HAPPENINGS

An Event Management App

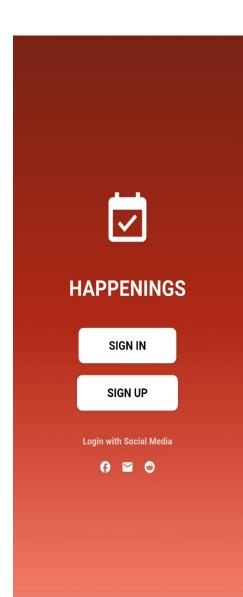
CE-152 : Savaliya Achyut

CE-162: Vadher Akshar



Group No.: 05

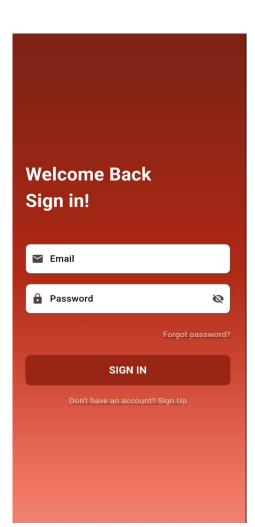
Initial Screen



```
Future<void> main() async {
 WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp(
   options: DefaultFirebaseOptions.currentPlatform,
 User? user = FirebaseAuth.instance.currentUser;
 runApp(
   MyApp(
       initialScreen: user == null
            ? const LoginSignupPage()
            : const MainScreen(currentIndex: 1)), // MyApp
```

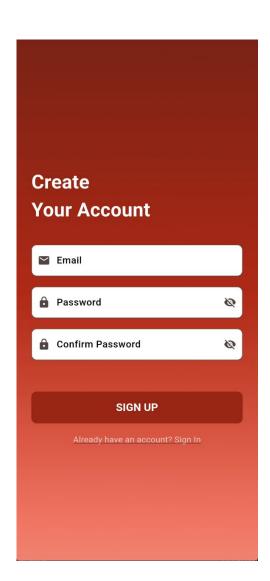
home: initialScreen,

Login



```
void handleSignIn() async {
 try {
   UserCredential userCredential =
        await FirebaseAuth.instance.signInWithEmailAndPassword(
     email: emailController.text,
      password: passwordController.text,
    _showSnackBar(
        'Sign in successful! Welcome back, ${userCredential.user?.email}');
   Navigator.pushReplacement(
      context,
     MaterialPageRoute(
        builder: (context) => const MainScreen(currentIndex: 1),
     ), // MaterialPageRoute
   on FirebaseAuthException catch (e) {
   _showSnackBar(e.message ?? 'Sign-in failed', isError: true);
```

Signup



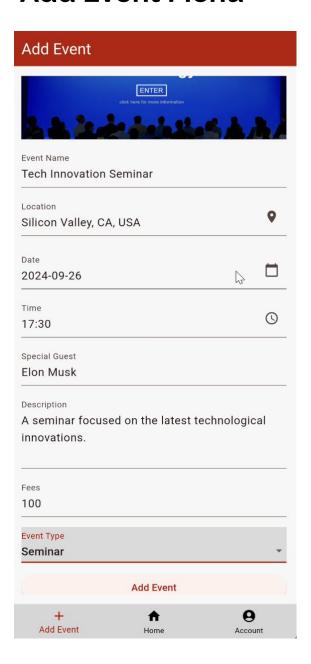
```
void _handleSignUp() async {
 try {
   if (passwordController.text == confirmPasswordController.text) {
     UserCredential userCredential =
          await FirebaseAuth.instance.createUserWithEmailAndPassword(
       email: emailController.text,
       password: passwordController.text,
     );
     _showSnackBar(
          'Sign up successful! Welcome, ${userCredential.user?.email}');
     Navigator.pushReplacement(
       context,
       MaterialPageRoute(
         builder: (context) => const MainScreen(currentIndex: 1),
        ), // MaterialPageRoute
    } else {
     _showSnackBar('Passwords do not match', isError: true);
  } on FirebaseAuthException catch (e) {
   _showSnackBar(e.message ?? 'Sign-up failed', isError: true);
```

