EVENT SYNC APP DOCUMENTATION

DEVELOPMENT OF MOBILE APPLICATIONS (EMA)

HOCHSCHULE WORMS - DEPARTMENT OF COMPUTER SCIENCE

TABLE OF CONTENTS

1. Introduction
2. Project Overview
3. Requirements Traceability
4. User Stories
5. Wireframes
6. Implementation Details
7. Code Quality
8. Maintainability
9. Development State and Future Plans
10. Conclusion

1. INTRODUCTION

This document provides an overview of the development and testing process for the mobile application developed as part of the "Entwicklung mobiler Anwendungen (EmA)" course at Hochschule Worms. The application is designed to enhance event planning and social interaction through features such as event notifications, group photo prompts, and social media integration. The app is developed using Google Flutter and integrates with Google Firebase for backend services.

2. PROJECT OVERVIEW

- **Project Name:** Event Sync
- **Course:** Development of Mobile Applications (EmA)
- **Institution:** Hochschule Worms
- **Instructor:** Professor Dr. Stephan Kurpjuweit
- **Developers:** Christian Chin (Matricule: 679605), Rassan Koshnau (Matricule: 679617), (Baran Dennis Koyun Matricule: 679620)

2.1 PROJECT IDEA

EventSync allows users to share their event experiences instantly on popular social media platforms and collaborate on event planning with friends.

The app integrates with Facebook, Instagram, and Twitter for posting updates, photos, and event highlights.

Users can also create event groups, add friends to a "Social Circle," receive notifications, and discuss event details collaboratively.

3. REQUIREMENTS TRACEABILITY

3.1 FUNCTIONAL REQUIREMENTS

1. **Multiple Screens:**
    - Onboarding screen
    - Splash screen 1/2/3
    - Welcome screen
    - Registration/Login screen
    - Forget Password Screen
    - Mail Verification screen
    - Home screen with list of events
    - Event detail screen with social media sharing options
    - Group creation and management screen

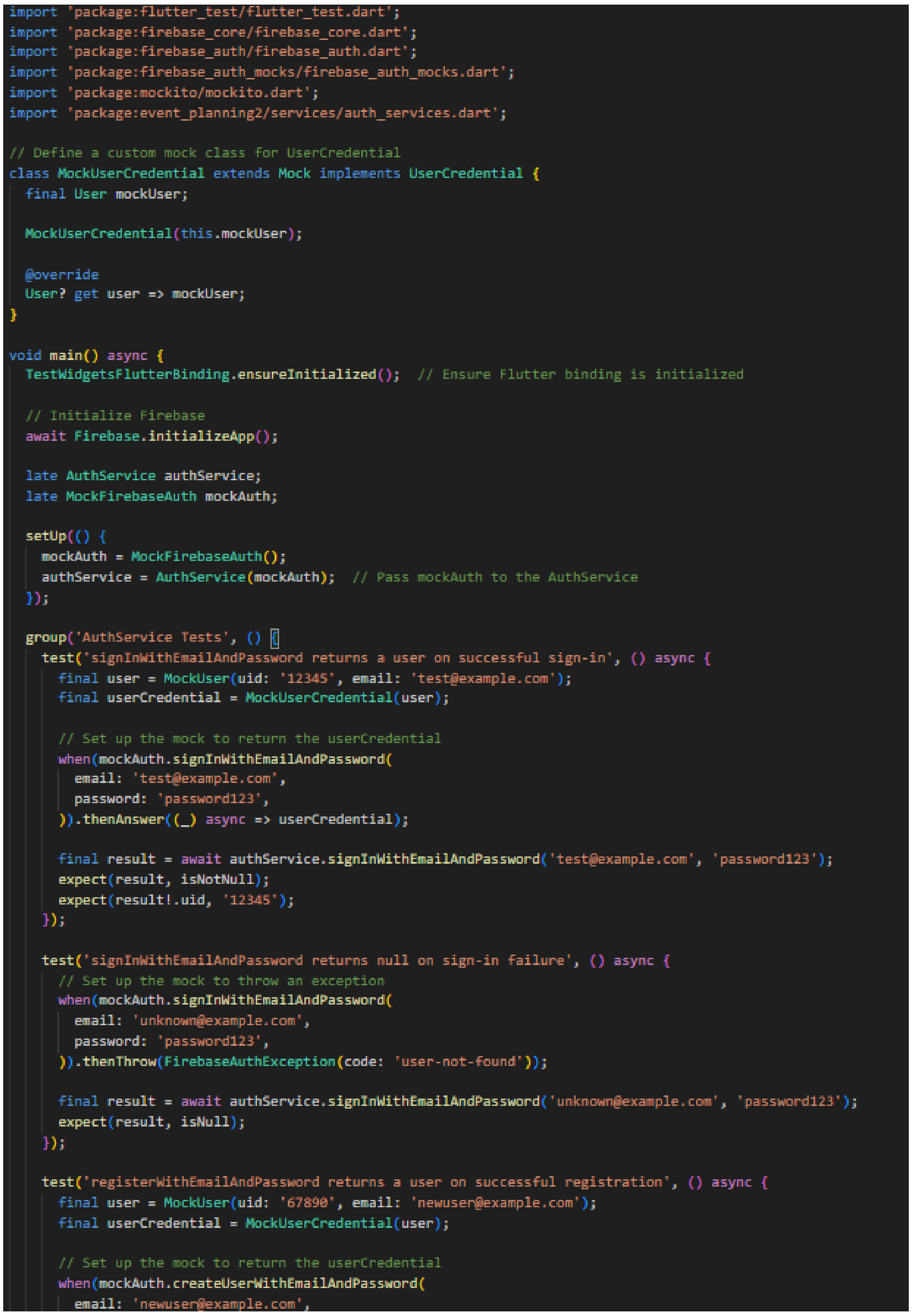
2. **User Generated Content:**
    - Users can create, edit, and delete events.
    - Users can post updates, photos, and highlights.

