Markme

BACHELOR OF

TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

BY

Pavan Kumar Chennupati	**(22501A05E0)**
Vignesh Varanasi	**(22501A05J1)**
Shaik Fakruddin	**(22501A05G1)**
Y. Eswar Aditya	**(22501A05J8)**

Under the Guidance of

Mr. Michael Sadgun Rao Kona,

Assistant Professor

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU: Kakinada, Approved by AICTE)

(An NBA & NAAC A++ accredited and ISO 9001:2015 certified institution)

Kanuru, Vijayawada-520007

2024-25

PRASAD V POTLURI

SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU :: Kakinada, Approved by AICTE)

(An NBA & NAAC A++ accredited and ISO 9001:2015 certified institution)

Kanuru, Vijayawada – 520007

CERTIFICATE

This is to certify that the project report title **"Markme"** is the bonafide work of **Pavan Kumar Chennupati (22501A05E0), Vignesh Varanasi (22501A05J1), Shaik Faruddin (22501A05G1), Yarlagadda Eswar Aditya (22501A05J8)** in partial fulfilment of completing the Academic project in Mobile App Development (20SA8651) during the academic year 2024-25.

Signature of the Incharge **Signature of the HOD**

INDEX

1.ABSTRACT

Mark Me is a mobile application designed to streamline event planning and management for college communities. It provides a structured platform for organizing, discovering, and participating in events efficiently. The app enables users to create Spaces, where multiple events can be hosted, allowing better categorization and collaboration among organizers.

Users can follow Spaces to stay updated on upcoming events and receive notifications. Event organizers can manage event visibility and set participant limits. Participants can "Mark Me" to register for events, ensuring seamless engagement. The app supports event states like Upcoming, Live, On-Hold and Archived, helping both organizers and attendees track event progress effectively.

Built with React Native, Mark Me ensures a smooth, cross-platform mobile experience. The backend, powered by Node.js and Express.js, handles event management and user interactions, while MongoDB efficiently stores event data and participant information. Real-time updates and notifications are powered by WebSockets, ensuring seamless communication.

By providing a centralized platform for event planning, Mark Me enhances collaboration, boosts student participation, and simplifies event organization, making it an essential tool for college communities.

2. SDG JUSTIFICATION REPORT

SDG Mapped: SDG 8 – Decent Work and Economic Growth

2.1 How This Project Supports SDG 8

Mark Me directly contributes to achieving SDG 8 by fostering skill development, improving event accessibility, and enhancing collaboration within college communities through:

- **Enhanced Student Engagement:** By streamlining event participation, the app encourages students to actively engage in workshops, hackathons, and networking events, fostering career growth and skill development.
- **Opportunities for Professional Development:** Spaces allow student organizations to host industry talks, career fairs, and training sessions, bridging the gap between academia and professional opportunities.
- **Efficient Event Management:** Organizers can efficiently plan and execute events, gaining real-world experience in project management, leadership, and teamwork.
- **Scalability for Broader Impact:** Mark Me can be adopted across multiple institutions, creating a larger ecosystem for learning, professional development, and entrepreneurship opportunities.

3. INTRODUCTION

MarkMe is a mobile-first event management platform designed to streamline event organization, attendee registration, and participation tracking. By providing a structured and intuitive system, MarkMe simplifies the process of hosting and managing events while ensuring seamless engagement between organizers and participants. With a focus on accessibility and efficiency, the platform enhances event experiences through real-time updates, automated workflows, and role-based access control.

MarkMe offers a range of features, including space-based event management, attendee check-ins, role-based administration, and event insights. Organizers can create dedicated Spaces, appoint admins, and manage events within them. Followers of a Space receive notifications about upcoming events and can register easily. Attendees can check in using unique codes, ensuring a secure and verified participation process. Events progress through well-defined states—Upcoming, Live, On-Hold, Ongoing, and Archived—keeping all stakeholders informed at every stage.

The platform consists of multiple sections to enhance user experience. The Home page provides an overview of active Spaces and upcoming events. The Space dashboard allows admins to manage events, send notifications, and monitor participation. The event details page provides event-specific information, including registration options and check-in status. Users can track their past and upcoming events through the Profile section, where they can also adjust notification preferences and manage their Spaces.

With its structured event workflow, real-time engagement features, and user-friendly design, MarkMe empowers both organizers and attendees to create and participate in events effortlessly. By integrating smart event management tools, the platform enhances event accessibility, ensures smooth coordination, and fosters community-driven participation.

