EXPERIMENT NO: - 04

Name:- Ronak Katariya **Class:-** D15A **Roll:No: - 23**

AIM: - To create an interactive Form using form widget.

Theory: -

A form in Flutter is a structured container that collects user input through various fields like text fields, dropdowns, checkboxes, and buttons. It plays a crucial role in applications that require user data entry, such as login pages, registration forms, and feedback submissions. Flutter provides the Form widget, which works alongside TextFormField and other input elements to manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience.

When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields. The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Creation of a Form

- ➢ While creating a form in Flutter, the **Form widget** is essential as it acts as a container for grouping multiple form fields and managing validation.
- ➢ A **GlobalKey<FormState>** is required to uniquely identify the form and enable validation or data retrieval from the form fields.
- ➢ The **TextFormField widget** is used to provide input fields where users can enter data such as names, phone numbers, or email addresses.
- ➢ To enhance the appearance and usability of input fields, **InputDecoration** is used, allowing customization of labels, icons, borders, and hint text.
- ➢ Validation plays a crucial role in forms, and the **validator property** within **TextFormField** ensures user input meets specific criteria before submission.

- Different types of input require appropriate **keyboard types**, such as TextInputType.number for numeric fields or TextInputType.emailAddress for email fields.
- Proper **state management** is needed to store and retrieve user input, ensuring the form data is processed correctly.
- A **submit button** is necessary to trigger form validation and submit the collected data for further processing.

Some Properties of Form Widget

- **key:** A GlobalKey that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.
- **child:** The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.
- **autovalidateMode:** An enum that specifies when the form should automatically validate its fields.

Some Methods of Form Widget

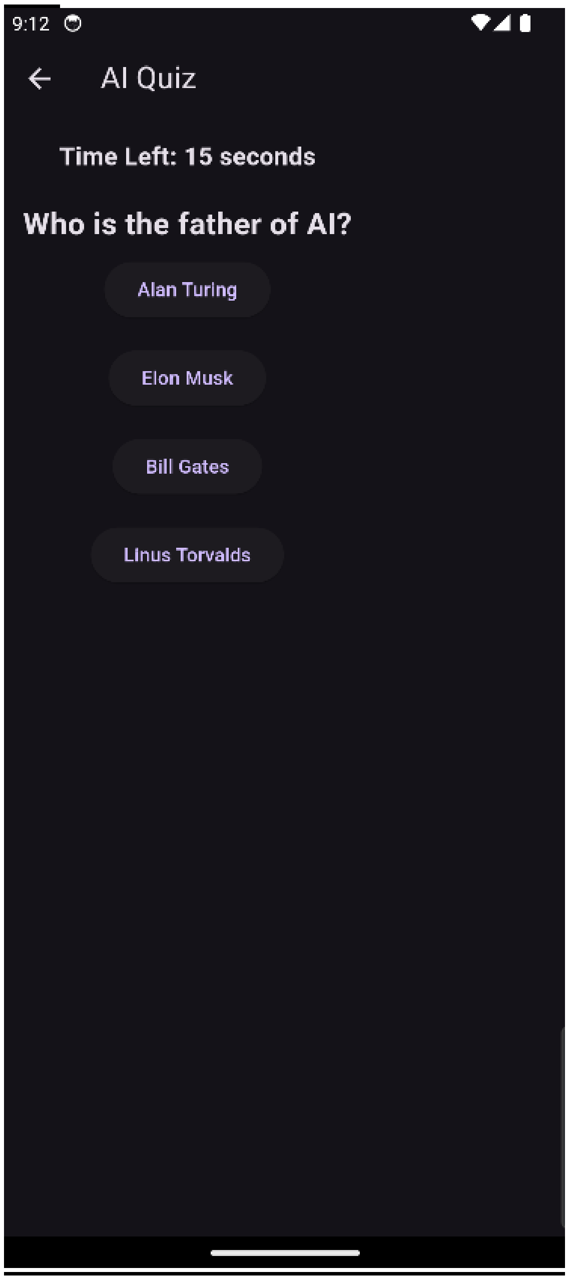
- **validate():** This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.
- **save():** This method is used to save the current values of all form fields. It invokes the onSaved callback for each field. Typically, this method is called after validation succeeds.
- **reset():** Resets the form to its initial state, clearing any user-entered data.
- **currentState:** A getter that returns the current FormState associated with the Form.

