

BSP SUMMER-INTERNSHIP PROJECT REPORT

"Effortless Image Sharing for Real-Time Site Incident Reporting" (Kumar Malay,Tanmay Singh)

Introduction

This project aims to improve incident reporting, foster safety, and enhance communication in dynamic work environments. By using technology to make incident reporting easier, "Effortless Image Sharing for Real-Time Site Incident Reporting" ensures that vital information reaches the appropriate stakeholders quickly, minimising delays and improving overall site safety.

Effortless Image Capture: Users can easily capture images of site incidents, eliminating the need for complicated documentation.

Real-Time Communication: The application guarantees real-time communication by sending incident reports directly to the server, making them accessible to relevant stakeholders immediately.

User-Friendly Interface: With a simple and user-friendly interface, the app can be easily used by on-site personnel without extensive training. Detailed

Descriptions: Users can provide detailed descriptions of incidents, facilitating clear understanding and action.

Efficient Incident Management: The central server stores all incident reports, enabling efficient incident management, tracking, and resolution.

Enhanced Safety: Prompt reporting and fast response to incidents contribute to improved safety measures at construction and industrial sites.

Objectives

The primary objective of the "Effortless Image Sharing for Real-Time Site Incident Reporting" project is to develop a user-friendly and efficient mobile application that facilitates seamless incident reporting and communication in real-time within construction and industrial work environments. The project aims to achieve the following key objectives:

Simplify Incident Reporting: Create a user-friendly interface that allows on-site personnel to effortlessly capture images of incidents, damages, or issues using their smartphones. This simplification of the reporting process eliminates the need for complex paperwork or documentation.

Enhance Communication: Establish a real-time communication channel between users and a central server. Ensure that incident reports, including images and descriptions, are instantly transmitted to the server for immediate access by authorised stakeholders.

Detailed Incident Documentation: Enable users to provide detailed descriptions and context along with the captured images. This documentation ensures that the nature and severity of incidents are well-understood by relevant parties.

Centralized Incident Storage: Implement a central server to store and organise all incident reports. This centralization enhances the efficiency of incident management, tracking, and resolution.

By achieving these objectives, the project aims to revolutionise incident reporting in industrial settings, promoting safety, efficiency, and real-time communication among all stakeholders involved in site incident management.

Methodology

The methodology of the project "Effortless Image Sharing for Real-Time Site Incident Reporting" involves a structured approach to plan, develop, and implement the mobile application and associated server system. Below is an outline of the project's methodology:

1. Project Planning and Analysis:

- **Needs Assessment:** Identify the specific needs and challenges within the target industrial or construction work environments related to incident reporting and communication.

- **User Requirements:** Gather requirements by conducting interviews and surveys with potential users to understand their preferences and expectations.
- **Technical Feasibility:** Evaluate the technical feasibility of the project, considering factors such as available technology stacks, platforms, and resources.

2. System Design:

- **Architecture Design:** Define the system's architecture, including the mobile application, server components, and database structure
- **User Interface (UI) Design:** Create wireframes and design mockups of the mobile application to ensure an intuitive and user-friendly interface.
- **Database Design:** Plan the database schema to efficiently store incident data, user information, and related content.

3. Development:

- **Frontend Development:** Develop the mobile application using a suitable framework (e.g., Flutter) with a focus on responsive design and user experience.
- **Backend Development:** Build the server-side components, including APIs for data transmission, user authentication, and incident management.
- **Database Implementation:** Create the database system based on the design specifications.

4. Testing and Quality Assurance:

- **Unit Testing:** Conduct unit tests to ensure the correctness of individual application and server components.
- **Integration Testing:** Verify the interaction and compatibility between the mobile app and server components.
- **User Acceptance Testing (UAT):** Involve target users in testing the application to identify and address usability issues.
- **Quality Assurance:** Implement best practices for code quality, security, and performance.

5. Deployment:

- **Server Deployment:** Deploy the server system on a suitable hosting platform or cloud infrastructure (NODE JS SERVER, FIREBASE).

6. User Training and Adoption:

- **Training Materials:** Create user guides or tutorials to help users understand how to use the application effectively.
- **Training Sessions:** Conduct training sessions for users and administrators to ensure proper adoption.

7. Data Management:

- **Data Collection:** Enable users to report incidents with images and descriptions, which are sent to and stored on the central server.
- **Data Security:** Implement robust security measures to protect sensitive incident data.

8. Documentation:

Document all aspects of the project, including design specifications, development details, and user guides.

Analysing the Code

- Main File

```
import 'package:alpha/pages/Cameras/CameraScreen.dart';
import 'package:alpha/pages/sqlite/DatabaseHelper.dart';
import 'package:alpha/pages/sqlite/authPage_sql.dart';
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
```

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  cameras = await availableCameras();
  final databaseHelper = DatabaseHelper();
  await databaseHelper.initializeDatabase();
  final prefs = await SharedPreferences.getInstance();
  runApp(const MyApp());
}
```

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});
```

```
  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: AuthPageSql(),
    );
  }
}
```

- User Authentication

```
import 'package:flutter/material.dart';
import 'CustomBottomNavigation.dart';
import 'HomePageNew.dart';
import 'LoginOrRegister.dart';
import 'package:shared_preferences/shared_preferences.dart';
```

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

```
  @override
  State<AuthPageSql> createState() => _AuthPageSqlState();
}
```

```
class _AuthPageSqlState extends State<AuthPageSql> {
```

```

@override
Widget build(BuildContext context) {
  return FutureBuilder<SharedPreferences>(
    future: SharedPreferences.getInstance(),
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return CircularProgressIndicator();
      } else if (snapshot.hasData) {
        final prefs = snapshot.data as SharedPreferences;
        final bool? isLoggedIn = prefs.getBool('isLoggedIn');
        if (isLoggedIn ?? false) {
          // User is logged in, navigate to HomePageNew
          return CustomBottomNaviagtion();
        } else {
          // User is not logged in, show login/register page
          return LoginOrRegisterPageSql();
        }
      } else {
        return const Text('Something went wrong');
      }
    },
  );
}

```

● Log-In and User Registration

```

import 'package:alpha/pages/sqlite/HomePageNew.dart';
import 'package:flutter/material.dart';

```

```

import 'HomePageNew_2.dart';
import 'RegisterPage_sqlite.dart';