3. **Local Data Storage:**
    - Events and user data are stored locally for offline access.

4. **Google Firebase Integration:**
    - Firebase Authentication for user registration and login.
    - Firebase Firestore for storing event details and user groups.

5. **Unit Test:**

- Example unit test for event creation functionality.



```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_auth_mocks/firebase_auth_mocks.dart';
import 'package:mockito/mockito.dart';
import 'package:event_planning2/services/auth_services.dart';

// Define a custom mock class for UserCredential
class MockUserCredential extends Mock implements UserCredential {
  final User mockUser;

  MockUserCredential(this.mockUser);

  @override
  User? get user => mockUser;
}

void main() async {
  TestWidgetsFlutterBinding.ensureInitialized();  // Ensure Flutter binding is initialized

  // Initialize Firebase
  await Firebase.initializeApp();

  late AuthService authService;
  late MockFirebaseAuth mockAuth;

  setUp(() {
    mockAuth = MockFirebaseAuth();
    authService = AuthService(mockAuth);  // Pass mockAuth to the AuthService
  });

  group('AuthService Tests', () {
    test('signInWithEmailAndPassword returns a user on successful sign-in', () async {
      final user = MockUser(uid: '12345', email: 'test@example.com');
      final userCredential = MockUserCredential(user);

      // Set up the mock to return the userCredential
      when(mockAuth.signInWithEmailAndPassword(
        email: 'test@example.com',
        password: 'password123',
      )).thenAnswer((_) async => userCredential);

      final result = await authService.signInWithEmailAndPassword('test@example.com', 'password123');
      expect(result, isNotNull);
      expect(result!.uid, '12345');
    });

    test('signInWithEmailAndPassword returns null on sign-in failure', () async {
      // Set up the mock to throw an exception
      when(mockAuth.signInWithEmailAndPassword(
        email: 'unknown@example.com',
        password: 'password123',
      )).thenThrow(FirebaseAuthException(code: 'user-not-found'));

      final result = await authService.signInWithEmailAndPassword('unknown@example.com', 'password123');
      expect(result, isNull);
    });

    test('registerWithEmailAndPassword returns a user on successful registration', () async {
      final user = MockUser(uid: '67890', email: 'newuser@example.com');
      final userCredential = MockUserCredential(user);

      // Set up the mock to return the userCredential
      when(mockAuth.createUserWithEmailAndPassword(
        email: 'newuser@example.com',
```

6. **Notification:**
    - Push up Notification for new content.

4. USER STORIES

1. GROUP PHOTO NOTIFICATIONS

As a memory enthusiast, I want to receive notifications during events for group photos, so that we can capture shared memories effortlessly.

Description:

- The app prompts us to take group photos during events.
- Photos are saved within the event details.
- We can easily share these photos with fellow attendees.

2. SOCIAL MEDIA SHARING

As a social media enthusiast, I want to share my event experiences instantly on social platforms, so that my friends and followers can join the fun virtually.

Description:

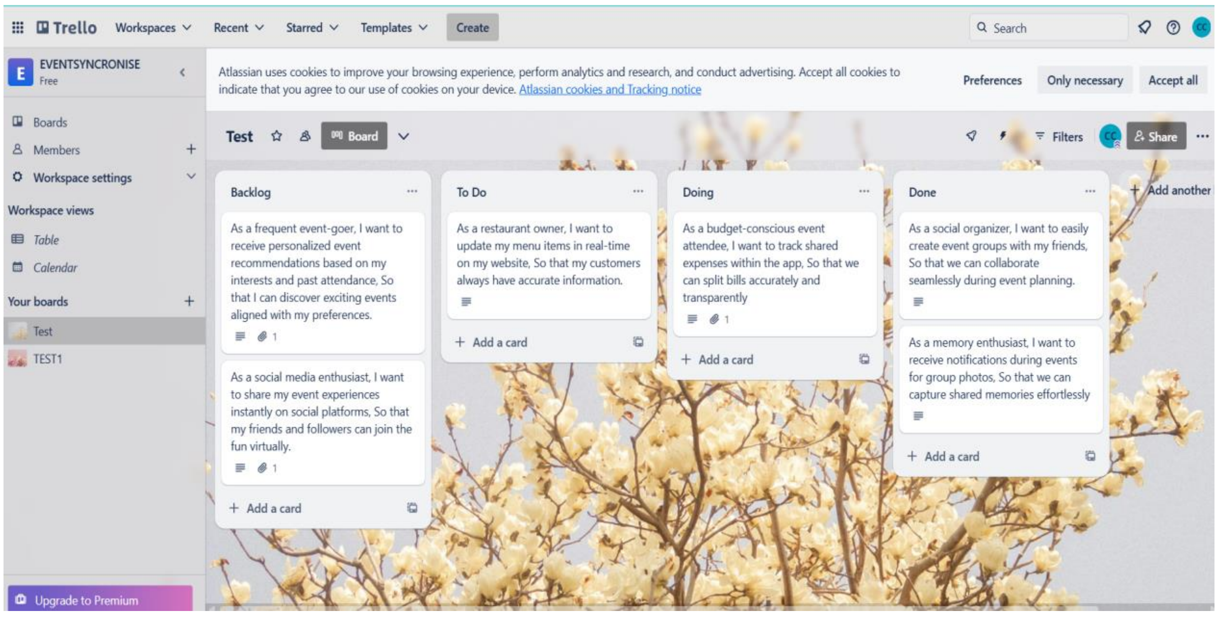
- The app integrates with popular social media platforms (Facebook, Instagram, Twitter).
- I can post updates, photos, and event highlights directly from the app.

3. EVENT GROUP CREATION

As a social organizer, I want to easily create event groups with my friends, so that we can collaborate seamlessly during event planning.

Description:

- I can add friends to a “Social Circle” within the app.
- Group members receive notifications about upcoming events.
- We can discuss and decide on event details together.



Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. Atlassian cookies and Tracking notice

Backlog

- As a frequent event-goer, I want to receive personalized event recommendations based on my interests and past attendance, So that I can discover exciting events aligned with my preferences.
- As a social media enthusiast, I want to share my event experiences instantly on social platforms, So that my friends and followers can join the fun virtually.

+ Add a card

To Do

- As a restaurant owner, I want to update my menu items in real-time on my website, So that my customers always have accurate information.

+ Add a card

Doing

- As a budget-conscious event attendee, I want to track shared expenses within the app, So that we can split bills accurately and transparently

+ Add a card

Done

- As a social organizer, I want to easily create event groups with my friends, So that we can collaborate seamlessly during event planning.
- As a memory enthusiast, I want to receive notifications during events for group photos, So that we can capture shared memories effortlessly