Code: -

quiz_screen.dart

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:quiz/screens/quiz_summary.dart';

class QuizScreen extends StatefulWidget {
  final String category;
  const QuizScreen({super.key, required
this.category});

  @override
  _QuizScreenState createState() =>
_QuizScreenState();
}

class _QuizScreenState extends State<QuizScreen>
{
  int currentQuestion = 0;
  int score = 0;
  late Timer timer;
  int timeLeft = 20;
  bool answered = false;
  List<Map<String, dynamic>> userResponses = [];

  final Map<String, List<Map<String, dynamic>>>
questionBank = {
    "CNS": [
      {
        "question": "What is a firewall?",
        "options": ["Security Device", "OS",
"Protocol", "Network"],
        "answer": "Security Device"
      },
      {
        "question": "What is encryption?",
        "options": ["Encoding data", "Deleting data",
"Sending messages", "Accessing files"],
        "answer": "Encoding data"
      },
      {
        "question": "What is a VPN?",
        "options": ["Virtual Private Network",
"Variable Public Network", "Visual Private
Network", "Virtual Protected Network"],
        "answer": "Virtual Private Network"
      },
      {
        "question": "What does IDS stand for?",
        "options": ["Intrusion Detection System", "Internal
Data Security", "Internet Data Service", "Integrated
Defense System"],
        "answer": "Intrusion Detection System"
      },
      {
        "question": "What is a DDoS attack?",
        "options": ["Distributed Denial of Service", "Direct
Denial of Service", "Distributed Data Service", "Direct
Data Service"],
        "answer": "Distributed Denial of Service"
      },
      {
        "question": "What is a proxy server?",
        "options": ["Intermediary for requests", "Data
storage device", "Network protocol", "Firewall type"],
        "answer": "Intermediary for requests"
      },
      {
        "question": "What does SSL stand for?",
        "options": ["Secure Sockets Layer", "Standard
Security Layer", "Simple Sockets Layer", "Secure
System Layer"],
        "answer": "Secure Sockets Layer"
      },
      {
        "question": "What is malware?",
        "options": ["Malicious software", "Machine
learning software", "Management software", "Multi-
layer software"],
        "answer": "Malicious software"
      },
      {
        "question": "What is phishing?",
        "options": ["Fraudulent attempt to obtain sensitive
info","Data encryption method","Network security
measure","Type of firewall"],
        "answer":"Fraudulent attempt to obtain sensitive
info"
      },
  {
      "question":"What is a botnet?",
      "options":["Network of infected
devices","Type of firewall","Data encryption
method","Software application"],
      "answer":"Network of infected devices"
    }
   ],
   "OS": [
    {"question": "What is CPU scheduling?",
"options": ["Round Robin", "Mutex", "Binary
Search", "Recursion"], "answer": "Round Robin"},
    {"question": "What does OS stand for?",
"options": ["Operating System", "Open Source",
"Output System", "Offline Storage"], "answer":
"Operating System"},
   ],
   "Cloud Computing": [
    {"question": "What is SaaS?", "options":
["Software as a Service", "Storage as a Service",
"Security as a Service", "System as a Service"],
"answer": "Software as a Service"},
    {"question": "AWS stands for?", "options":
["Amazon Web Services", "Azure Web Solutions",
"Advanced Web Server", "Automated Web
Security"], "answer": "Amazon Web Services"},
   ],
   "AI": [
    {"question": "Who is the father of AI?",
"options": ["Alan Turing", "Elon Musk", "Bill
Gates", "Linus Torvalds"], "answer": "Alan
Turing"},
    {"question": "What is NLP?", "options":
["Natural Language Processing", "New Logic
Programming", "Network Layer Protocol", "Next
Level Prediction"], "answer": "Natural Language
Processing"},
   ],
  };

  late List<Map<String, dynamic>> questions;