```

```

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

```

```

  @override
  State<CustomBottomNaviagtion> createState() =>
    _CustomBottomNaviagtionState();
}

```

```

class _CustomBottomNaviagtionState extends
  State<CustomBottomNaviagtion> {
  final PageController _pageController = PageController();
  int _currentIndex = 0;

```

```

  @override

```

```

void dispose() {
  _pageController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return SafeArea(
    child: Scaffold(
      body: Stack(
        children: [
          _buildPage(0, HomePageNew()), // Index 0: Home Page
          _buildPage(1, HomePageNew2()), // Index 1: Received Page
        ],
      ),
      bottomNavigationBar: BottomNavigationBar(
        elevation: 0.0,
        iconSize: 25,

        backgroundColor: Colors.blue.shade700,
        currentIndex: _currentIndex,
        onTap: (index) {
          setState(() {
            _currentIndex = index;
          });
        },
        items: [
          BottomNavigationBarItem(
            icon: Icon(
              Icons.send,
            ),
            label: 'Send',
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.post_add,
            ),
            label: 'Received',
          ),
        ],
        unselectedItemColor: Colors.white,
        selectedLabelStyle: TextStyle(
          fontSize: 15,
          fontWeight: FontWeight.bold,
        ),
        fixedColor: Colors.red,
        unselectedLabelStyle: TextStyle(
          fontSize: 12,
          fontWeight: FontWeight.bold,

```

```

    ),
  ),
),
);
}

```

```

Widget _buildPage(int pageIndex, Widget pageContent) {
  return AnimatedOpacity(
    opacity: _currentIndex == pageIndex ? 1.0 : 0.0,
    duration: Duration(milliseconds: 300),
    child: IgnorePointer(
      ignoring: _currentIndex != pageIndex,
      child: pageContent,
    ),
  );
}
}

```

→ Login Page

```

import 'package:flutter/material.dart';
import 'package:path/path.dart';
import '../compenents/my_button.dart';
import '../compenents/squaretile.dart';
import '../compenents/textfield.dart';
import 'CustomBottomNavigation.dart';
import 'DatabaseHelper.dart';
import 'HomePageNew.dart';

```

```

class LoginPageSql extends StatefulWidget {
  final Function()? onTap;

```

```

  const LoginPageSql({this.onTap, Key? key}) : super(key: key);

```

```

  @override
  State<LoginPageSql> createState() => _LoginPageSqlState();
}

```

```

class _LoginPageSqlState extends State<LoginPageSql> {
  final emailController = TextEditingController();
  final passwordController = TextEditingController();
  bool isLoading = false;
  bool showPassword = false;

```

```

  late DatabaseHelper _databaseHelper;

```

```

  @override
  void initState() {
    super.initState();
    _databaseHelper = DatabaseHelper();

```



```

_initializeDatabase();
}

Future<void> _initializeDatabase() async {
  await _databaseHelper.initializeDatabase();
}

void togglePasswordVisibility() {
  setState(() {
    showPassword = !showPassword;
  });
}

void signIn() async {
  // Check if email and password fields are not empty
  if (emailController.text.isEmpty || passwordController.text.isEmpty) {
    // Do not sign in
    showErrorMessage(
      this.context, "Please enter all the required information.");
    return;
  }
  try {
    final user = await _databaseHelper.getUser(emailController.text);
    if (user != null && user.password == passwordController.text) {
      // User login successful
      // Navigate to the homepage
      Navigator.pushAndRemoveUntil(
        this.context,
        MaterialPageRoute(
          builder: (context) => CustomBottomNavigation(),
        ),
        (route) => false);
    } else {
      showErrorMessage(this.context, "Invalid Email or Password.");
    }
  } catch (e) {
    showErrorMessage(this.context, "Error logging in.");
  }
}

// Error message
void showErrorMessage(BuildContext context, String message) {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        backgroundColor: Colors.blue.shade700,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(20.0),

```

```

        side: BorderSide(
          color: Colors.white, // Transparent border color
          width: 2.0, // Border width
        ),
      ),
    title: Center(
      child: Text(
        message,
        style: const TextStyle(
          color: Colors.white,
          fontSize: 20,
          fontWeight: FontWeight.bold),
      ),
    ),
  );
},
);
}
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.blue.shade700,
    body: SafeArea(
      child: Center(
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              // Logo
              const Icon(
                Icons.lock_person,
                size: 120,
                color: Colors.white,
              ),
              const SizedBox(height: 15),
              Text(
                "Welcome Back!",
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const SizedBox(height: 20),
              MyTextField(
                controller: emailController,
                hintText: "Email",
                obscureText: false,

```

```

),
const SizedBox(height: 20),
MyTextField(
  controller: passwordController,
  hintText: "Password",
  obscureText: !showPassword,
  suffixIcon: GestureDetector(
    onTap: togglePasswordVisibility,
    child: Icon(
      showPassword ? Icons.visibility : Icons.visibility_off,
      color: Colors.white,
    ),
  ),
),
const SizedBox(height: 25),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text(
      "Forgot Password?",
      style: TextStyle(
        color: Colors.white, fontWeight: FontWeight.bold),
    ),
  ],
),
const SizedBox(height: 25),
MyButton(
  onTap: signInUser,
  text: "Sign In",
),
const SizedBox(height: 20),
Row(
  children: [
    Expanded(
      child: Divider(
        thickness: 2,
        color: Colors.white,
      ),
    ),
    Padding(
      padding: const EdgeInsets.symmetric(horizontal: 10),
      child: Text(
        "Or",
        style: TextStyle(
          color: Colors.white, fontWeight: FontWeight.bold),
      ),
    ),
    Expanded(
      child: Divider(

```

```

        thickness: 2,
        color: Colors.white,
      ),
    ),
  ],
),
const SizedBox(height: 10),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    SquareTile(imagePath: "images/G.png"),
    SizedBox(
      width: 10,
    ),
    SquareTile(imagePath: "images/A.png"),
  ],
),
const SizedBox(height: 35),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text(
      "Not a Member?",
      style: TextStyle(
        color: Colors.white,
        fontWeight: FontWeight.bold,
        fontSize: 15),
    ),
    const SizedBox(
      width: 5,
    ),
    GestureDetector(
      onTap: widget.onTap,
      child: Text(
        "Register Now",
        style: TextStyle(
          color: Colors.redAccent,
          fontWeight: FontWeight.bold,
          fontSize: 15),
      ),
    ),
  ],
),
),
),
),
),
),
);

