import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:intl/intl.dart';

import 'package:share/share.dart';

class Admin {

  String name;

  String role;

  String password;

  bool hideUserPage;

  bool hideStudentViewPage;

  Admin({

    required this.name,

    required this.role,

    required this.password,

    required this.hideUserPage,

    required this.hideStudentViewPage,

  });

}

class Faculty {

  String name;

  String role;

  String details;

  // New property for username

  String password;

  bool loggedIn;

  List<Notice> notices;

  Faculty(

      {required this.name,

      required this.role,

      required this.details,

      required this.password,

      required this.notices,

      this.loggedIn = false});

}

class Notice {

  String title;

  String description;

  bool isHidden;

  bool hideOnStudentView;

  bool isUnhideByAdmin;

  DateTime timestamp;

  bool isUnhideByFaculty;

  Notice(

      {required this.title,

      this.hideOnStudentView = false,

      this.isUnhideByAdmin = false,

      this.isUnhideByFaculty = false,

      required this.timestamp,

      required this.description,

      this.isHidden = false});

}

class NoticeBoardApp extends StatefulWidget {

  @override

  \_NoticeBoardAppState createState() => \_NoticeBoardAppState();

}

List<Notice> unhiddenNotices = [];

class \_NoticeBoardAppState extends State<NoticeBoardApp> {

  Admin admin = Admin(

    name: 'Ahmad\_hassan',

    role: 'Admin',

    password: '047',

    hideUserPage: false,

    hideStudentViewPage: false, // Add this line

  );

  List<Faculty> faculties = [];

  TextEditingController \_nameController = TextEditingController();

  TextEditingController \_roleController = TextEditingController();

  TextEditingController \_detailsController = TextEditingController();

  TextEditingController \_adminNameController = TextEditingController();

  TextEditingController \_passwordController = TextEditingController();

  TextEditingController \_facultyLoginController = TextEditingController();

  void \_addFaculty(String password) {

    String name = \_nameController.text;

    String role = \_roleController.text;

    String details = \_detailsController.text;

    setState(() {

      faculties.add(Faculty(

          name: name, role: role, details: details, notices: [], password: ''));

    });

    \_nameController.clear();

    \_roleController.clear();

    \_detailsController.clear();

  }

  void \_removeFaculty(Faculty faculty) {

    setState(() {

      faculties.remove(faculty);

    });

  }

  void \_loginAdmin() {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return Dialog(

          backgroundColor: Colors

              .transparent, // Set the dialog background color to transparent

          child: Container(

            decoration: BoxDecoration(

              gradient: LinearGradient(

                colors: [

                  Color.fromARGB(255, 181, 253, 255)

                      .withOpacity(0.9), // Light smoke gradient color

                  Color.fromARGB(206, 216, 210, 255)

                      .withOpacity(0.5), // Light smoke gradient color

                ],

                begin: Alignment.topCenter,

                end: Alignment.bottomCenter,

              ),

            ),

            child: AlertDialog(

              title: const Text('Admin Login'),

              content: Column(

                crossAxisAlignment: CrossAxisAlignment.start,

                mainAxisSize: MainAxisSize.min,

                children: <Widget>[

                  TextField(

                    controller: \_adminNameController,

                    decoration: const InputDecoration(

                      labelText: 'Admin Name',

                    ),

                  ),

                  TextField(

                    controller: \_passwordController,

                    obscureText: true,

                    decoration: const InputDecoration(

                      labelText: 'Password',

                    ),

                  ),

                ],

              ),

              actions: <Widget>[

                TextButton(

                  child: const Text('Cancel'),

                  onPressed: () {

                    Navigator.of(context).pop();

                  },

                ),

                TextButton(

                  child: const Text('Login'),

                  onPressed: () {

                    String adminName = \_adminNameController.text;

                    String password = \_passwordController.text;

                    if (adminName == admin.name && password == admin.password) {

                      Navigator.of(context).pop();

                      \_showAdminOptions();

                    } else {

                      showDialog(

                        context: context,

                        builder: (BuildContext context) {

                          return AlertDialog(

                            title: const Text('Login Failed'),

                            content:

                                const Text('Invalid admin name or password.'),

                            actions: <Widget>[

                              TextButton(

                                child: const Text('OK'),

                                onPressed: () {

                                  Navigator.of(context).pop();

                                },

                              ),

                            ],

                          );

                        },

                      );

                    }

                  },

                ),

              ],

            ),

          ),

        );

      },

    );