  @override
  void initState() {
   super.initState();
   questions = questionBank[widget.category] ?? [];
   timer = Timer.periodic(const Duration(seconds:
1), (timer) {
    if (timeLeft == 0) {
     nextQuestion();
    } else {
     setState(() {
      timeLeft--;
     });
    }
   });
  }

  void nextQuestion() {
   if (currentQuestion < questions.length - 1) {
    setState(() {
     currentQuestion++;
     timeLeft = 20;
     answered = false;
    });
   } else {
    Navigator.pushReplacement(
     context,
     MaterialPageRoute(
      builder: (_) => QuizSummaryScreen(score: score,
total: questions.length, userResponses: userResponses),
     ),
    );
   }
  }

  void selectAnswer(String selected) {
   if (!answered) {
    bool isCorrect = selected ==
questions[currentQuestion]['answer'];
    if (isCorrect) score += 1;

    userResponses.add({
     "question": questions[currentQuestion]['question'],
     "selected": selected,
     "correct": questions[currentQuestion]['answer'],
     "isCorrect": isCorrect,
    });

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

  @override
  void dispose() {
   timer.cancel();
   super.dispose();
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("${widget.category}
Quiz")),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.center,
          children: [
            Text("Time Left: $timeLeft seconds", style:
const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold)),
            const SizedBox(height: 20),
            Text(questions[currentQuestion]['question'],
style: const TextStyle(fontSize: 22, fontWeight:
FontWeight.bold)),

...questions[currentQuestion]['options'].map((option
) => Padding(
              padding: const EdgeInsets.symmetric(vertical:
8.0),
              child: ElevatedButton(
                onPressed: () => selectAnswer(option),
                child: Text(option),
              ),
            )),
            if (answered)
              ElevatedButton(
                onPressed: nextQuestion,
                child: const Text("Next Question"),
              ),
          ],
        ),
      ),
    );
  }
}
        }
```

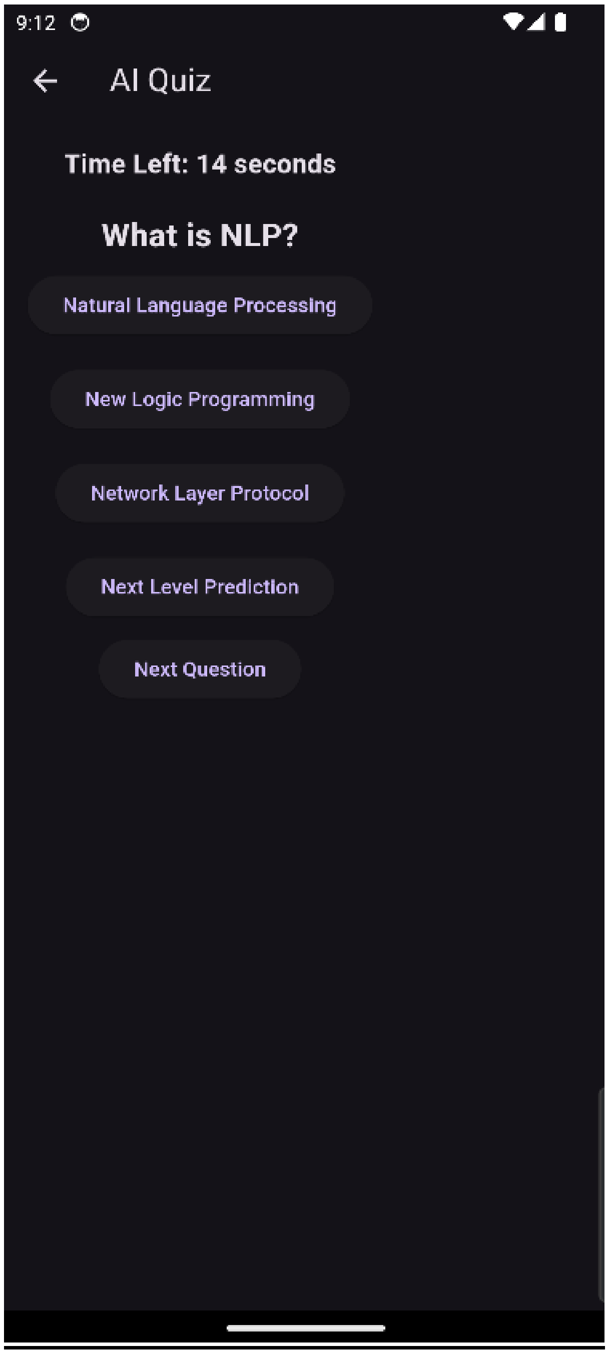
OUTPUT: -


9:12

AI Quiz

Time Left: 15 seconds

Who is the father of AI?


9:12

AI Quiz

Time Left: 14 seconds

What is NLP?