```

```

    }
  }

```

Register Page

```

import 'package:flutter/material.dart';
import '../compenents/my_button.dart';
import '../compenents/squaretile.dart';
import '../compenents/textfield.dart';
import 'DatabaseHelper.dart';
import 'HomePageNew.dart';
import 'LoginPage_sqlite.dart';

class RegisterPageSql extends StatefulWidget {
  final Function()? onTap;

  const RegisterPageSql({this.onTap, super.key});

  @override
  State<RegisterPageSql> createState() => _RegisterPageSqlState();
}

class _RegisterPageSqlState extends State<RegisterPageSql> {
  final emailController = TextEditingController();
  final passwordController = TextEditingController();
  final confirmPasswordController = TextEditingController();
  bool isPasswordVisible = false;
  bool isConfirmPasswordVisible = false;
  bool isLoading = false;

  late DatabaseHelper _databaseHelper;

  @override
  void initState() {
    super.initState();
    _databaseHelper = DatabaseHelper();
    _initializeDatabase();
  }

  Future<void> _initializeDatabase() async {
    await _databaseHelper.initializeDatabase();
  }

  void togglePasswordVisibility() {
    setState(() {
      isPasswordVisible = !isPasswordVisible;
    });
  }

  void toggleConfirmPasswordVisibility() {

```

```

    setState() {
      isConfirmPasswordVisible = !isConfirmPasswordVisible;
    });
  }

  void signUp(BuildContext context) async {
    // Check if passwords match
    if (passwordController.text == confirmPasswordController.text) {
      // Check if email is empty
      if (emailController.text.isEmpty) {
        showErrorMessage(context, "Please enter email");
        return;
      }

      // Check if password is empty
      if (passwordController.text.isEmpty) {
        showErrorMessage(context, "Please enter password");
        return;
      }

      final existingUser = await
        _databaseHelper.getUser(emailController.text);
      if (existingUser != null) {
        showErrorMessage(context, "Email is already used.");
        return;
      }

      // Insert user into database
      await _databaseHelper.insertUser(
        User(
          email: emailController.text,
          password: passwordController.text,
        ),
      );
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => LoginPageSql(),
        ),
      );
      showErrorMessage(context, "Successfully registered. You can now log
in.");
    } else {
      showErrorMessage(context, "Passwords don't match!");
    }
  }

  // Error message
  void showErrorMessage(BuildContext context, String message) {

```

```

showDialog(
  context: context,
  builder: (BuildContext context) {
    return AlertDialog(
      backgroundColor: Colors.blue.shade700,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(20.0),
        side: BorderSide(
          color: Colors.white, // Transparent border color
          width: 2.0, // Border width
        ),
      ),
      title: Center(
        child: Text(
          message,
          style: const TextStyle(
            color: Colors.white,
            fontSize: 20,
            fontWeight: FontWeight.bold),
        ),
      ),
    );
  },
);
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.blue.shade700,
    body: SafeArea(
      child: Center(
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              // const SizedBox(
              //   height: 10,
              // ),
              //logo
              const Icon(
                Icons.account_box_outlined,
                size: 120,
                color: Colors.white,
              ),
              //welcome back
              const SizedBox(height: 20),
              Text(
                "Let's create an account for you!",

```

```

        style: TextStyle(
          color: Colors.white,
          fontSize: 20,
          fontWeight: FontWeight.bold,
        ),
      ),

      const SizedBox(
        height: 20,
      ),
      //user textfield
      MyTextField(
        controller: emailController,
        hintText: "Email",
        obscureText: false,
      ),

      const SizedBox(
        height: 20,
      ),
      // password
      MyTextField(
        controller: passwordController,
        hintText: "Password",
        obscureText: !isPasswordVisible,
        // Toggle obscureText based on state
        suffixIcon: GestureDetector(
          onTap: togglePasswordVisibility,
          // Toggle password visibility on tap
          child: Icon(
            isPasswordVisible
              ? Icons.visibility
              : Icons.visibility_off,
            color: Colors.white,
          ),
        ),
      ),
      const SizedBox(
        height: 20,
      ),
      // Confirm Password TextField with Show Password functionality
      MyTextField(
        controller: confirmPasswordController,
        hintText: "Confirm Password",
        obscureText: !isConfirmPasswordVisible,
        // Toggle obscureText based on state
        suffixIcon: GestureDetector(
          onTap: toggleConfirmPasswordVisibility,
          // Toggle password visibility on tap

```



```

        child: Icon(
          isConfirmPasswordVisible
            ? Icons.visibility
            : Icons.visibility_off,
          color: Colors.white,
        ),
      ),
    ),
  ),

  const SizedBox(
    height: 25,
  ),
  //sign in

  // const MyButton(),
  MyButton(
    onTap: () => signUserUp(context),
    text: "Sign Up",
  ),
  const SizedBox(
    height: 20,
  ),
  // continue with
  //or continue with
  Row(
    children: [
      Expanded(
        child: Divider(
          thickness: 2,
          color: Colors.white,
        ),
      ),
      Padding(
        padding: const EdgeInsets.symmetric(horizontal: 10),
        child: Text(
          "Or",
          style: TextStyle(
            color: Colors.white, fontWeight: FontWeight.bold),
        ),
      ),
      Expanded(
        child: Divider(
          thickness: 2,
          color: Colors.white,
        ),
      ),
    ],
  ),
),

```

```

const SizedBox(
  height: 10,
),
const Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    SquareTile(imagePath: "images/G.png"),
    SizedBox(
      width: 10,
    ),
    SquareTile(imagePath: "images/A.png"),
  ],
),
const SizedBox(
  height: 35,
),

Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text(
      "Already have a account?",
      style: TextStyle(
        color: Colors.white,
        fontWeight: FontWeight.bold,
        fontSize: 15),
    ),
    const SizedBox(
      width: 4,
    ),
    GestureDetector(
      onTap: widget.onTap,
      child: const Text(
        "Login now",
        style: TextStyle(
          color: Colors.redAccent,
          fontWeight: FontWeight.bold,
          fontSize: 15),
      ),
    ),
  ],
),
],
),
),
),
),
),
),
);
}

```

```
}
```

- HomePage(Sending)

