by the pages build and deployment workflow.
Custom domain
Activate Windows
Go to Settings to activate Windows.


Pages
Netflix India – Watch TV Shows
yashrahate.github.io/PWA-LAB/
Install app
Netflix India Clone
yashrahate.github.io
Install
Cancel
NETFLIX
English
Sign In
Yash Rahate Netflix
Watch anywhere. Cancel anytime.
Ready to watch? Enter your email to create or restart your membership.
Email address
Get Started >
Activate Windows
Go to Settings to activate Windows.



The system is available for downloading from the link

Conclusion:

In this experiment, we deployed the website PWA on the internet with the help of git pages. We took proper care of manifest.json file and serviceworker.js file. A link was generated by Git Pages. It took a lot of time for the link to be generated, but after the link creation, the website was visible and working.

Practical 11

Name: Yash Rahate
Class: D15B
Roll No.: 44

Aim:

To use google Lighthouse PWA Analysis Tool to test the PWA functioning

Theory:

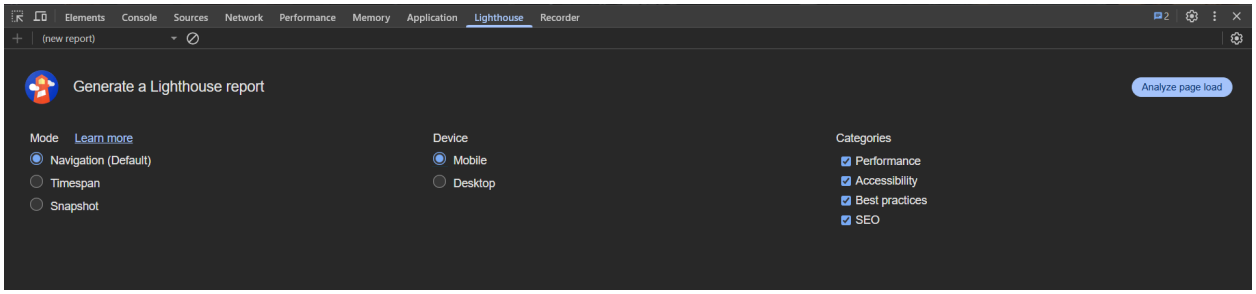
Lighthouse is a tool provided by Google that helps us check how well our web app performs as a Progressive Web App (PWA). It analyzes important factors like performance, accessibility, SEO, best practices, and PWA features.

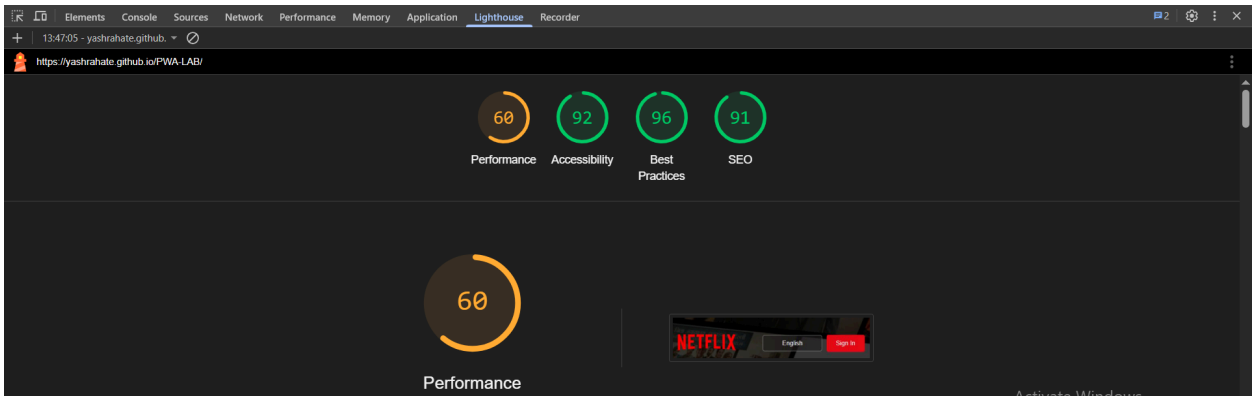
In this experiment, we used Lighthouse in Chrome DevTools to test our deployed PWA. Lighthouse runs a series of audits and gives a score out of 100 for each category. These scores show how well our app performs and where we can improve.