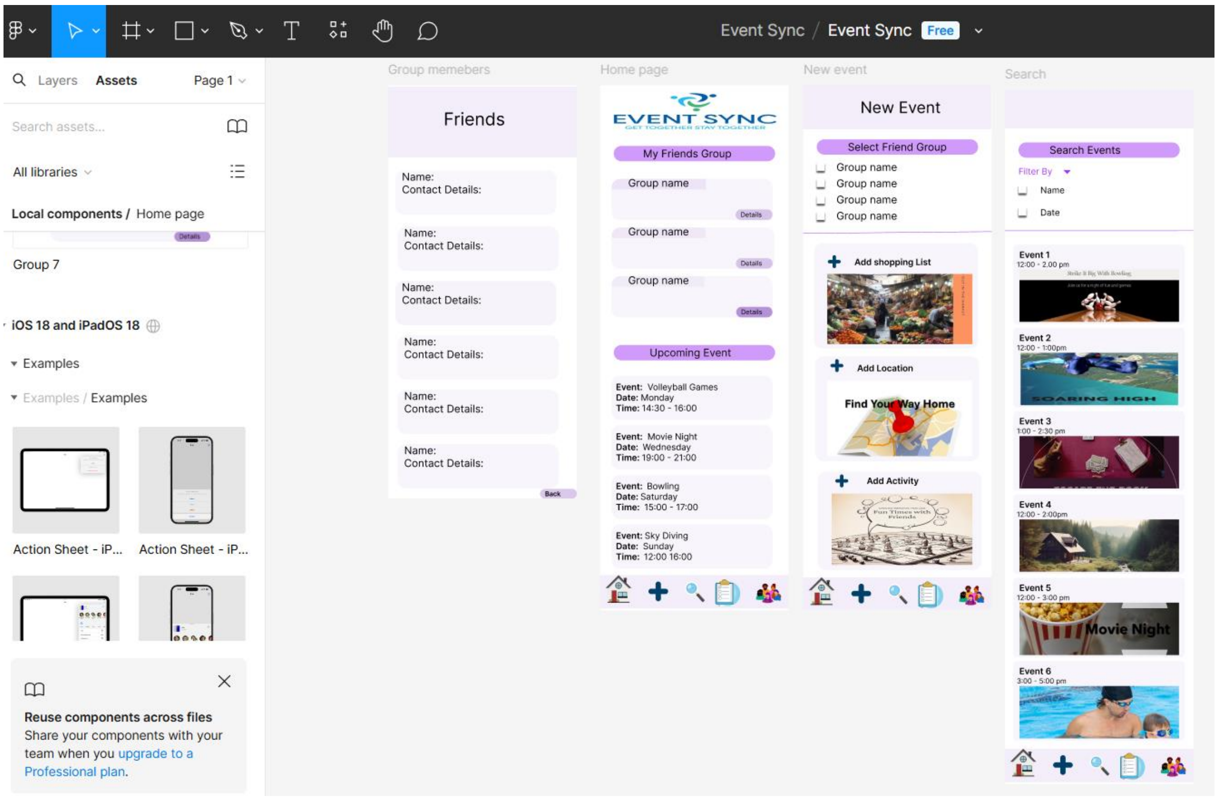
+ Add a card

5. WIREFRAMES AND DESIGN EVOLUTION

WIREFRAMES

The initial design of the application was created using Figma. Below are some screenshots of the wireframes that outline the basic structure and user interface elements of the app.

FIGURE 1: WIREFRAME OF THE DIFFERENT SCREENS.

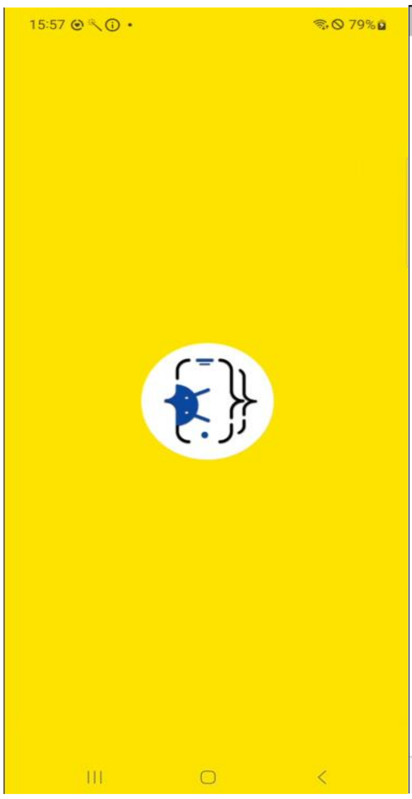


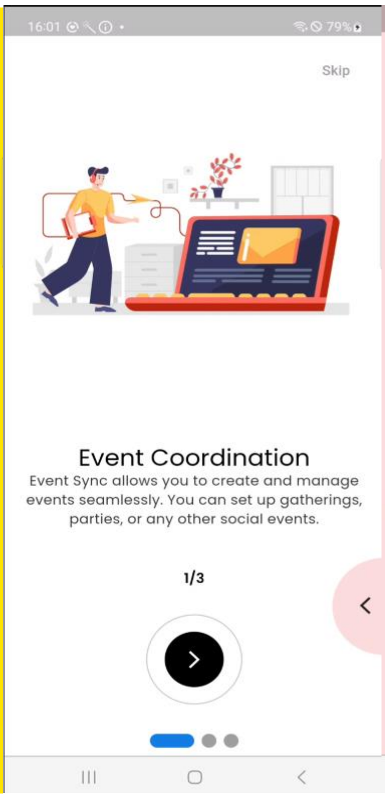
DESIGN EVOLUTION

During the development process, we deviated from the initial wireframes to better meet the user requirements and improve the overall user experience. Changes included:

- Simplification of the navigation structure.
- Enhanced visual aesthetics based on user feedback.
- Improved accessibility features.
- Additional functionalities that were not part of the initial wireframes.

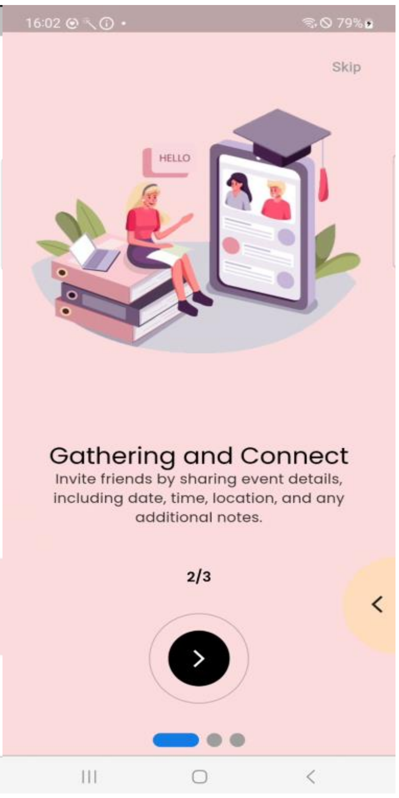
We would like to provide you with sample screens of our current app to show the actual implementation and how it differs from our initial wireframes.