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:shared_preferences/shared_preferences.dart';
import '../Cameras/Camera.dart';
import 'DatabaseHelper.dart';
import 'RegisterPage_sqlite.dart';
import 'authPage_sql.dart';

class HomePageNew extends StatefulWidget {
  const HomePageNew({Key? key}) : super(key: key);

  @override
  State<HomePageNew> createState() => _HomePageNewState();
}

class _HomePageNewState extends State<HomePageNew> {
  final DatabaseHelper databaseHelper = DatabaseHelper();
  String formattedDate = DateFormat('yyyy, dd
MMMM').format(DateTime.now());
  String formattedDay = DateFormat('EEEE').format(DateTime.now());

  @override
  void initState() {
    super.initState();
  }

  void signUserOut() async {
    // Navigate to the login page
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(
        builder: (context) => const AuthPageSql(),
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.blue.shade700,
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            children: [
              Row(
```

```

mainAxisAlignment: MainAxisAlignment.spaceBetween,
children: [
  Padding(
    padding: const EdgeInsets.symmetric(
      horizontal: 20, vertical: 15),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          formattedDay,
          style: TextStyle(
            color: Colors.white,
            fontSize: 28,
            fontWeight: FontWeight.bold,
          ),
        ),
        SizedBox(
          height: 2,
        ),
        Text(
          formattedDate,
          style: TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        ),
      ],
    ),
  ),
  Padding(
    padding: const EdgeInsets.all(8.0),
    child: Container(
      decoration: BoxDecoration(
        color: Colors.blue.shade600,
        borderRadius: BorderRadius.circular(15),
        border: Border.all(
          color: Colors.transparent,
          width: 2,
        ),
      ),
      padding: EdgeInsets.all(15),
      child: GestureDetector(
        onTap: signUserOut,
        child: Icon(
          Icons.logout,
          color: Colors.white,
          size: 25,
          semanticLabel: "Logout",
        ),
      ),
    ),
  ),
],
),

```

```

    ),
    ),
    ),
  ],
),
 SizedBox(
  height: 10,
),
 Padding(
  padding:
    const EdgeInsets.symmetric(vertical: 0, horizontal: 10),
  child: Container(
    decoration: BoxDecoration(
      color: Colors.blue.shade600,
      borderRadius: BorderRadius.circular(10),
      border: Border.all(
        color: Colors.transparent,
        width: 1,
      ),
    ),
    padding: const EdgeInsets.symmetric(
      horizontal: 10, vertical: 15),
    child: const Column(
      children: [
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            Text(
              "Actions",
              style: TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.bold,
                fontSize: 25,
              ),
            ),
            Icon(
              Icons.more_horiz,
              color: Colors.white,
            ),
          ],
        ),
      ],
    ),
  ),
),
const SizedBox(
  height: 15,
),
 Padding(

```

```

padding: const EdgeInsets.all(15.0),
child: Container(
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(35),
    border: Border.all(
      color: Colors.white, // Specify the border color here
      width: 1, // Adjust the border width as needed
    ),
  ),
child: ClipRRect(
  borderRadius: BorderRadius.circular(35),
  child: Container(
    height: 570,
    color: Colors.blue.shade800,
    child: Center(
      child: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          children: [
            Row(
              mainAxisAlignment:
                MainAxisAlignment.spaceBetween,
              children: [
                Text("Upload",
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                    fontSize: 24,
                    color: Colors.white)),
                Icon(
                  Icons.more_horiz,
                  color: Colors.white,
                ),
              ],
            ),
            SizedBox(
              height: 20,
            ),
            Container(
              height: 90,
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(20),
                color: Colors.blue.shade600,
                border: Border.all(
                  color: Colors.transparent,
                ),
              ),
            ),
            child: Align(
              alignment: Alignment.center,
              child: GestureDetector(

```


- HomePage(Receiving)

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'DatabaseHelper.dart';
import 'authPage_sql.dart';
import 'package:http/http.dart' as http;

class HomePageNew2 extends StatefulWidget {
  const HomePageNew2({Key? key}) : super(key: key);

  @override
  State<HomePageNew2> createState() => _HomePageNew2State();
}

class _HomePageNew2State extends State<HomePageNew2> {
  final DatabaseHelper databaseHelper = DatabaseHelper();
  String formattedDate = DateFormat('yyyy, dd
MMMM').format(DateTime.now());
  String formattedDay = DateFormat('EEEE').format(DateTime.now());
  List<ImageInfo> images = [];

  @override
  void initState() {
    super.initState();
    fetchImagesFromServer();
  }

  void signUserOut() async {
    // Navigate to the login page
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(
        builder: (context) => const AuthPageSql(),
      ),
    );
  }

  Future<void> fetchImagesFromServer() async {
    try {
      final response = await http.get(Uri.parse(
        'http://192.168.29.87:3000/images')); // Replace with your server URL
      if (response.statusCode == 200) {
        // Images fetched successfully
        final List<dynamic> imageList = json.decode(response.body);
        setState(() {
          images = imageList.map((data) => ImageInfo.fromJson(data)).toList();
          images = images.reversed.toList(); // Reverse the order
        });
      }
    }
  }
}
```



```

    } else {
      // Handle the error when fetching images
      print(
        'Failed to fetch images with status code: ${response.statusCode}');
    }
  } catch (e) {
    print('Error fetching images from server: $e');
  }
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.blue.shade700,
    body: SafeArea(
      child: SingleChildScrollView(
        child: Column(
          children: [
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Padding(
                  padding: const EdgeInsets.symmetric(
                    horizontal: 20, vertical: 15),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text(
                        formattedDay,
                        style: TextStyle(
                          color: Colors.white,
                          fontSize: 28,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      SizedBox(
                        height: 2,
                      ),
                      Text(
                        formattedDate,
                        style: TextStyle(
                          color: Colors.white,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                    ],
                  ),
                ),
                Padding(

