CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING
CHANDUBHAI S. PATEL INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Subject : Mobile Application Development **Semester:** 5

Subject Code: AIML308 **Academic Year :**2025-26(ODD)

NAME : CHOKSI OM CHIRAGBHAI **ID: 23AIML010**

Practical 2

Problem Definition

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Develop a temperature converter app using Dart functions and input widgets.

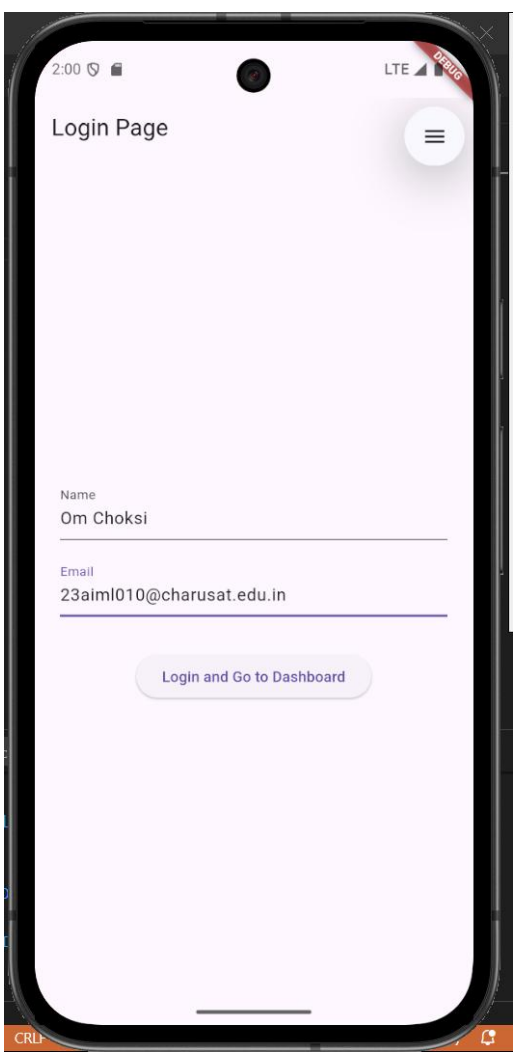
Technical Approach:

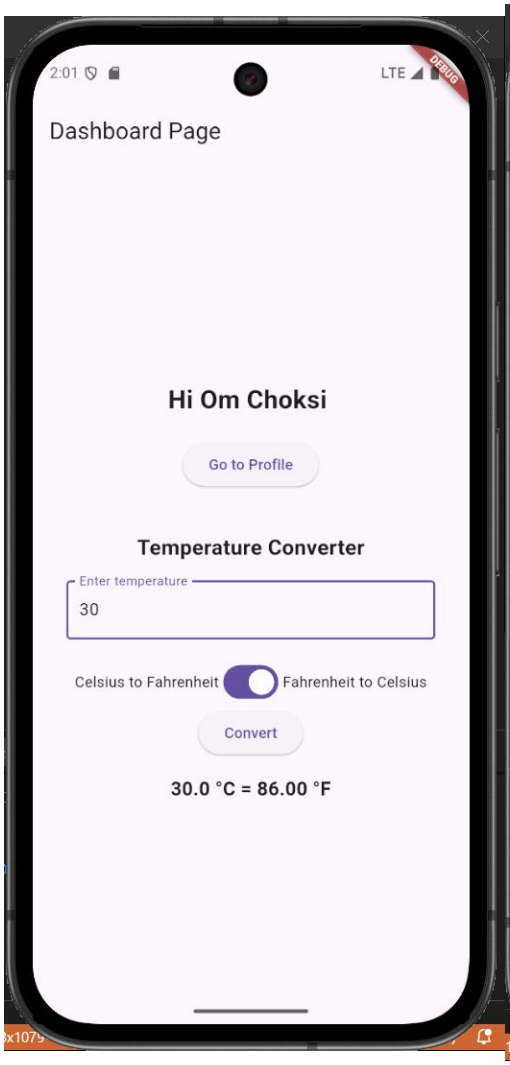
This Flutter app is a multiscreen application with login, dashboard (temperature converter), and profile screens. It uses setState for UI updates, TextField for user input with double.tryParse validation, and Navigator for routing. Core widgets include StatefulWidget, TextField, ElevatedButton, Switch, and Scaffold.