4. OBJECTIVES AND SCOPE OF THE PROJECT

Mark Me is designed to simplify event planning and management within college communities by providing a structured and interactive platform for organizers and participants. The project focuses on the following key objectives:

Enhance Event Organization and Planning: Mark Me provides a streamlined system for creating and managing events within Spaces, allowing organizers to categorize and coordinate multiple events efficiently. The platform ensures structured event planning with features like registration management, event status tracking, and customizable participation forms, making event execution smoother.

Boost Student Engagement and Participation: The platform enables students to follow Spaces of interest, receive event notifications, and easily register for events using the "Mark Me" feature. By centralizing event information, the app ensures that students stay informed about upcoming activities, increasing engagement and participation in college events.

Ensure Seamless and Secure Event Management: Mark Me is designed with real-time updates powered by WebSockets, ensuring smooth interactions between organizers and participants. The app provides secure event registrations, participant tracking, and customizable access control, preventing unauthorized participation. Organizers can monitor attendance trends and optimize event management through analytics.

Enable Future Scalability and Integration: The platform is built with scalability in mind, allowing future enhancements such as AI-driven event recommendations, calendar synchronization, integration with college portals, and automated reminders. Features like feedback collection and performance analytics will help organizers improve future events based on participant responses.

Scope of the Project: Mark Me extends beyond a simple event registration tool; it serves as a comprehensive event planning ecosystem for colleges. By facilitating efficient event management, enhancing student engagement, and providing real-time updates, Mark Me empowers both organizers and participants. With its intuitive mobile interface and structured event management system, the platform redefines how college events are planned and executed, making event coordination more interactive, organized, and accessible.

5. SOFTWARE USED – EXPLANATION

Mark Me is developed using modern mobile development technologies to provide a seamless and interactive event management experience. The key technologies used include:

Frontend Technologies:

React Native

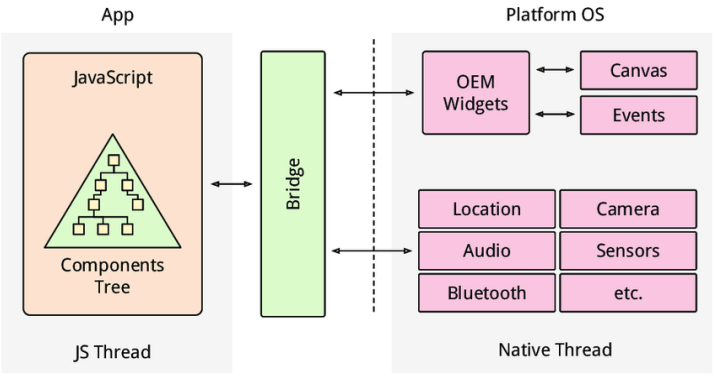
- A cross-platform mobile framework for building native mobile applications.
- Uses a component-based architecture to create reusable UI elements.
- Ensures high performance with native modules and direct API access.
- Enables rapid development with features like hot reloading and fast refresh.

Expo

- A framework and toolchain for developing React Native applications with minimal configuration.
- Provides built-in APIs for device functionalities such as camera, location, and notifications.

TypeScript (TS)

- A statically typed superset of JavaScript that enhances code quality and maintainability.
- Provides type safety, reducing runtime errors and improving developer productivity.
- Supports modern ECMAScript features while ensuring compatibility with JavaScript libraries.
- Enhances readability and scalability, making it ideal for large applications.



Backend Technologies:

Express.js

- A Node.js web framework that handles API interactions, authentication, and data processing.
- Manages server-side logic, routing, and communication between the frontend and database.

Node.js

- A runtime environment that allows JavaScript execution on the server side.
- Supports an event-driven, non-blocking architecture for high-performance.