  }

  void \_loginFaculty(Faculty faculty) {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Faculty Login'),

          content: Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            mainAxisSize: MainAxisSize.min,

            children: <Widget>[

              Text(

                'Faculty: ${faculty.name}',

                style: const TextStyle(

                  fontWeight: FontWeight.bold,

                ),

              ),

              TextField(

                controller: \_facultyLoginController,

                obscureText: true,

                decoration: const InputDecoration(

                  labelText: 'Password',

                ),

              ),

            ],

          ),

          actions: <Widget>[

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

            TextButton(

              child: const Text('Login'),

              onPressed: () {

                String password = \_facultyLoginController.text;

                if (password == \_passwordController.text) {

                  // Compare with the value from \_passwordController

                  Navigator.of(context).pop();

                  \_showFacultyEditPage(faculty);

                } else {

                  showDialog(

                    context: context,

                    builder: (BuildContext context) {

                      return AlertDialog(

                        title: const Text('Login Failed'),

                        content: const Text('Invalid password.'),

                        actions: <Widget>[

                          TextButton(

                            child: const Text('OK'),

                            onPressed: () {

                              Navigator.of(context).pop();

                            },

                          ),

                        ],

                      );

                    },

                  );

                }

              },

            ),

          ],

        );

      },

    );

  }

  void \_showAdminOptions() {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Admin Options'),

          content: Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            mainAxisSize: MainAxisSize.min,

            children: <Widget>[

              ListTile(

                leading: const Icon(Icons.add),

                title: const Text('Add Faculty'),

                onTap: () {

                  Navigator.of(context).pop();

                  \_showAddFacultyDialog();

                },

              ), /////////////////////////////

              ////////////////////

              ListTile(

                leading: const Icon(Icons.delete),

                title: const Text('Remove Faculty'),

                onTap: () {

                  Navigator.of(context).pop();

                  \_showRemoveFacultyDialog();

                },

              ),

              ListTile(

                leading: const Icon(Icons.star),

                title: const Text('Pro Page'),

                onTap: () {

                  List<Notice> mergedNotices = [];

                  // Retrieve the notices from each faculty and merge them into the mergedNotices list

                  for (Faculty faculty in faculties) {

                    mergedNotices.addAll(faculty.notices);

                  }

                  Navigator.of(context).pop();

                  Navigator.push(

                    context,

                    MaterialPageRoute(

                      builder: (context) => ProPage(

                        notices: mergedNotices,

                      ),

                    ),

                  );

                },

              ),

              ListTile(

                leading: const Icon(Icons.edit),

                title: const Text('Edit Admin Name'),

                onTap: () {

                  Navigator.of(context).pop();

                  \_showEditAdminNameDialog();

                },

              ),

              ListTile(

                leading: Icon(

                  admin.hideStudentViewPage

                      ? Icons.visibility

                      : Icons.visibility\_off,

                ),

                title: Text(

                  admin.hideStudentViewPage

                      ? 'View Student Page'

                      : 'Hide Student Page',

                ),

                onTap: () {

                  Navigator.of(context).pop();

                  setState(() {

                    admin.hideStudentViewPage = !admin.hideStudentViewPage;

                  });

                },

              ),

            ],

          ),

          actions: <Widget>[

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    );

  }

  void \_showAddFacultyDialog() {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Add Faculty'),

          content: Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            mainAxisSize: MainAxisSize.min,

            children: <Widget>[

              TextField(

                controller: \_nameController,

                decoration: const InputDecoration(

                  labelText: 'Name',

                ),

              ),

              TextField(

                controller: \_roleController,

                decoration: const InputDecoration(

                  labelText: 'Role',

                ),

              ),

              TextField(

                controller: \_detailsController,

                decoration: const InputDecoration(

                  labelText: 'Details',

                ),

              ),

              TextField(

                onChanged: (value) {

                  \_passwordController.text =

                      value; // Update the password value in the controller

                },

                obscureText: true,

                decoration: const InputDecoration(

                  labelText: 'Password',

                ),

              ),

            ],

          ),

          actions: <Widget>[

            GestureDetector(

              onTap: () {

                \_shareFacultyDetails();

              },

              child: IconButton(

                icon: const Icon(Icons.share),

                color: Colors.black,

                onPressed: null,

              ),

            ),

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

            TextButton(

              child: const Text('Add'),

              onPressed: () {

                Navigator.of(context).pop();

                \_addFaculty(\_passwordController.text);

              },

            ),

          ],

        );

      },

    );

  }

///////////////////////

  void \_shareFacultyDetails() {

    String name = \_nameController.text;

    String role = \_roleController.text;

    String password = \_passwordController.text;

    String message = 'Name: $name\nRole: $role\nPassword: $password';

    Share.share(message);