```

```
padding: const EdgeInsets.all(8.0),  
child: Container(  
  decoration: BoxDecoration(  
    color: Colors.blue.shade600,  
    borderRadius: BorderRadius.circular(15),  
    border: Border.all(  
      color: Colors.transparent,  
      width: 2,  
    ),  
  ),  
  padding: EdgeInsets.all(15),  
  child: GestureDetector(  
    onTap: signUserOut,  
    child: Icon(  
      Icons.logout,  
      color: Colors.white,  
      size: 25,  
      semanticLabel: "Logout",  
    ),  
  ),  
),  
],  
),  
 SizedBox(  
  height: 10,  
),  
 Padding(  
  padding:  
const EdgeInsets.symmetric(vertical: 0, horizontal: 10),  
  child: Container(  
    decoration: BoxDecoration(  
      color: Colors.blue.shade600,  
      borderRadius: BorderRadius.circular(10),  
      border: Border.all(  
        color: Colors.transparent,  
        width: 1,  
      ),  
    ),  
    padding:  
const EdgeInsets.symmetric(horizontal: 10, vertical: 15),  
    child: const Column(  
      children: [  
        Row(  
          mainAxisAlignment: MainAxisAlignment.spaceBetween,  
          children: [  
            Text(  
              "Post",  
              style: TextStyle(  

```

```

        color: Colors.white,
        fontWeight: FontWeight.bold,
        fontSize: 25,
      ),
    ),
    Icon(
      Icons.more_horiz,
      color: Colors.white,
    ),
  ],
)
],
),
),
),
const SizedBox(
  height: 15,
),
for (final imageInfo in images)
  ImageAndDescriptionCard(imageInfo: imageInfo),
],
),
),
),
);
}
}

```

// Define a class to represent the image and description

```

class ImageInfo {
  final String fileName;
  final String description;
  final String image;

```

```

  ImageInfo({
    required this.fileName,
    required this.description,
    required this.image,
  });

```

// Create an instance from JSON data

```

factory ImageInfo.fromJson(Map<String, dynamic> json) {
  return ImageInfo(
    fileName: json['fileName'] as String,
    description: json['description'] as String,
    image: json['image'] as String,
  );
}
}

```

```

class ImageAndDescriptionCard extends StatelessWidget {
  final ImageInfo imageInfo;

  const ImageAndDescriptionCard({Key? key, required this.imageInfo})
    : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(10.0),
      child: Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(35),
          border: Border.all(
            color: Colors.white, // Specify the border color here
            width: 1, // Adjust the border width as needed
          ),
        ),
        child: ClipRRect(
          borderRadius: BorderRadius.circular(35),
          child: Container(
            // height: 570,
            color: Colors.blue.shade800,
            child: Center(
              child: Padding(
                padding: const EdgeInsets.all(20.0),
                child: Column(
                  children: [
                    Row(
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      children: [
                        Text("New Posts",
                          style: TextStyle(
                            fontWeight: FontWeight.bold,
                            fontSize: 24,
                            color: Colors.white)),
                        Icon(
                          Icons.more_horiz,
                          color: Colors.white,
                        ),
                      ],
                    ),
                    SizedBox(height: 10),
                    Image.memory(
                      // Decode base64 image data
                      base64.decode(imageInfo.image),
                      fit: BoxFit.contain,
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}

```

```

        SizedBox(height: 10),
        Text(
          imageInfo.description,
          style: TextStyle(
            color: Colors.white,
          ),
        ),
      ],
    ),
  ),
),
),
),
),
),
),
),
);
}
}

```

- Camera Functionality

- CameraScreen

```

import 'dart:io';
import 'dart:math';
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import
'package:flutter_image_compress/flutter_image_compress.dart';
import 'package:path_provider/path_provider.dart';
import 'CameraView.dart';
import 'package:intl/intl.dart';
import 'package:firebase_storage/firebase_storage.dart';

```

```

List<CameraDescription> cameras = [];

```

```

class CameraScreen extends StatefulWidget {
  CameraScreen({Key? key}) : super(key: key);

```

```

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

```

```

class _CameraScreenState extends State<CameraScreen> {
  late CameraController _cameraController;
  late Future<void> cameraValue;
  bool flash = false;
  bool iscamerafront = false;
  double transform = 0;
  int pictureCounter = 0; // Counter to keep track of captured
  pictures

```

```

@override
void initState() {
  super.initState();
  _cameraController = CameraController(cameras[0],
    ResolutionPreset.high);
  cameraValue = _cameraController.initialize();
}

@override
void dispose() {
  super.dispose();
  _cameraController.dispose();
}

Future<void> takePhoto(BuildContext context) async {
  try {
    final XFile picture = await _cameraController.takePicture();
    pictureCounter++;

    final DateTime now = DateTime.now();
    final String formattedDateTime =
      DateFormat('yyyyMMdd_HH:mm:ss').format(now);
    final String fileName =
      'IMG_${formattedDateTime}_${pictureCounter}.png';

    final appDir = await getApplicationDocumentsDirectory();
    final permanentImagePath = '${appDir.path}/${fileName}';
    final File permanentImageFile = File(permanentImagePath);

    final compressedBytes = await
FlutterImageCompress.compressWithFile(
      picture.path,
      quality: 85, // Adjust the quality as needed (0 - 100)
    );

    // Write the compressed bytes to the permanent location
    await
permanentImageFile.writeAsBytes(compressedBytes?.toList() ?? []);
    // Navigate to the CameraViewPage with the permanent image
    path
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(
        builder: (context) => CameraViewPage(path:
permanentImagePath),
      ),
    );
  } catch (e) {

```

```

        print("Error taking picture: $e");
    }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      leading: IconButton(
        icon: Icon(
          Icons.arrow_back,
          color: Colors.white,
        ),
        onPressed: () {
          Navigator.pop(context); // Pop the current screen to go back
        },
      ),
      backgroundColor: Colors.transparent, // Make the app bar
transparent
      elevation: 0, // Remove the shadow
    ),
    backgroundColor: Colors.black,
    body: Stack(
      children: [
        FutureBuilder(
          future: cameraValue,
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.done) {
              return Container(
                width: MediaQuery.of(context).size.width,
                height: MediaQuery.of(context).size.height,
                child: Stack(
                  children: [
                    Positioned(
                      top: 0,
                      bottom: 135,
                      left: 6,
                      right: 6,
                      child: AspectRatio(
                        aspectRatio: 16 / 9,
                        child: CameraPreview(_cameraController),
                      ),
                    ),
                  ],
                ),
              );
            } else {
              return Center(
                child: Container(