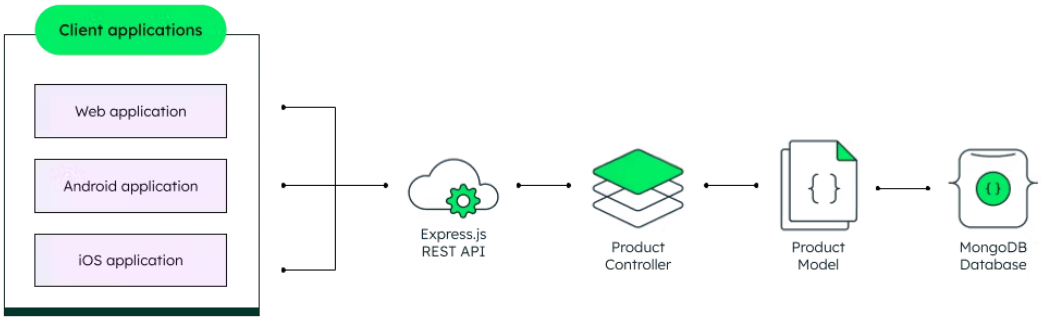
Database and Storage Tools:

MongoDB

- NoSQL database used for storing user data, event details, and media files.
- Provides flexible and scalable data management, making it efficient for handling large datasets.

Mongoose

- An Object Data Modeling (ODM) library for MongoDB in Node.js.
- Simplifies database interactions by providing schema-based data modeling.



By integrating these technologies, Mark Me delivers a native, high-performance mobile experience, ensuring a smooth and structured UI for event planning and management.

6. PROPOSED MODEL

The proposed model for MarkMe aims to create an efficient and user-friendly platform for event registration, attendance tracking, and community engagement. The system is structured into various key components:

Home Page & Navigation: The landing page provides an intuitive interface where users can explore their subscribed Spaces, view upcoming events, and access event management features. The streamlined navigation ensures a smooth user experience, allowing attendees and organizers to interact effortlessly.

Space-Based Event Management: MarkMe introduces Spaces, which serve as dedicated hubs for event hosting. Organizers can create multiple events within a Space, assign administrative roles, and send notifications to followers. Users can subscribe to Spaces to stay updated on upcoming events and announcements.

Event Creation & Management: Organizers can create events within a Space by defining event details such as name, date, location, capacity, and participant permissions. Events follow a structured life cycle, transitioning through states like Upcoming, Live, On-Hold, and Archived, ensuring clear status visibility.

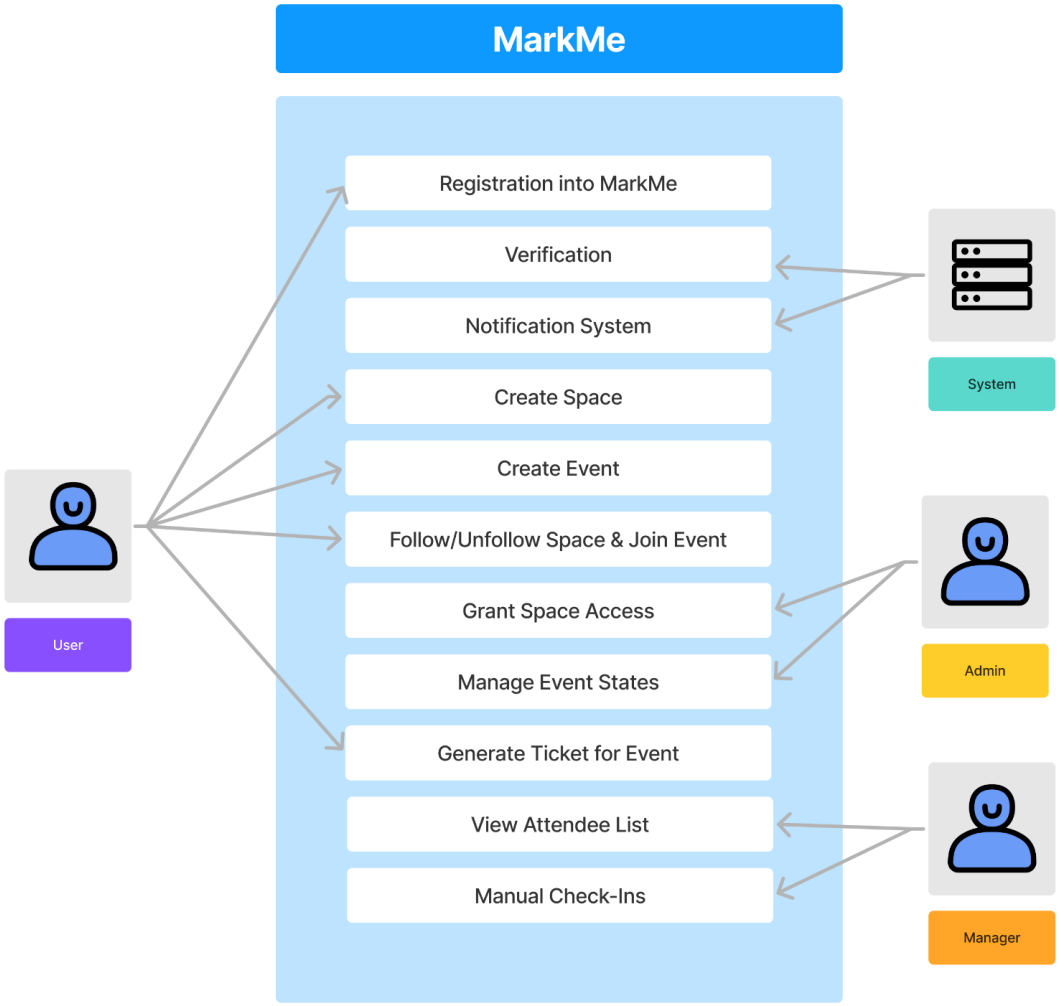
Attendee Registration & Check-In: Participants can register for events and receive a unique check-in code, which is used for seamless on-site or virtual verification. Organizers can track real-time attendance, ensuring a smooth event experience while maintaining role-based access for event management operations.