  }

  void \_showRemoveFacultyDialog() {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Remove Faculty'),

          content: SingleChildScrollView(

            child: Column(

              crossAxisAlignment: CrossAxisAlignment.start,

              mainAxisSize: MainAxisSize.min,

              children: faculties.map((faculty) {

                return ListTile(

                  title: Text(faculty.name),

                  subtitle: Text(faculty.role),

                  onTap: () {

                    Navigator.of(context).pop();

                    \_removeFaculty(faculty);

                  },

                );

              }).toList(),

            ),

          ),

          actions: <Widget>[

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    );

  }

  void \_showEditAdminNameDialog() {

    TextEditingController \_adminNameController = TextEditingController();

    TextEditingController \_passwordController = TextEditingController();

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Edit Admin Details'),

          content: Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            mainAxisSize: MainAxisSize.min,

            children: <Widget>[

              TextField(

                controller: \_adminNameController,

                decoration: const InputDecoration(

                  labelText: 'New Admin Name',

                ),

              ),

              TextField(

                controller: \_passwordController,

                obscureText: true,

                decoration: const InputDecoration(

                  labelText: 'New Password',

                ),

              ),

            ],

          ),

          actions: <Widget>[

            IconButton(

              icon: const Icon(Icons.share),

              onPressed: () {

                String adminName = \_adminNameController.text;

                String password = \_passwordController.text;

                String textToShare =

                    'Admin Name: $adminName\nPassword: $password';

                Share.share(textToShare);

              },

            ),

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

            TextButton(

              child: const Text('Save'),

              onPressed: () {

                setState(() {

                  admin.name = \_adminNameController.text;

                  admin.password = \_passwordController.text;

                });

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    );

  }

///////////////////////////////

  void \_showFacultyEditPage(Faculty faculty) {

    TextEditingController \_titleController = TextEditingController();

    TextEditingController \_descriptionController = TextEditingController();

    List<Notice> filteredNotices = []; // Added variable for filtered notices

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return Scaffold(

          appBar: AppBar(

            title: Text(

              'Notice Board',

              textAlign: TextAlign.center,

              style: TextStyle(

                fontWeight: FontWeight.bold,

                foreground: Paint()

                  ..shader = LinearGradient(

                    colors: [

                      Color.fromRGBO(16, 17, 72, 1), // Dark blue

                      Color.fromRGBO(219, 225, 160, 1), // Light orange

                    ],

                  ).createShader(

                    Rect.fromLTWH(0.0, 0.0, 200.0, 70.0),

                  ),

              ),

            ),

            centerTitle: true,

            flexibleSpace: Container(

              decoration: BoxDecoration(

                gradient: LinearGradient(

                  colors: [

                    Color.fromRGBO(16, 17, 72, 1), // Dark blue

                    Color.fromRGBO(219, 225, 160, 1), // Light orange

                  ],

                ),

              ),

            ),

          ),

          body: Column(

            children: [

              Row(

                mainAxisAlignment: MainAxisAlignment.end,

                children: [

                  IconButton(

                    icon: const Icon(Icons.add),

                    onPressed: () {

                      showDialog(

                        context: context,

                        builder: (BuildContext context) {

                          return AlertDialog(

                            title: const Text('Add Notice'),

                            content: Column(

                              crossAxisAlignment: CrossAxisAlignment.start,

                              mainAxisSize: MainAxisSize.min,

                              children: <Widget>[

                                TextField(

                                  controller: \_titleController,

                                  decoration: const InputDecoration(

                                    labelText: 'Title',

                                  ),

                                ),

                                TextField(

                                  controller: \_descriptionController,

                                  decoration: const InputDecoration(

                                    labelText: 'Description',

                                  ),

                                ),

                              ],

                            ),

                            actions: <Widget>[

                              TextButton(

                                child: const Text('Cancel'),

                                onPressed: () {

                                  Navigator.of(context).pop();

                                },

                              ),

                              TextButton(

                                child: const Text('Add'),

                                onPressed: () {

                                  String title = \_titleController.text;

                                  String description =

                                      \_descriptionController.text;

                                  Notice newNotice = Notice(

                                    title: title,

                                    description: description,

                                    timestamp: DateTime

                                        .now(), // Provide a timestamp value

                                  );

                                  setState(() {

                                    faculty.notices.add(newNotice);

                                  });

                                  Navigator.of(context).pop();

                                },

                              ),

                            ],

                          );

                        },

                      );