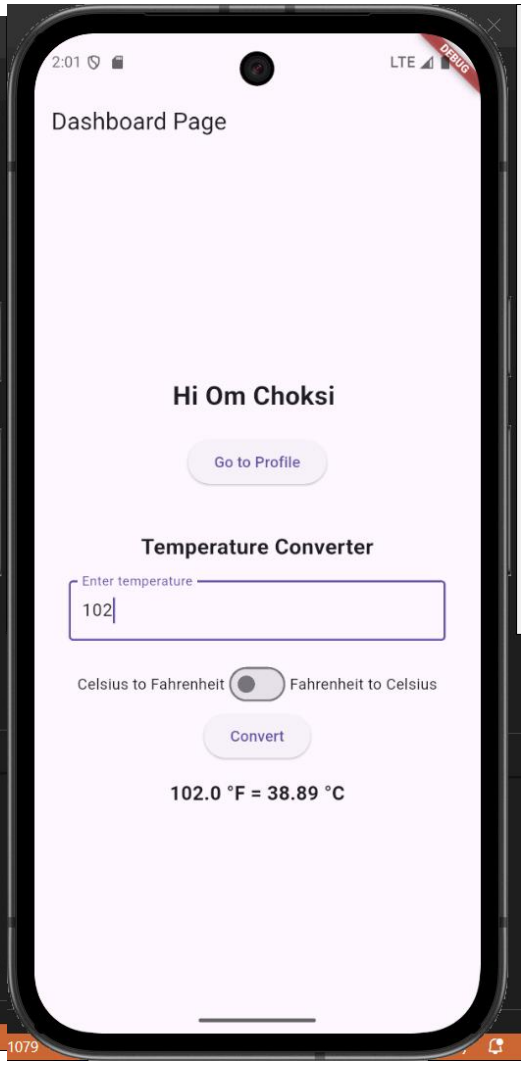
File Structure:

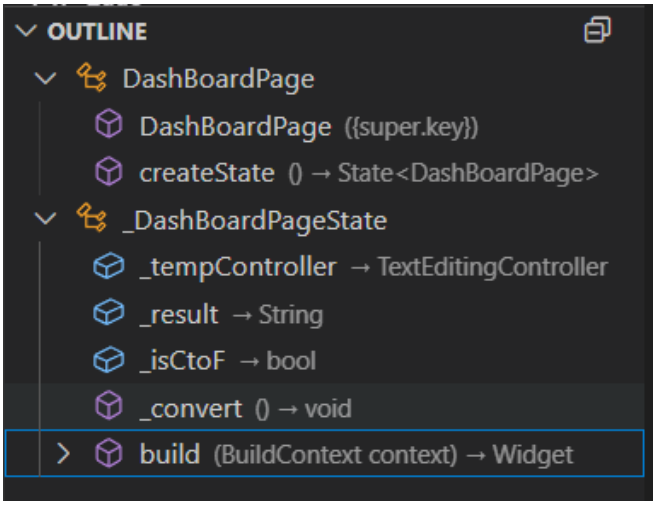
```
lib/
├── main.dart          # Main entry point of the Flutter app, sets up routes and app structure
├── login.dart         # Login screen with TextField inputs for name and email, handles navigation to dashboard
├── dashboard.dart     # Dashboard screen temperature converter with Switch for mode toggle and conversion logic
└── profile.dart       # Profile screen that displays username and email retrieved from navigation arguments
```

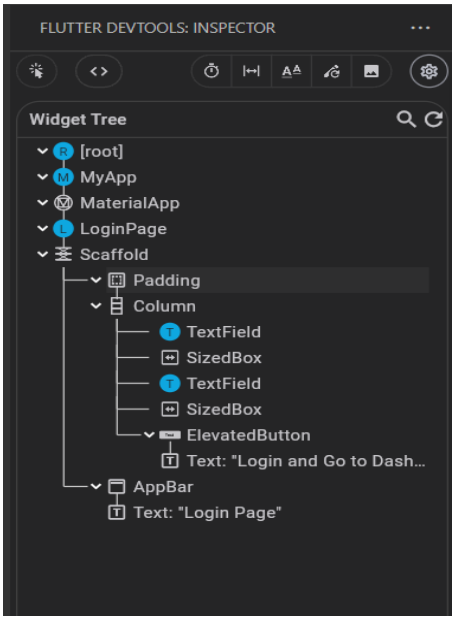
Screenshots:











Key Questions:

1. How to take and validate user input?
Ans: User input is captured using TextField widgets with controllers (e.g., _tempController in dashboard.dart). Validation is performed by parsing inputs with double.tryParse in the _convert function, displaying error messages via setState if invalid, ensuring only numeric values are processed for temperature conversion.

2. What is the role of Dart functions?
Ans: Dart functions encapsulate reusable logic, such as the _convert function in dashboard.dart, which handles input validation, calculation, and UI updates. They enable modular code by separating concerns, like navigation in the login button's onPressed callback, allowing for clean, maintainable event handling and state management.

3. How to update UI on user action?
Ans: UI updates are triggered using setState in stateful widgets, as seen in the Switch's onChanged callback and the _convert function in dashboard.dart, which rebuilds the widget tree to reflect changes like toggled modes or computed results, providing immediate visual feedback.

Key Skills to be Understand:

Dart Basics, Stateful Widget, Event Handling