Event Insights & History: Event organizers can access past event data, including attendance records, engagement metrics, and feedback, to improve future event planning. Archived events remain accessible for reference, ensuring structured event documentation and insights.

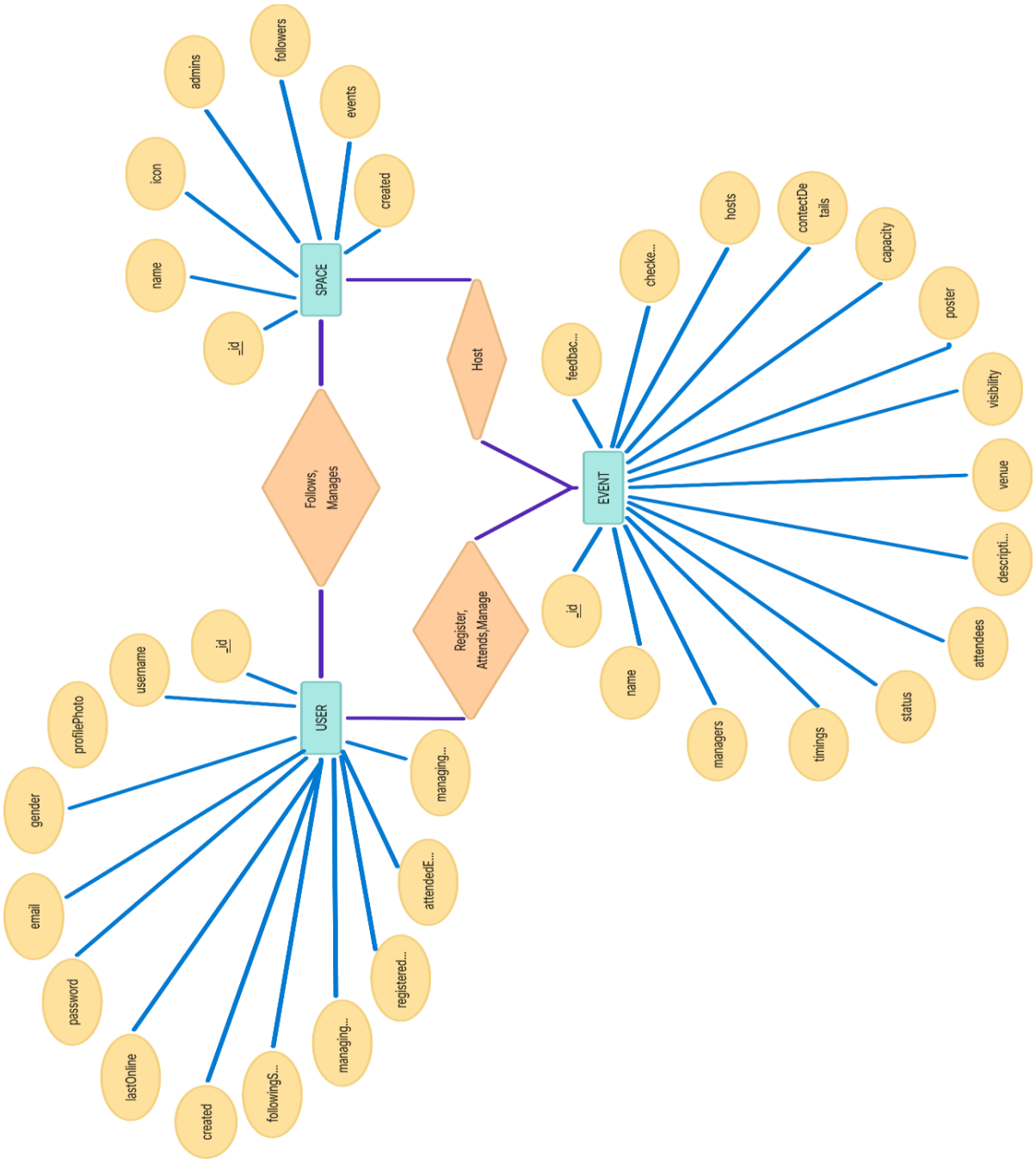
Security & Role-Based Access Control: MarkMe implements role-based access control (RBAC) to ensure that only authorized users can create, manage, and modify events. Secure check-in verification prevents unauthorized access, and system logs track all activities to maintain event integrity.