                    },

                  ),

                  IconButton(

                    icon: const Icon(Icons.search, color: Colors.yellow),

                    onPressed: () {

                      showDialog(

                        context: context,

                        builder: (BuildContext context) {

                          return AlertDialog(

                            title: const Text('Search Notice',

                                style: TextStyle(color: Colors.blue)),

                            backgroundColor: Colors.white,

                            shape: RoundedRectangleBorder(

                              borderRadius: BorderRadius.circular(8.0),

                            ),

                            content: Column(

                              crossAxisAlignment: CrossAxisAlignment.start,

                              mainAxisSize: MainAxisSize.min,

                              children: <Widget>[

                                TextField(

                                  controller: \_titleController,

                                  decoration: const InputDecoration(

                                    labelText: 'Title',

                                  ),

                                ),

                              ],

                            ),

                            actions: <Widget>[

                              TextButton(

                                child: const Text('Cancel',

                                    style: TextStyle(color: Colors.red)),

                                onPressed: () {

                                  Navigator.of(context).pop();

                                },

                              ),

                              TextButton(

                                child: const Text('Search',

                                    style: TextStyle(color: Colors.green)),

                                onPressed: () {

                                  String searchTitle = \_titleController.text;

                                  // Filter the notices based on the search title

                                  filteredNotices = faculty.notices

                                      .where((notice) =>

                                          notice.title.contains(searchTitle))

                                      .toList();

                                  Navigator.of(context).pop();

                                  // Rebuild the UI with filtered notices

                                  setState(() {});

                                },

                              ),

                            ],

                          );

                        },

                      );

                    },

                  ),

                ],

              ),

              Expanded(

                child: Builder(

                  builder: (BuildContext context) {

                    // Use Builder to access the context

                    return ListView.builder(

                      itemCount: filteredNotices.isNotEmpty

                          ? filteredNotices.length

                          : faculty.notices.length,

                      itemBuilder: (BuildContext context, int index) {

                        Notice notice = filteredNotices.isNotEmpty

                            ? filteredNotices[index]

                            : faculty.notices[index];

                        return ListTile(

                          title: Text(notice.title),

                          subtitle: Text(notice.description),

                          trailing: PopupMenuButton(

                            itemBuilder: (BuildContext context) {

                              return [

                                PopupMenuItem(

                                  child: ListTile(

                                    leading: const Icon(Icons.edit),

                                    title: const Text('Update'),

                                    onTap: () {

                                      showDialog(

                                        context: context,

                                        builder: (BuildContext context) {

                                          TextEditingController

                                              \_updateTitleController =

                                              TextEditingController(

                                                  text: notice.title);

                                          TextEditingController

                                              \_updateDescriptionController =

                                              TextEditingController(

                                                  text: notice.description);

                                          return AlertDialog(

                                            title: const Text('Update Notice'),

                                            content: Column(

                                              crossAxisAlignment:

                                                  CrossAxisAlignment.start,

                                              mainAxisSize: MainAxisSize.min,

                                              children: <Widget>[

                                                TextField(

                                                  controller:

                                                      \_updateTitleController,

                                                  decoration:

                                                      const InputDecoration(

                                                    labelText: 'Title',

                                                  ),

                                                ),

                                                TextField(

                                                  controller:

                                                      \_updateDescriptionController,

                                                  decoration:

                                                      const InputDecoration(

                                                    labelText: 'Description',

                                                  ),

                                                ),

                                              ],

                                            ),

                                            actions: <Widget>[

                                              TextButton(

                                                child: const Text('Cancel'),

                                                onPressed: () {

                                                  Navigator.of(context).pop();

                                                },

                                              ),

                                              TextButton(

                                                child: const Text('Update'),

                                                onPressed: () {

                                                  String updatedTitle =

                                                      \_updateTitleController

                                                          .text;

                                                  String updatedDescription =

                                                      \_updateDescriptionController

                                                          .text;

                                                  setState(() {

                                                    notice.title = updatedTitle;

                                                    notice.description =

                                                        updatedDescription;

                                                  });

                                                  Navigator.of(context).pop();

                                                },

                                              ),

                                            ],

                                          );

                                        },

                                      );

                                    },

                                  ),

                                ),

                                PopupMenuItem(

                                  child: ListTile(

                                    leading: Icon(

                                      Icons.print,

                                      color: Colors.black,

                                    ),

                                    title: const Text(

                                      'Print',

                                      style: TextStyle(

                                        color: Colors.black,

                                      ),

                                    ),

                                    onTap: () {

                                      \_shareNotice(notice);

                                    },

                                  ),

                                ),

                                PopupMenuItem(

                                  child: ListTile(

                                    leading: Icon(

                                      Icons.delete,

                                      color: Colors.black,

                                    ),

                                    title: const Text(

                                      'Delete',

                                      style: TextStyle(

                                        color: Colors.black,

                                      ),

                                    ),

                                    onTap: () {

                                      showDialog(

                                        context: context,

                                        builder: (BuildContext context) {

                                          return AlertDialog(

                                            title: const Text('Delete Notice'),

                                            content: const Text(

                                                'Are you sure you want to delete this notice?'),

                                            actions: <Widget>[

                                              TextButton(

                                                child: const Text('Cancel'),

                                                onPressed: () {

                                                  Navigator.of(context).pop();