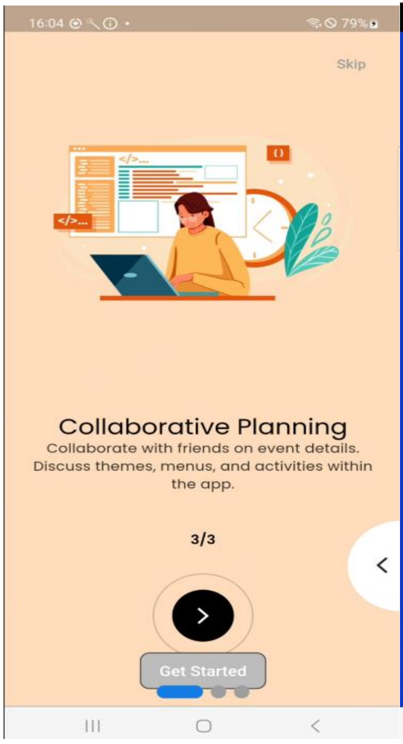
Event Coordination

Event Sync allows you to create and manage events seamlessly. You can set up gatherings, parties, or any other social events.



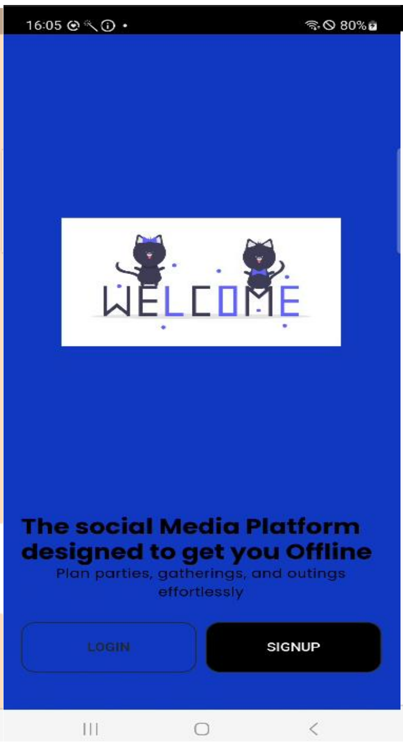
Gathering and Connect

Invite friends by sharing event details, including date, time, location, and any additional notes.



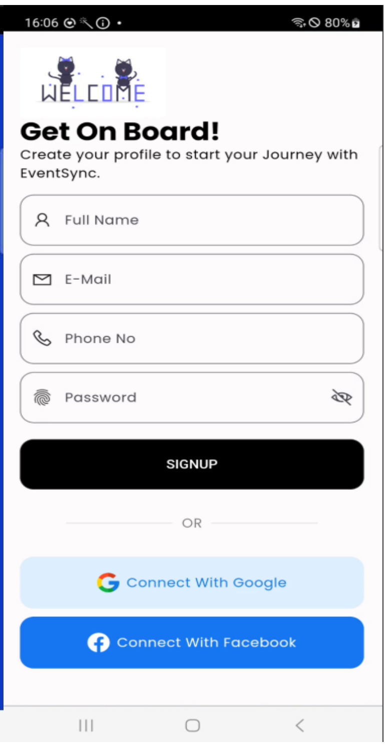
Collaborative Planning

Collaborate with friends on event details. Discuss themes, menus, and activities within the app.



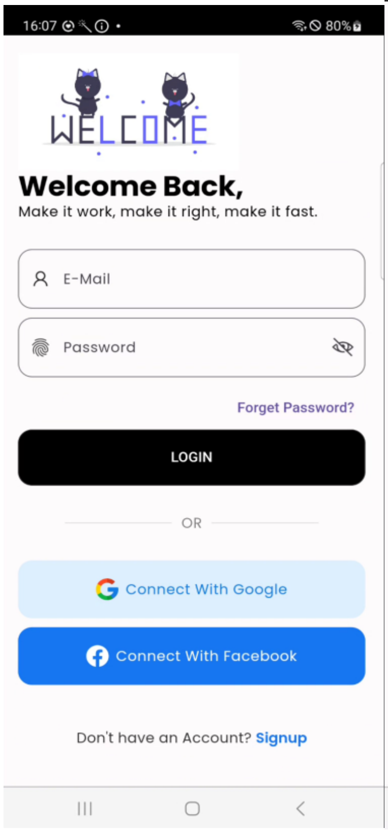
The social Media Platform designed to get you Offline