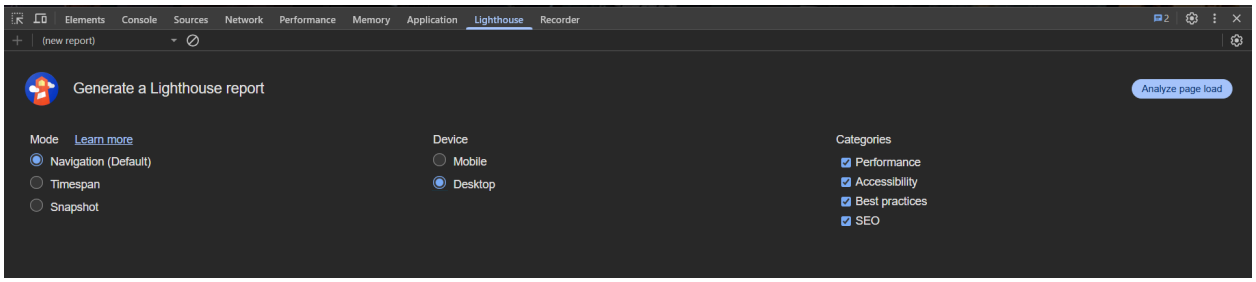
What we implemented:

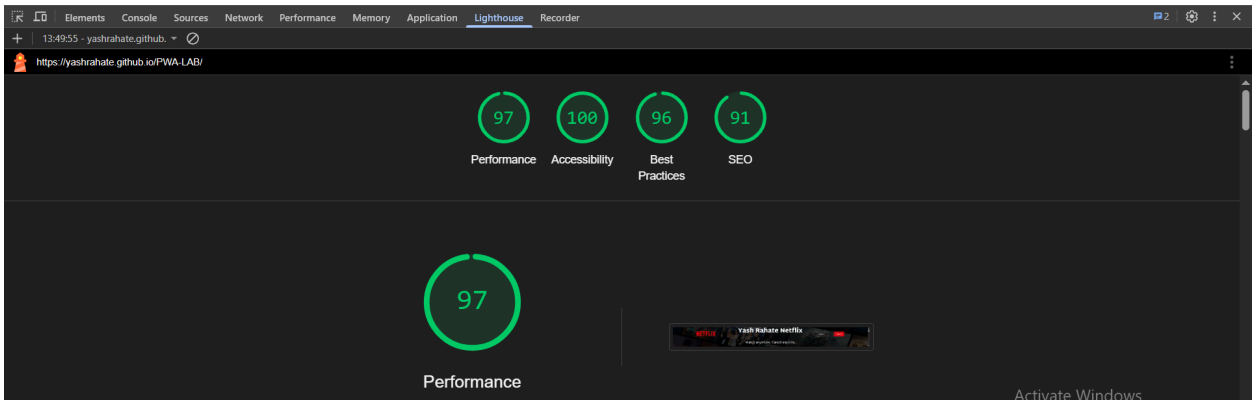
- We deployed our PWA Festival-Combo on GitHub Pages.
- Then, we ran Lighthouse from Chrome DevTools on the live site.
- Lighthouse gave us excellent scores:
  Performance: 97
  Accessibility: 100
  Best Practices: 96
  SEO: 91

These high scores show that our app is fast, user-friendly, and optimized for search engines. Some minor suggestions were also given, like improving color contrast and adding a meta description, which can help make the app even better.









Conclusion:

In this experiment, we tested the PWA using Google Lighthouse and achieved high scores in performance, accessibility, best practices, and SEO. Initially, we faced an error where all images weren't visible, which was resolved by adding the repository name into the serviceworker.js file before the image file.