                                                },

                                              ),

                                              TextButton(

                                                child: const Text('Delete'),

                                                onPressed: () {

                                                  setState(() {

                                                    faculty.notices

                                                        .remove(notice);

                                                    if (filteredNotices

                                                        .isNotEmpty) {

                                                      filteredNotices

                                                          .remove(notice);

                                                    }

                                                  });

                                                  Navigator.of(context).pop();

                                                },

                                              ),

                                            ],

                                            shape: RoundedRectangleBorder(

                                              borderRadius:

                                                  BorderRadius.circular(16),

                                            ),

                                            backgroundColor: Colors.white,

                                          );

                                        },

                                      );

                                    },

                                  ),

                                ),

                                // Only show "Unhide" option if the notice is hidden on student view

                                PopupMenuItem(

                                  child: ListTile(

                                    leading: Icon(

                                      notice.isUnhideByAdmin

                                          ? Icons.visibility

                                          : Icons.visibility\_off,

                                      color: notice.isUnhideByAdmin

                                          ? Colors.green

                                          : Colors.red,

                                    ),

                                    title: Text(

                                      notice.isUnhideByAdmin

                                          ? 'Hide'

                                          : 'Unhide',

                                      style: TextStyle(

                                        color: notice.isUnhideByAdmin

                                            ? Colors.green

                                            : Colors.red,

                                      ),

                                    ),

                                    onTap: () {

                                      setState(() {

                                        notice.isUnhideByAdmin =

                                            !notice.isUnhideByAdmin;

                                      });

                                    },

                                  ),

                                ),

                              ];

                            },

                          ),

                        );

                      },

                    );

                  },

                ),

              ),

              ////////

            ],

          ),

          floatingActionButton: FloatingActionButton(

            onPressed: () {

              // ...

              Navigator.of(context).push(

                MaterialPageRoute(

                  builder: (context) => ProPage(notices: faculty.notices),

                ),

              );

            },

            child: const Icon(Icons.print),

          ),

        );

      },

    );

  }

//////////////////////////////////////////////////

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

          'Notifications ',

          textAlign: TextAlign.center,

          style: TextStyle(

            fontWeight: FontWeight.bold,

            color: Colors.white,

            shadows: [

              Shadow(

                blurRadius: 2.0,

                color: Colors.black,

                offset: Offset(2.0, 2.0),

              ),

            ],

          ),

        ),

        centerTitle: true,

        flexibleSpace: Container(

          decoration: BoxDecoration(

            gradient: LinearGradient(

              colors: [

                Color.fromARGB(138, 192, 25, 210),

                Color.fromARGB(255, 250, 184, 208),

              ],

              begin: Alignment.topLeft,

              end: Alignment.bottomRight,

            ),

          ),

        ),

        elevation: 0, // Remove elevation of the app bar

      ),

      body: Container(

        decoration: BoxDecoration(

          gradient: LinearGradient(

            colors: [

              Colors.blueGrey.shade100,

              Colors.blueGrey.shade50,

            ],

            begin: Alignment.topCenter,

            end: Alignment.bottomCenter,

          ),

        ),

        child: Center(

          child: Card(

            elevation: 5,

            shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(10.0),

            ),

            child: Container(

              width: 300,

              height: 200,

              decoration: BoxDecoration(

                gradient: LinearGradient(

                  colors: [

                    Color.fromARGB(174, 5, 5, 5),

                    Color.fromARGB(223, 183, 183, 183),

                  ],

                  begin: Alignment.topLeft,

                  end: Alignment.bottomRight,

                ),

                borderRadius: BorderRadius.circular(10.0),

              ),

              child: Column(

                mainAxisAlignment: MainAxisAlignment.center,

                children: [

                  ListTile(

                    leading: Icon(

                      Icons.star,

                      color: Colors.yellow,

                    ),

                    title: Text(

                      admin.name,

                      style: TextStyle(

                        color: Colors.white,

                        fontSize: 24,

                      ),

                    ),

                    onTap: \_loginAdmin,

                  ),

                  ListTile(

                    leading: Icon(

                      Icons.person,

                      color: Colors.white,

                    ),

                    title: Text(

                      'SUB ADMIN',

                      style: TextStyle(

                        color: Colors.white,

                        fontWeight: FontWeight.bold,

                      ),

                    ),

                    onTap: () {

                      showDialog(

                        context: context,

                        builder: (BuildContext context) {

                          return AlertDialog(

                            title: Text(

                              'SUB ADMIN',

                              style: TextStyle(

                                color: Colors.black,

                              ),

                            ),

                            content: SingleChildScrollView(

                              child: Column(

                                crossAxisAlignment: CrossAxisAlignment.start,

                                mainAxisSize: MainAxisSize.min,

                                children: faculties.map((faculty) {

                                  return ListTile(

                                    title: Text(faculty.name),

                                    subtitle: Text(faculty.role),

                                    trailing: IconButton(

                                      icon: Icon(Icons.login),

                                      color: Color.fromARGB(255, 135, 251, 166),

                                      onPressed: () {

                                        Navigator.of(context).pop();