Plan parties, gatherings, and outings effortlessly



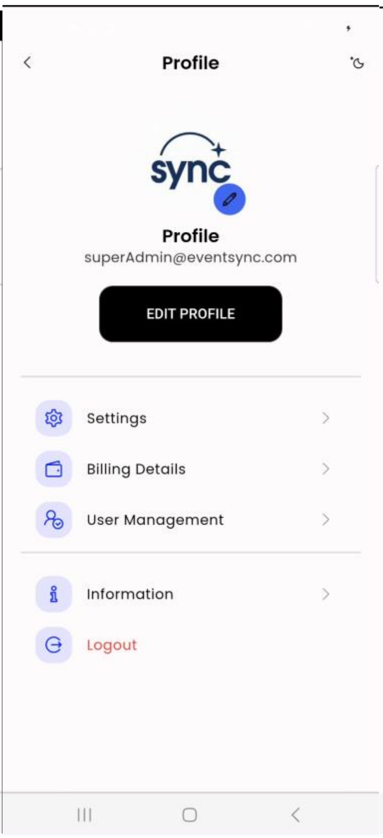
Get On Board!

Create your profile to start your Journey with EventSync.



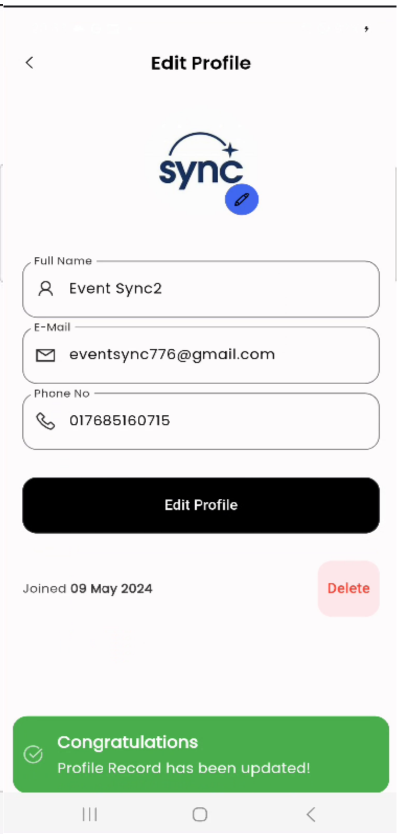
Welcome Back,

Make it work, make it right, make it fast.



Profile

superAdmin@eventsync.com



Full Name: Event Sync2

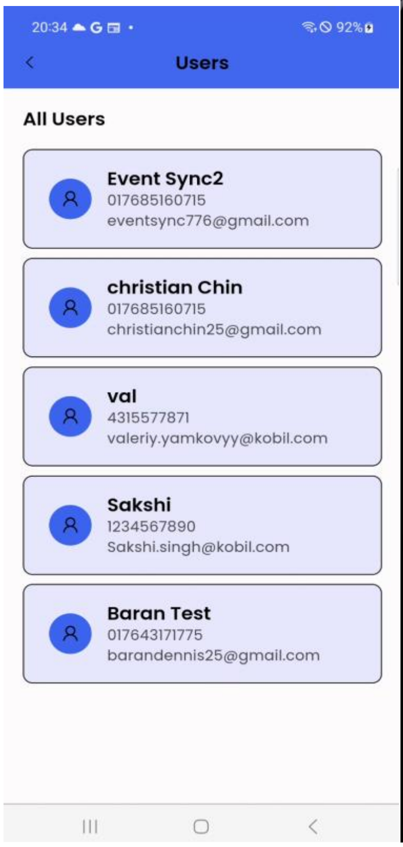
E-Mail: eventsync776@gmail.com

Phone No: 017685160715

Joined 09 May 2024

Congratulations

Profile Record has been updated!



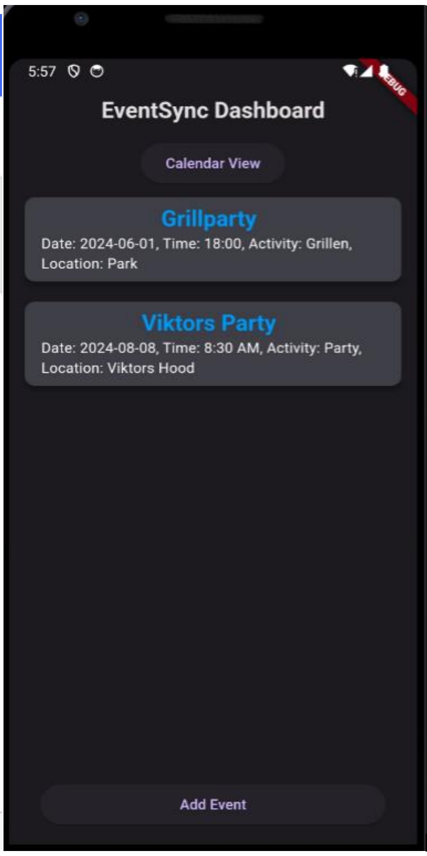
Event Sync2
017685160715
eventsync776@gmail.com