```

```

width: 100, // Set the desired width for the container
height: 50, // Set the desired height for the container
decoration: BoxDecoration(
  borderRadius: BorderRadius.circular(8.0),
  border: Border.all(color: Colors.transparent),
  color:
    Colors.white, // Set the background color to white
),
child: Center(
  child: CircularProgressIndicator(
    color: Colors.white,
  ),
),
),
);
}
}),
Positioned(
  bottom: 5,
  child: SafeArea(
    child: Container(
      color: Colors.transparent,
      padding: const EdgeInsets.all(30),
      width: MediaQuery.of(context).size.width,
      child: Column(
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.max,
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: [
              IconButton(
                icon: Icon(
                  flash ? Icons.flash_on : Icons.flash_off,
                  color: Colors.white,
                  size: 35,
                ),
                onPressed: () {
                  setState(() {
                    flash = !flash;
                  });
                  flash
                    ? _cameraController
                      .setFlashMode(FlashMode.torch)
                    : _cameraController
                      .setFlashMode(FlashMode.off);
                },
              ),
              InkWell(
                onTap: () {
                  takePhoto(context);
                },
              ),
            ],
          ),
        ],
      ),
    ),
  ),
);

```



```

    },
    child: Icon(
      Icons.panorama_fish_eye,
      color: Colors.white,
      size: 80,
    ),
  ),
  IconButton(
    icon: Transform.rotate(
      angle: transform,
      child: const Icon(
        Icons.flip_camera_android,
        color: Colors.white,
        size: 35,
      ),
    ),
    onPressed: () async {
      setState(() {
        iscamerafront = !iscamerafront;
        transform = transform + pi;
      });
      int cameraPos = iscamerafront ? 0 : 1;
      _cameraController = CameraController(
        cameras[cameraPos], ResolutionPreset.high);
      cameraValue = _cameraController.initialize();
    },
  ],
),
const SizedBox(
  height: 0,
),
const Text(
  "Capture",
  style: TextStyle(
    color: Colors.white,
    fontSize: 12,
    fontWeight: FontWeight.bold),
  textAlign: TextAlign.center,
)
],
),
),
),
),
),
],
),
);
}
}

```

→ CameraView

```

import 'dart:convert';
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

import '../sqlite/CustomBottomNavigationBar.dart';
import '../sqlite/HomePageNew.dart';

class CameraViewPage extends StatelessWidget {
  CameraViewPage({Key? key, required this.path}) : super(key: key);
  final String path;
  TextEditingController _descriptionController =
    TextEditingController();

  void showErrorMessage(BuildContext context, String message) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          backgroundColor: Colors.blue.shade700,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(20.0),
            side: BorderSide(
              color: Colors.white, // Transparent border color
              width: 2.0, // Border width
            ),
          ),
          title: Center(
            child: Text(
              message,
              style: const TextStyle(
                color: Colors.white,
                fontSize: 20,
                fontWeight: FontWeight.bold),
            ),
          ),
        );
      },
    );
  }

  Future<void> sendImageToServer(
    BuildContext context, File imageFile, String description) async {
    try {
      // Replace with your server URL
      final Uri serverUri = Uri.parse(
        'http://192.168.29.87:3000/upload'); // Replace with your actual
      server URL
    }
  }

```

```

    // Create a multipart request to send both the image and
description
    final http.MultipartRequest request =
        http.MultipartRequest('POST', serverUri);

    // Add the image file to the request
    final http.MultipartFile imageMultipartFile =
        await http.MultipartFile.fromPath(
            'image', // The field name for the image on the server
            imageFile.path, // Path to the image file
        );

    request.files.add(imageMultipartFile);

    // Add the description as a field in the request
    request.fields['description'] = description;

    // Send the request
    final http.StreamedResponse response = await request.send();

    if (response.statusCode == 200) {
        // Request was successful, handle the server's response here
        // You can display a success message or navigate to a new
screen as needed.
        Navigator.pushReplacement(
            context,
            MaterialPageRoute(
                builder: (context) =>
                    CustomBottomNaviagtion(), // Replace with the actual
name of your HomePageNew widget
            ),
        );
        showErrorMessage(context, 'Image uploaded successfully');
    } else {
        // Request failed, handle the error here
        Navigator.pushReplacement(
            context,
            MaterialPageRoute(
                builder: (context) =>
                    CustomBottomNaviagtion(), // Replace with the actual
name of your HomePageNew widget
            ),
        );
        showErrorMessage(context,
            'Image upload failed with status code:
${response.statusCode}! Try Again');
    }
    } catch (e) {

```

```

Navigator.pushReplacement(
  context,
  MaterialPageRoute(
    builder: (context) =>
      CustomBottomNaviagtion(), // Replace with the actual
name of your HomePageNew widget
  ),
);
// Handle any exceptions or errors here
showErrorMessage(context, 'Error sending image to server: $e');
}
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.black,
    appBar: AppBar(
      backgroundColor: Colors.black,
      actions: [],
    ),
    body: Container(
      width: MediaQuery.of(context).size.width,
      height: MediaQuery.of(context).size.height,
      child: Stack(
        children: [
          Container(
            width: MediaQuery.of(context).size.width,
            height: MediaQuery.of(context).size.height,
            child: Stack(
              children: [
                Positioned(
                  top: 0,
                  bottom: 100,
                  left: 5,
                  right: 5,
                  child: Image.file(
                    File(path),
                  ),
                )
              ],
            ),
          ),
          Positioned(
            bottom: 0,
            child: Container(
              color: Colors.black,
              width: MediaQuery.of(context).size.width,
              padding: EdgeInsets.symmetric(vertical: 15, horizontal: 5),
              child: TextFormField(

```

```

        controller: _descriptionController,
        style: TextStyle(
          color: Colors.white,
          fontSize: 16,
        ),
        maxLength: 300,
        maxLines: null,
        minLines: 1,
        decoration: InputDecoration(
          border: InputBorder.none,
          hintText: "Add Description...(300 Words)",
          prefixIcon: Icon(
            Icons.text_fields,
            color: Colors.white,
            size: 25,
          ),
          hintStyle: TextStyle(
            color: Colors.white,
            fontSize: 15,
          ),
          suffixIcon: IconButton(
            icon: CircleAvatar(
              radius: 40,
              backgroundColor: Colors.green,
              child: Icon(
                Icons.check,
                color: Colors.white,
                size: 20,
              ),
            ),
          ),
          onPressed: () async {
            String description = _descriptionController.text;
            File imageFile = File(path);
            sendImageToServer(context, imageFile, description);
          },
        ),
      ),
    ),
  ],
),
);
}
}