                                        \_loginFaculty(faculty);

                                      },

                                    ),

                                  );

                                }).toList(),

                              ),

                            ),

                            actions: <Widget>[

                              TextButton(

                                child: Text('Close'),

                                onPressed: () {

                                  Navigator.of(context).pop();

                                },

                              ),

                            ],

                          );

                        },

                      );

                    },

                  ),

                  ListTile(

                    leading: Icon(

                      Icons.school\_sharp,

                      color: Colors.white,

                    ),

                    title: Text(

                      'USERS',

                      style: TextStyle(

                        color: Colors.white,

                        fontWeight: FontWeight.bold,

                      ),

                    ),

                    onTap: () {

                      List<Notice> mergedNotices = [];

                      // Retrieve the notices from each faculty and merge them into the mergedNotices list

                      for (Faculty faculty in faculties) {

                        // Check if the notices are unhidden by the admin

                        List<Notice> unhiddenNotices = faculty.notices

                            .where((notice) => !notice.isUnhideByAdmin)

                            .toList();

                        mergedNotices.addAll(unhiddenNotices);

                      }

                      Navigator.push(

                        context,

                        MaterialPageRoute(

                          builder: (context) => StudentViewPage(

                            notices: mergedNotices,

                          ),

                        ),

                      );

                    },

                  ),

                ],

              ),

            ),

          ),

        ),

      ),

    );

  }

  void \_showChangeFacultyPasswordDialog() {}

  void \_shareNotice(Notice notice) {

    Share.share('${notice.title}\n${notice.description}');

  }

}

class StudentViewPage extends StatefulWidget {

  final List<Notice> notices;

  StudentViewPage({required this.notices});

  @override

  \_StudentViewPageState createState() => \_StudentViewPageState();

}

class \_StudentViewPageState extends State<StudentViewPage> {

  Set<int> viewedNotices = Set<int>();

  void \_toggleNoticeView(int index) {

    setState(() {

      if (viewedNotices.contains(index)) {

        viewedNotices.remove(index);

      } else {

        viewedNotices.add(index);

      }

    });

  }

  @override

  Widget build(BuildContext context) {

    // Set status bar and system navigation bar styles

    SystemChrome.setSystemUIOverlayStyle(

      SystemUiOverlayStyle(

        statusBarColor: Colors.transparent, // Make status bar transparent

        systemNavigationBarColor:

            Colors.black, // Set system navigation bar color

        systemNavigationBarIconBrightness:

            Brightness.light, // Set system navigation bar icon color

      ),

    );

    return Scaffold(

      appBar: AppBar(

        title: Text(

          'USER VIEW MODE',

          textAlign: TextAlign.center,

          style: TextStyle(

            fontWeight: FontWeight.bold,

            foreground: Paint()

              ..shader = LinearGradient(

                colors: [

                  Color.fromRGBO(26, 16, 25, 0.608), // Dark blue

                  Color.fromRGBO(242, 82, 186, 0.435), // Light orange

                ],

              ).createShader(

                Rect.fromLTWH(0.0, 0.0, 200.0, 70.0),

              ),

          ),

        ),

        centerTitle: true,

        flexibleSpace: Container(

          decoration: BoxDecoration(

            gradient: LinearGradient(

              colors: [

                Color.fromRGBO(175, 255, 194, 0.812), // Dark blue

                Color.fromRGBO(255, 251, 143, 0.788), // Light orange

              ],

            ),

          ),

        ),

      ),

      body: ListView.builder(

        reverse: true, // Reverse the list to show latest messages at the bottom

        itemCount: widget.notices.length,

        itemBuilder: (BuildContext context, int index) {

          Notice notice = widget.notices[index];

          bool isViewed = viewedNotices.contains(index);

          return Card(

            elevation: 0, // Remove the elevation from the card

            shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(10),

            ),

            margin: EdgeInsets.symmetric(horizontal: 16, vertical: 8),

            child: InkWell(

              borderRadius: BorderRadius.circular(10),

              onTap: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(

                    builder: (context) => NoticeDetailsPage(notice: notice),

                  ),

                ).then((\_) {

                  \_toggleNoticeView(index);