christian Chin
017685160715
christianchin25@gmail.com

val
4315577871
valeriy.yamkovyy@kobil.com

Sakshi
1234567890
Sakshi.singh@kobil.com

Baran Test
017643171775
barandennis25@gmail.com

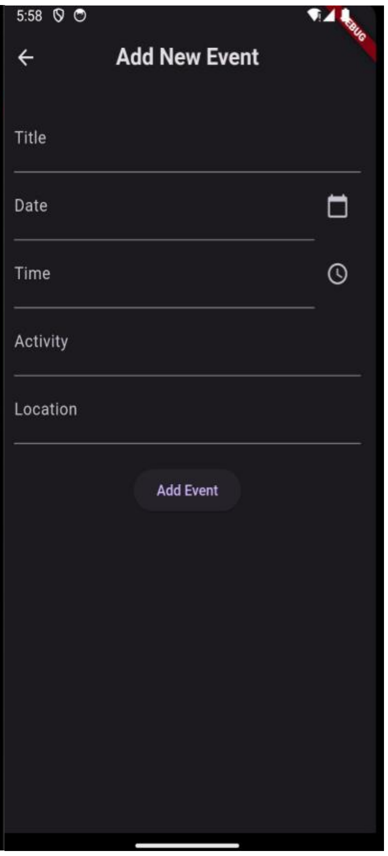


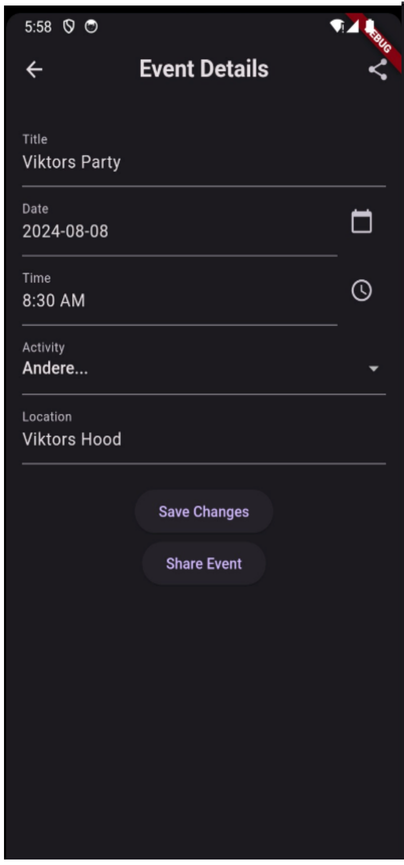
Grillparty

Date: 2024-06-01, Time: 18:00, Activity: Grillen, Location: Park

Viktors Party

Date: 2024-08-08, Time: 8:30 AM, Activity: Party, Location: Viktors Hood




5:58

Event Details

Title
Viktors Party

Date
2024-08-08

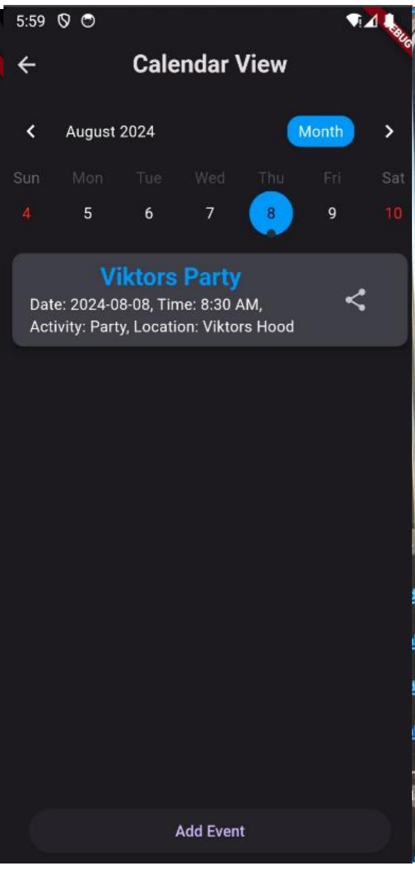
Time
8:30 AM

Activity
Andere...