```

- Server Hosting(Node Js)

```

const express = require('express');
const multer = require('multer');
const app = express();
const port = process.env.PORT || 3000;
const fs = require('fs');
const path = require('path');

let pictureCounter = 0;

const storage = multer.memoryStorage();
const upload = multer({ storage: storage });

// Set the directory where images and descriptions will be saved
const uploadDirectory = 'ServerImages';

// Ensure the upload directories exist
if (!fs.existsSync(uploadDirectory)) {
  fs.mkdirSync(uploadDirectory);
}

app.get('/images', (req, res) => {
  try {
    // Read the list of image files from the 'ServerImages' directory
    const imageFiles = fs.readdirSync(uploadDirectory);

    // Create an array to store image and description data
    const images = [];

    // Read each image file and corresponding JSON file
    for (const imageFile of imageFiles) {
      if (imageFile.endsWith('.json')) {
        const imagePath = path.join(uploadDirectory, imageFile);
        const jsonContent = fs.readFileSync(imagePath, 'utf-8');
        const imageData = JSON.parse(jsonContent);
        images.push(imageData);
      }
    }

    res.status(200).json(images);
  } catch (error) {
    console.error('Error fetching images:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});

app.post('/upload', upload.single('image'), (req, res) => {
  try {
    const imageBuffer = req.file.buffer;
  }
});

```

```

const description = req.body.description;

const now = new Date();
const formattedDateTime = now.toISOString().replace(/[-T:]/g, "");
const fileName1 = `IMG_${formattedDateTime}_${pictureCounter}.json`;
const fileName2 = `IMG_${formattedDateTime}_${pictureCounter}.jpg`;

const savePath = path.join(uploadDirectory, fileName1);
const savePath2 = path.join(uploadDirectory, fileName2);

const imageInfo = {
  fileName: fileName1,
  description,
  image: imageBuffer.toString('base64'),
};

fs.writeFileSync(savePath, JSON.stringify(imageInfo, null, 2));
fs.writeFileSync(savePath2, imageBuffer);
pictureCounter++;

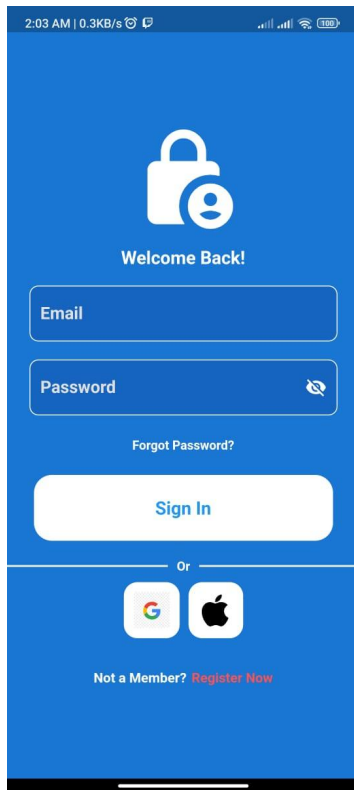
res.status(200).json({ message: 'Image and description uploaded successfully' });
} catch (error) {
  console.error('Error handling upload:', error);
  res.status(500).json({ error: 'Internal server error' });
}
});

app.listen(port, () => {
  console.log(`Server is listening on port ${port}`);
});

```

Output: -

- LoginPage



2:03 AM | 0.3KB/s

Welcome Back!



Email

Password

Forgot Password?

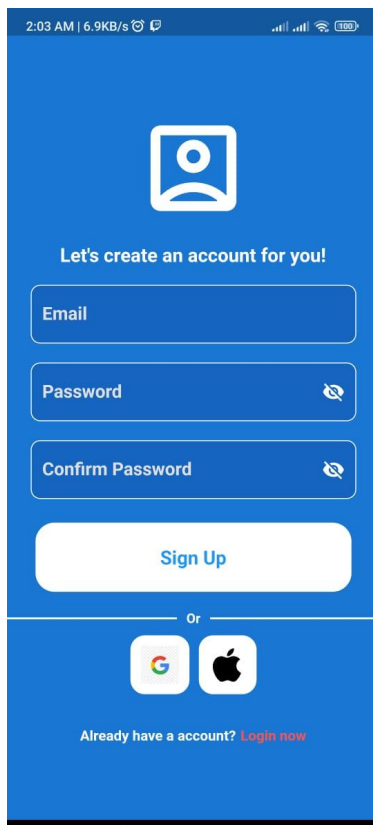
Sign In

Or

Not a Member? [Register Now](#)

- RegisterPage



2:03 AM | 6.9KB/s

Let's create an account for you!



Email

Password

Confirm Password

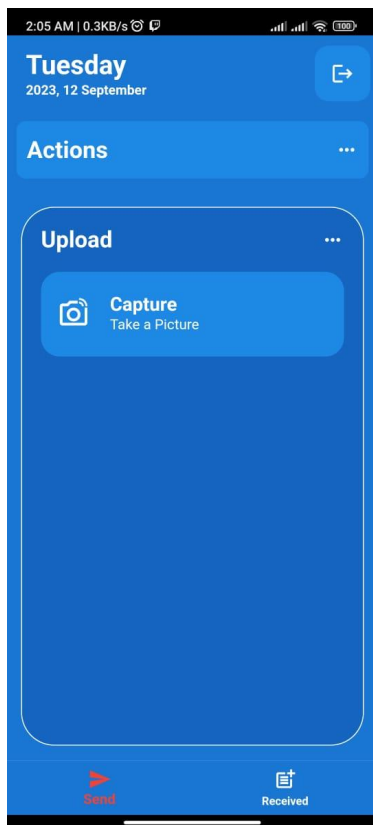
Sign Up

Or

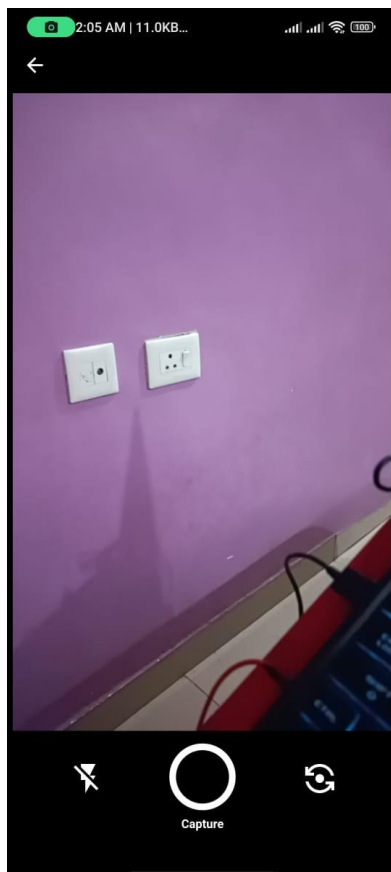
 

Already have an account? [Login now](#)