                });

              },

              child: Container(

                decoration: BoxDecoration(

                  color: isViewed

                      ? Colors.grey.shade200

                      : Colors.white, // Change color based on viewed status

                  borderRadius: BorderRadius.circular(10),

                ),

                child: ListTile(

                  title: Text(

                    notice.title,

                    style: TextStyle(fontWeight: FontWeight.bold),

                  ),

                  subtitle: Text(

                    notice.description,

                    style: TextStyle(color: Colors.grey),

                  ),

                  trailing: Column(

                    mainAxisAlignment: MainAxisAlignment.center,

                    children: [

                      Text(

                        DateFormat('hh:mm a')

                            .format(notice.timestamp.toLocal()),

                        // Show the timestamp

                        style: TextStyle(color: Colors.grey, fontSize: 12),

                      ),

                      SizedBox(height: 4),

                      Icon(

                        isViewed ? Icons.check : Icons.check\_circle\_outline,

                        color: isViewed ? Colors.grey : Colors.green,

                        size: 16,

                      ),

                    ],

                  ),

                ),

              ),

            ),

          );

        },

      ),

    );

  }

}

class ProPage extends StatefulWidget {

  final List<Notice> notices;

  ProPage({required this.notices});

  @override

  \_ProPageState createState() => \_ProPageState();

}

class \_ProPageState extends State<ProPage> {

  List<Notice> filteredNotices = [];

  bool showSuccessMessage = false;

  void \_updateNotice(

      Notice notice, String updatedTitle, String updatedDescription) {

    setState(() {

      notice.title = updatedTitle;

      notice.description = updatedDescription;

      showSuccessMessage = true;

    });

    Future.delayed(const Duration(seconds: 2), () {

      setState(() {

        showSuccessMessage = false;

      });

    });

  }

  void \_deleteNotice(Notice notice) {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Delete Notice'),

          content: const Text('Are you sure you want to delete this notice?'),

          actions: <Widget>[

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

            TextButton(

              child: const Text('Delete'),

              onPressed: () {

                setState(() {

                  widget.notices.remove(notice);

                });

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    );

  }

  void \_viewNoticeDetails(Notice notice) {

    Navigator.push(

      context,

      MaterialPageRoute(

        builder: (context) => NoticeDetailsPage(notice: notice),

      ),

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.white,

      appBar: AppBar(

        title: Text(

          'Updating page ',

          textAlign: TextAlign.center,

          style: TextStyle(

            fontWeight: FontWeight.bold,

            foreground: Paint()

              ..shader = LinearGradient(

                colors: [

                  Color.fromRGBO(0, 0, 0, 1), // Dark blue

                  Color.fromRGBO(16, 27, 25, 0.631), // Light orange

                ],

              ).createShader(

                Rect.fromLTWH(0.0, 0.0, 200.0, 70.0),

              ),

          ),

        ),

        centerTitle: true,

        flexibleSpace: Container(

          decoration: BoxDecoration(

            gradient: LinearGradient(

              colors: [

                Color.fromRGBO(160, 250, 186, 0.875), // Dark blue

                Color.fromRGBO(232, 242, 144, 0.751), // Light orange

              ],

            ),

          ),

        ),

      ),

      body: ListView.builder(

        itemCount: widget.notices.length,

        itemBuilder: (BuildContext context, int index) {

          Notice notice = widget.notices[index];

          return GestureDetector(

            onTap: () {

              \_viewNoticeDetails(notice);

            },

            child: Container(

              padding: EdgeInsets.all(16.0),

              decoration: BoxDecoration(

                border: Border(

                  bottom: BorderSide(color: Colors.grey.shade300),

                ),

              ),

              child: Row(

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  CircleAvatar(

                    backgroundColor: Colors.grey,

                    radius: 24.0,

                    child: Icon(

                      Icons.person,

                      color: Colors.white,

                    ),

                  ),

                  SizedBox(width: 16.0),

                  Expanded(

                    child: Column(

                      crossAxisAlignment: CrossAxisAlignment.start,

                      children: [

                        Row(

                          children: [

                            Text(

                              notice.title,

                              style: TextStyle(

                                fontWeight: FontWeight.bold,

                              ),

                            ),

                            if (showSuccessMessage)

                              Icon(

                                Icons.check\_circle,

                                color: Colors.green,

                                size: 20.0,

                              ),

                          ],

                        ),

                        SizedBox(height: 8.0),

                        Text(

                          notice.description,

                          style: TextStyle(

                            color: Colors.grey.shade700,

                          ),

                        ),

                      ],

                    ),

                  ),

                  SizedBox(width: 16.0),

                  PopupMenuButton(

                    icon: Icon(Icons.more\_vert),

                    itemBuilder: (BuildContext context) {

                      return [

                        PopupMenuItem(

                          child: ListTile(

                            leading: Icon(Icons.edit),

                            title: Text('Update'),

                            onTap: () {

                              showDialog(

                                context: context,

                                builder: (BuildContext context) {

                                  TextEditingController \_updateTitleController =

                                      TextEditingController(text: notice.title);