Location
Viktors Hood

Save Changes

Share Event


5:59

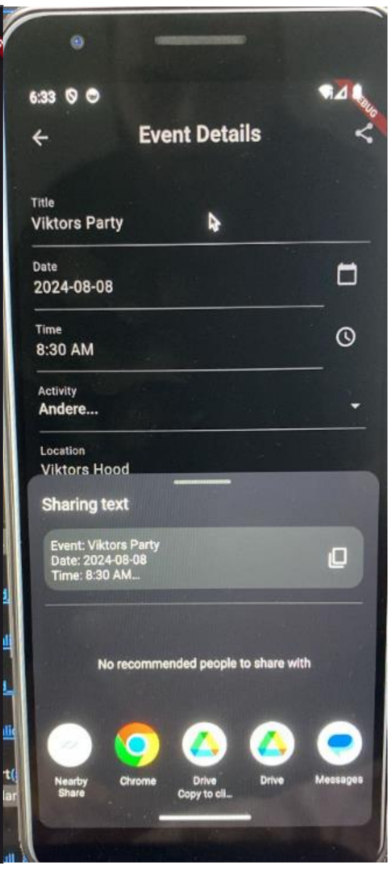
Calendar View

August 2024 Month

Sun	Mon	Tue	Wed	Thu	Fri	Sat
4	5	6	7	8	9	10

Viktors Party
Date: 2024-08-08, Time: 8:30 AM,
Activity: Party, Location: Viktors Hood

Add Event


6:33

Event Details

Title
Viktors Party

Date
2024-08-08

Time
8:30 AM

Activity
Andere...

Location
Viktors Hood

Sharing text

Event: Viktors Party
Date: 2024-08-08
Time: 8:30 AM...

No recommended people to share with

3.2 ADD-ONS

1. Responsive Layout:
   - Supports both portrait and landscape modes.
   - Optimized for tablets and smartphones.

2. Crashlytics/Analytics:
   - Integration with Firebase Crashlytics for error tracking.
   - Firebase Analytics for user behavior tracking.(in Progress)

3. Background Task:
   - Notifications for group discussions and event updates.

4. Offline Mode:
   - Full functionality available offline with data syncing once back online.(in Progress)

5. **Animations:**
    - Smooth transitions between screens.

6. **Admin Interface:**
    - Admin dashboard for managing user content (in progress).

7. **Integration of APIs:**
    - Integration with Facebook, Instagram, and Twitter APIs for social media sharing.

8. **Search:**
    - Search functionality to find specific events or groups.

9. **Use of Camera/Sensors:**
    - Use of device camera to take photos for event highlights.(in Progress)

6. IMPLEMENTATION DETAILS

4.1 ARCHITECTURE

- **Pattern:** MVVM (Model-View-ViewModel)
- **Framework:** Flutter
- **Backend:** Firebase (Authentication, Firestore, Storage)

4.2 KEY MODULES

1. **Authentication Module:**
    - Registration and login using Firebase Auth.
2. **Event Module:**
    - Create, edit, delete, and view events.
    - Post updates, photos, and highlights.
3. **Social Sharing Module:**
    - Share posts on Facebook, Instagram, and Twitter.
4. **Group Module:**
    - Create and manage event groups.
    - Add friends to "Social Circle" and manage notifications.
5. **Notification Module:**
    - Sends notifications for group discussions and upcoming events.

4.3 DATA FLOW

1. **User Authentication:**
    - Users register and login via Firebase Auth.
2. **Data Storage:**
    - Events and groups stored in Firebase Firestore.
    - Photos stored in Firebase Storage.
3. **Local Storage:**
    - Use of SQLite for offline data access and syncing.(in Progress)

7. CODE QUALITY

5.1 CODE STANDARDS

- **Language:** Dart
- **Style Guide:** Effective Dart
- **Linting:** Dart Analysis configured to enforce code standards.

5.2 TESTING

- **Unit Testing:** Use of Flutter's built-in testing framework.
- **Integration Testing:** Testing interaction between components.(in Progress)
- **End-to-End Testing:** Integration with Flutter Driver for E2E tests.(in Progress)

5.3 CODE REVIEW

- Regular peer reviews.
- Checklist for coding standards, unit tests, and security checks.

8. MAINTAINABILITY

6.1 DOCUMENTATION

- **Code Documentation:** Inline comments and Dartdoc.

6.2 VERSION CONTROL

- **Repository:** GitHub
- **Branching Strategy:** GitFlow
- **Commit Message Convention:** Conventional Commits.

6.3 DEPENDENCY MANAGEMENT

- **Package Management:** Pub.dev for Flutter packages.
- Regular updates and security audits using pub outdated.

9. DEVELOPMENT STATE AND FUTURE PLANS

7.1 CURRENT DEVELOPMENT STATE

- **Completed Features:** User authentication, event management, social media sharing, group creation and management, notifications.
- **In-Progress Features:** Offline mode, admin interface, and advanced search capabilities.

7.2 UPCOMING PLANS

- **New Features:** Integration with additional social media platforms, enhanced group collaboration tools, and multi-language support.
- **Improvements:** UI/UX enhancements, performance optimizations, and additional language support.
- **Bug Fixes:** Address user-reported bugs and issues found during testing.

10. CONCLUSION

This document provides a detailed overview of the Event sync app, ensuring a comprehensive understanding of its features, implementation, and future plans. The app meets the course requirements and is built to be scalable and maintainable for future enhancements.