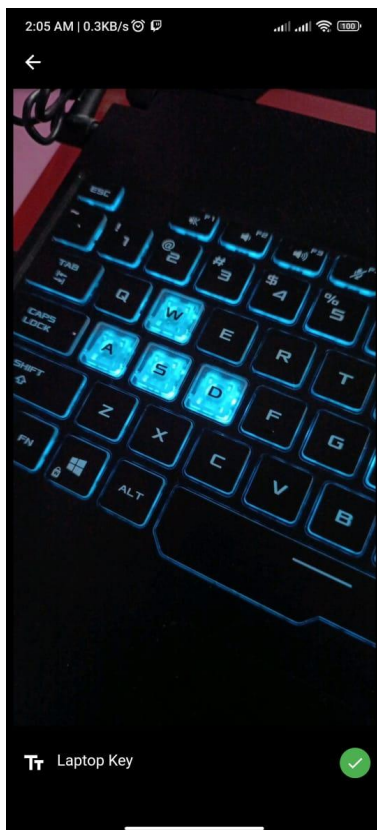
- HomePage(Sending Page)



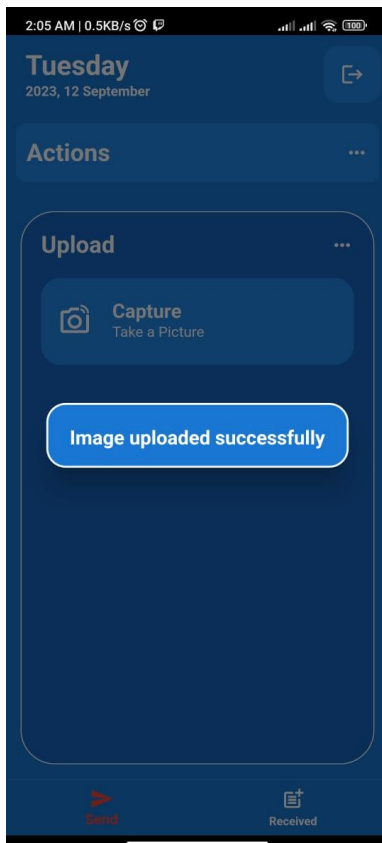
- CameraScreen



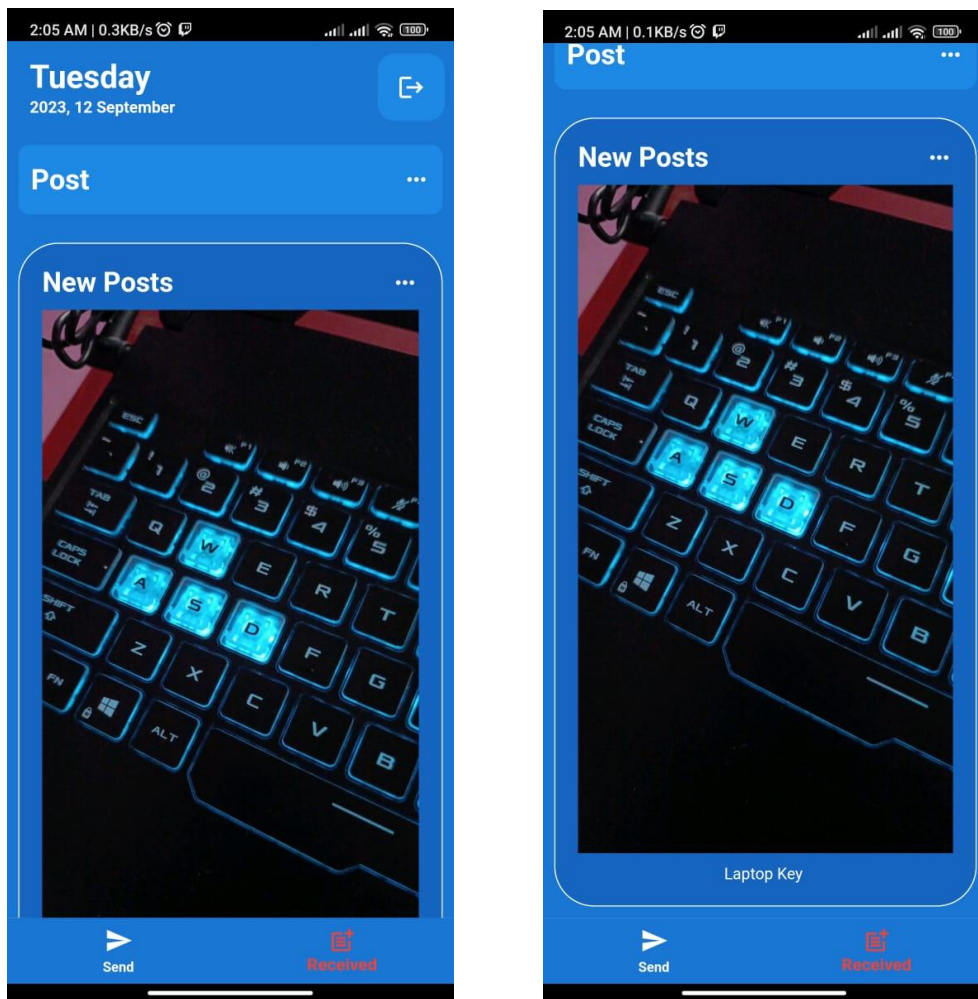
- CameraViewPage



- Server Status



- HomePage(Receiving Page)



Conclusion:

This project has made an Android app using flutter that lets users capture and share site incident images in real time, with ease and clarity. The app aims to enhance site incident reporting, communication, and collaboration. The app has features like image capture, annotation, sharing, dashboard, The app is designed for Android 4.4 or higher, and uses Android Studio and various APIs and libraries. The project team has found that the app has met the project's objectives and requirements, and has shown its effectiveness and potential. The project team has also identified some limitations and future work of the app.