Events Page

```
Events
My Events
            Leadership Skills
                                      27
  EADERSHII Workshop
                                      Sep
            9 Boston, MA, USA
            Data Science Bootcamp 10
             Chicago, IL, USA
                                       Oct
            Digital Transformation
   ITAL
INSFORMATI
                                      31
            Seminar
                                      Oct
             Seattle, WA, USA
            Cloud Computing
                                      14
            Conference
                                      Nov
             Miami, FL, USA
Other Events
             Tech Innovation
                                      26
            Seminar
                                      Sep
             Silicon Valley, CA, USA
            Sustainable Energy
                                      27
            Seminar
                                      Sep
             Austin, TX, USA
                                      0
                     Home
   Add Event
```

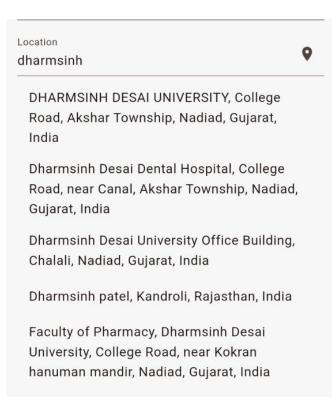
```
Future<Map<String, List<Event>>> fetchEvents() async {
 QuerySnapshot querySnapshot =
     await FirebaseFirestore.instance.collection('events').get();
 DateTime now = DateTime.now();
 final user = FirebaseAuth.instance.currentUser;
 final currentUserId = user?.uid ?? '';
 List<Event> events = querySnapshot.docs.map((doc) {
   return Event.fromMap(doc.data() as Map<String, dynamic>, doc.id);
 }).toList();
 List<Event> myEvents = [];
 List<Event> otherEvents = [];
 for (var event in events) {
   try {
     String eventDateTimeString = event.time != null && event.time.isNotEmpty
         ? "${event.date} ${event.time}"
         : "${event.date} 00:00:00";
     DateTime eventDateTime = DateTime.parse(eventDateTimeString);
     print("Event: ${event.name}, DateTime: $eventDateTime");
     if (eventDateTime.isAfter(now)) {
       if (event.creatorId == currentUserId) {
         myEvents.add(event);
       } else {
         otherEvents.add(event);
   } catch (e) {
     print('Error parsing date for event: ${event.name}, error: $e');
 myEvents.sort((a, b) => compareEventsByDate(a, b));
 otherEvents.sort((a, b) => compareEventsByDate(a, b));
 return {
    'myEvents': myEvents,
    'otherEvents': otherEvents,
```

Add Event Menu



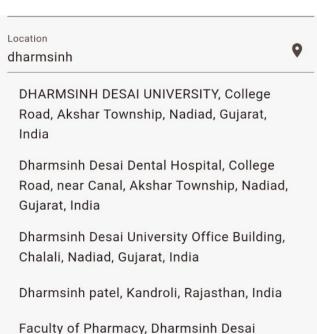
```
Future<void> _submitEvent() async {
 if ( formKey.currentState?.validate() ?? false) {
   try {
     final user = FirebaseAuth.instance.currentUser;
     String creatorId = user?.uid ?? '';
     List<String> uploadedImageUrls =
         await uploadImagesToFirebase( imagePaths);
     final newEvent = {
        'name': nameController.text,
        'location': _locationController.text,
        'date': _dateController.text,
        'time': _timeController.text,
        'guest': _guestController.text,
        'description': _descriptionController.text,
        'fees': feesController.text,
        'type': _selectedType,
        'imagePaths': uploadedImageUrls,
        'creatorId': creatorId,
     await FirebaseFirestore.instance.collection('events').add(newEvent);
     ScaffoldMessenger.of(context).showSnackBar(
       const SnackBar(content: Text('Event added successfully')),
     Navigator.pushReplacement(
       context,
       MaterialPageRoute(
         builder: (context) => const MainScreen(
             currentIndex: 1), // MainScreen
       ), // MaterialPageRoute
   } catch (e) {
     ScaffoldMessenger.of(context).showSnackBar(
       SnackBar(content: Text('Failed to add event: $e')),
```

Add Event Menu (Location)



```
Future<void> _getPlacePredictions(String input) async {
   ApiServices apiService = ApiServices();
   GetPlaces places =
        await apiService.getPlaces(input);
   setState(() {
      _placePredictions = places.predictions ?? [];
   print('Error fetching place predictions: $e');
Codeium: Refactor | Explain | Generate Function Comment | X
Future<void> _selectPlace(String placeId) async {
 final String requestUrl =
      'https://maps.googleapis.com/maps/api/place/details/json?place_id=$placeId&key=$_googleApiKey'
  final response = await http.get(Uri.parse(requestUrl));
 if (response.statusCode == 200) {
   final responseBody = json.decode(response.body);
```