                                  TextEditingController

                                      \_updateDescriptionController =

                                      TextEditingController(

                                          text: notice.description);

                                  return AlertDialog(

                                    title: Text('Update Notice'),

                                    content: Column(

                                      crossAxisAlignment:

                                          CrossAxisAlignment.start,

                                      mainAxisSize: MainAxisSize.min,

                                      children: <Widget>[

                                        TextField(

                                          controller: \_updateTitleController,

                                          decoration: InputDecoration(

                                            labelText: 'Title',

                                          ),

                                        ),

                                        TextField(

                                          controller:

                                              \_updateDescriptionController,

                                          decoration: InputDecoration(

                                            labelText: 'Description',

                                          ),

                                        ),

                                      ],

                                    ),

                                    actions: <Widget>[

                                      TextButton(

                                        child: Text('Cancel'),

                                        onPressed: () {

                                          Navigator.of(context).pop();

                                        },

                                      ),

                                      TextButton(

                                        child: Text('Update'),

                                        onPressed: () {

                                          String updatedTitle =

                                              \_updateTitleController.text;

                                          String updatedDescription =

                                              \_updateDescriptionController.text;

                                          \_updateNotice(

                                            notice,

                                            updatedTitle,

                                            updatedDescription,

                                          );

                                          Navigator.of(context).pop();

                                        },

                                      ),

                                    ],

                                  );

                                },

                              );

                            },

                          ),

                        ),

                        PopupMenuItem(

                          child: ListTile(

                            leading: Icon(Icons.delete),

                            title: Text('Delete'),

                            onTap: () {

                              \_deleteNotice(notice);

                            },

                          ),

                        ),

                      ];

                    },

                  ),

                ],

              ),

            ),

          );

        },

      ),

    );

  }

}

void \_viewNoticeDetails(Notice notice) {

  // Handle view notice details

}

class NoticeDetailsPage extends StatelessWidget {

  final Notice notice;

  const NoticeDetailsPage({required this.notice});

  @override

  Widget build(BuildContext context) {

    // Set status bar and system navigation bar styles

    SystemChrome.setSystemUIOverlayStyle(

      SystemUiOverlayStyle(

        statusBarColor:

            Color.fromARGB(166, 105, 195, 196), // Set status bar color

        systemNavigationBarColor: Color.fromARGB(

            204, 130, 224, 187), // Set system navigation bar color

        systemNavigationBarIconBrightness:

            Brightness.light, // Set system navigation bar icon color

      ),

    );

    return Scaffold(

      appBar: AppBar(

        title: Text('Updates'),

        backgroundColor: Colors.black, // Set app bar color

        elevation: 0, // Remove app bar shadow

        actions: [

          IconButton(

            icon: Icon(

              Icons.share,

              color: Colors.white, // Set icon color

            ),

            onPressed: () {

              // Implement the action for the share icon

              \_shareNotice(context);

            },

          ),

        ],

      ),

      body: Container(

        padding: EdgeInsets.symmetric(horizontal: 16, vertical: 8),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.stretch,

          children: [

            Container(

              padding: EdgeInsets.all(8),

              decoration: BoxDecoration(

                color: Colors.white,

                borderRadius: BorderRadius.circular(8),

              ),

              child: Text(

                notice.title,

                style: TextStyle(

                  fontWeight: FontWeight.bold,

                  fontSize: 17.0,

                ),

              ),

            ),

            SizedBox(height: 8),

            Container(

              padding: EdgeInsets.all(8),

              decoration: BoxDecoration(

                color: Colors.white,

                borderRadius: BorderRadius.circular(8),

              ),

              child: Text(

                notice.description,

                style: TextStyle(

                  color: Colors.black,

                ),

              ),

            ),

          ],

        ),

      ),

    );

  }

  void \_shareNotice(BuildContext context) {

    final String message = '${notice.title}\n\n${notice.description}';

    // Share notice message using Share widget

    Share.share(message);

  }

}

List<Notice> mergeNotices(List<Faculty> faculties) {

  List<Notice> mergedNotices = [];

  for (Faculty faculty in faculties) {

    mergedNotices.addAll(faculty.notices);

  }

  return mergedNotices;

}

void main() {

  runApp(MaterialApp(

    home: NoticeBoardApp(),

    debugShowCheckedModeBanner: false,

  ));

}