Api Behind the scenes



University, College Road, near Kokran hanuman mandir, Nadiad, Gujarat, India

```
Future < GetPlaces > getPlaces (String placeName) async {
  final Uri url =
      Uri.parse('$ baseUrl/autocomplete/json?input=$placeName&key=$ gcpKey');
  try {
    final response = await http.get(url);
    if (response.statusCode == 200) {
      return GetPlaces.fromJson(jsonDecode(response.body));
    } else {
      throw Exception('Failed to load place predictions');
   catch (error) {
    throw Exception('API ERROR: $error');
```

Event Details Page





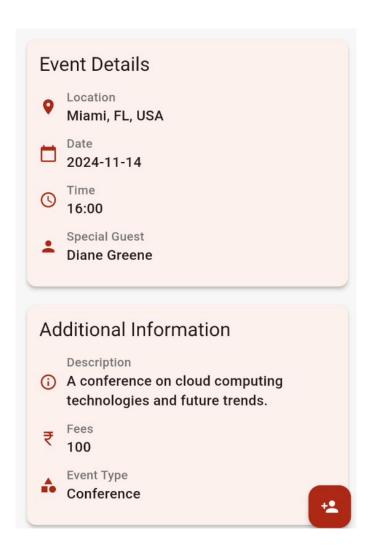






```
Widget buildImageGallery() {
  return SizedBox(
    height: 80,
    child: ListView.builder(
      scrollDirection: Axis.horizontal,
      itemCount: widget.event.imagePaths.length,
      itemBuilder: (context, index) {
        final imagePath = widget.event.imagePaths[index];
        return GestureDetector(
          onTap: () {
            setState(() {
              mainImagePath = imagePath;
          child: Container(
            margin: const EdgeInsets.only(right: 8.0),
            width: 80,
            decoration: BoxDecoration(
             border: Border.all(
               color: mainImagePath == imagePath
                    ? Theme.of(context).primaryColor
                    : □Colors.transparent,
                width: 2,
              borderRadius: BorderRadius.circular(8),
            ), // BoxDecoration
            child: ClipRRect(
             borderRadius: BorderRadius.circular(6),
              child: Image.network(
                imagePath,
                fit: BoxFit.cover,
                errorBuilder: (context, error, stackTrace) {
                 return const Center(child: Text('Image not found'));
```

Event Details Page



```
Widget _buildEventDetails() {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      buildDetailCard('Event Details', [
        DetailItem(
            icon: Icons.location on,
            label: 'Location',
            value: widget.event.location), // DetailItem
        DetailItem(
            icon: Icons.calendar today,
            label: 'Date',
            value: widget.event.date), // DetailItem
        if (widget.event.time.isNotEmpty)
          DetailItem(
              icon: Icons.access time,
              label: 'Time',
              value: widget.event.time), // DetailItem
        if (widget.event.guest.isNotEmpty)
          DetailItem(
              icon: Icons.person,
              label: 'Special Guest',
              value: widget.event.guest), // DetailItem
      const SizedBox(height: 16),
      buildDetailCard('Additional Information', [
        if (widget.event.description.isNotEmpty)
          DetailItem(
              icon: Icons.info outline,
              label: 'Description',
              value: widget.event.description), // DetailItem
        DetailItem(
            icon: Icons.currency rupee,
            label: 'Fees',
            value: widget.event.fees), // DetailItem
        if (widget.event.type != null)
          DetailItem(
              icon: Icons.category,
              label: 'Event Type',
              value: widget.event.type!), // DetailItem
```

Edit Event

← Edit Event

Event Name Cloud Computing Conference Location Miami, FL, USA Date 2024-11-14 Time 16:00 Special Guest Diane Greene Description A conference on cloud computing technologies an Fees 100 Event Type Conference **Update Event**

```
Future<void> _updateEvent() async {
  if (_formKey.currentState!.validate()) {
    try {
      final updatedEvent = {
        'name': nameController.text,
        'location': _locationController.text,
        'date': _dateController.text,
        'time': timeController.text,
        'guest': _guestController.text,
        'description': _descriptionController.text,
        'fees': _feesController.text,
        'type': _selectedType,
      };
      await FirebaseFirestore.instance
          .collection('events')
          .doc(widget.event.id)
          .update(updatedEvent);
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Event updated successfully')),
      );
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => const MainScreen(
              currentIndex: 1), // MainScreen
        ), // MaterialPageRoute
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Failed to update event: $e')),
      );
```

Delete Event

Delete Event

Are you sure you want to delete this event?
This action cannot be undone.

Cancel D

Delete

```
Future<void> _confirmDeleteEvent() async {
 final confirmation = await showDialog<bool>(
   context: context,
   builder: (context) {
     return AlertDialog(
       title: const Text('Delete Event'),
       content: const Text(
           'Are you sure you want to delete this event? This action cannot be undone.'), // Text
       actions: [
         TextButton(
           onPressed: () => Navigator.of(context).pop(false), // Cancel
           child: const Text('Cancel'),
        TextButton(
          onPressed: () => Navigator.of(context).pop(true), // Confirm
          child: const Text('Delete', style: TextStyle(color: ■Colors.red)),
     ); // AlertDialog
 if (confirmation == true) {
   // Proceed with deletion
   await _deleteEvent();
Future<void> _deleteEvent() async {
    await FirebaseFirestore.instance
        .collection('events')
        .doc(widget.event.id)
        .delete();
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Event deleted successfully')),
   Navigator.pushReplacement(
      context,
      MaterialPageRoute(
        builder: (context) => const MainScreen(
            currentIndex: 1), // MainScreen // Navigate to MainScreen with currentIndex
      ), // MaterialPageRoute
 } catch (e) {
   ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(content: Text('Failed to delete event: $e')),
```

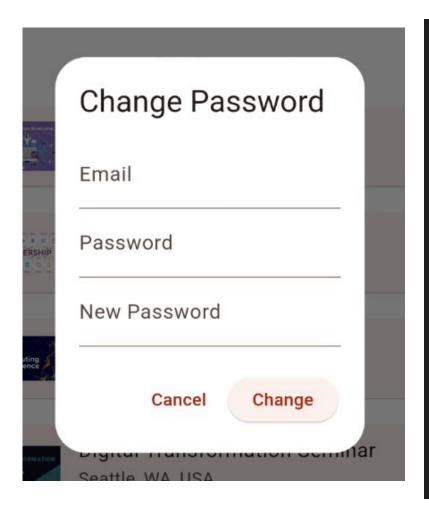
Account & My Events Summary

Account Welcome, rudra@gmail.com **Change Password** Log Out My Events Data Science Bootcamp Chicago, IL, USA Leadership Skills Workshop Boston, MA, USA Cloud Computing Conference Miami, FL, USA **Digital Transformation Seminar**

Seattle, WA, USA

```
class _AccountPageState extends State<AccountPage> {
  late User? user;
  List<Event> _myEvents = [];
  @override
  Codeium: Refactor | Explain | Generate Function Comment | X
  void initState() {
    super.initState();
    user = FirebaseAuth.instance.currentUser;
    _fetchMyEvents();
  Codeium: Refactor | Explain | Generate Function Comment | X
  Future<void> _fetchMyEvents() async {
    if (user != null) {
      QuerySnapshot querySnapshot = await FirebaseFirestore.instance
          .collection('events')
          .where('creatorId', isEqualTo: user!.uid)
          .get();
      List<Event> events = querySnapshot.docs.map((doc) {
        return Event.fromMap(doc.data() as Map<String, dynamic>, doc.id);
      }).toList();
      setState(() {
        _myEvents = events;
      });
```

Credential Change



```
ElevatedButton(
 onPressed: () async {
   String email = emailController.text;
   String password = passwordController.text;
   String newPassword = newPasswordController.text;
   if (newPassword.isNotEmpty &&
       email.isNotEmpty &&
       password.isNotEmpty) {
     try {
       UserCredential userCredential =
           await FirebaseAuth.instance.signInWithEmailAndPassword(
         email: email,
         password: password,
       await userCredential.user?.updatePassword(newPassword);
       ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(
             content: Text('Password updated successfully')), // SnackBar
       Navigator.of(context).pop();
     } catch (e) {
       ScaffoldMessenger.of(context).showSnackBar(
         SnackBar(content: Text('Error: $e')),
 child: const Text('Change'),
   // ElevatedButton
```

Thank You!