Responsive Design & Scalability: MarkMe is designed with a mobile-first approach, ensuring a seamless user experience across smartphones, tablets, and desktops. The platform supports high user engagement and event management without performance bottlenecks. Future enhancements could include QR code-based check-ins for faster and contactless attendee verification

Use Case Diagram:



Entity-Relationship (ER) Diagram:



7. SAMPLE CODE

GitHub repository links:

https://github.com/pavancos/markme
https://github.com/vigneshvaranasi/markmeEngine

Folder Structure:

```
Mark Me
├── app
│   ├── index.tsx
│   ├── _layout.tsx
│   ├── (tabs)
│   │   ├── create.tsx
│   │   ├── explore.tsx
│   │   ├── notifications.tsx
│   │   ├── profile.tsx
│   │   ├── settings.tsx
│   │   ├── _layout.tsx
│   │   ├── home
│   │   │   ├── _layout.tsx
│   │   │   ├── screens
│   │   │   │   ├── Past.tsx
│   │   │   │   ├── Upcoming.tsx
│   │   ├── space
│   │   │   ├── [id].tsx
│   ├── auth
│   │   ├── login.tsx
│   │   ├── otp.tsx
│   │   ├── signup.tsx
│   │   ├── _layout.tsx
│
├── components
│   ├── AppHeader.tsx
│   ├── CreateEventSheet.tsx
│   ├── CreateSheet.tsx
│   ├── CreateSpaceSheet.tsx
│   ├── Event.tsx
```

```
│   ├── EventItem.tsx
│   ├── EventPage.tsx
│   ├── EventSheet.tsx
│   ├── EventTextBox.tsx
│   ├── ProfileHeader.tsx
│   ├── SpaceEditSheet.tsx
│   ├── SpaceItem.tsx
│   ├── SpacePicker.tsx
│   ├── TextBox.tsx
│   ├── svgs
│   │   ├── AppHeaderLogo.tsx
│   │   ├── BackNavigation.tsx
│   │   ├── LocationPin.tsx
│   │   ├── PersonGroup.tsx
│   │   ├── ProfileSettings.tsx
│   │   └── SpaceArrow.tsx
│
├── hook
│   ├── useHaptic.tsx
│
├── constants
│   ├── config.ts
│
├── utils
│   ├── dateUtils.ts
│   ├── eventUtils.ts
│
├── stores
│   ├── authStore.ts
│   ├── exploreStore.ts
│   ├── homeStore.ts
│   ├── profileStore.ts
│   ├── sheetStore.ts
│
├── assets
│   ├── fonts
│   │   ├── BasicSans-Regular.ttf
│   │   ├── SpaceMono-Regular.ttf
```

_layout.tsx

```
import { Stack } from "expo-router";
import React, { useEffect, useState } from "react";
import { GestureHandlerRootView } from 'react-native-gesture-handler';
import { toast, Toasts } from '@backpackapp-io/react-native-toast';
import AsyncStorage from "@react-native-async-storage/async-storage";
import EventSheet from "@/components/EventSheet";
import { View } from "react-native";
import CreateSheet from "@/components/CreateSheet";
import CreateEventSheet from "@/components/CreateEventSheet";
import CreateSpaceSheet from "@/components/CreateSpaceSheet";
import SpaceEditSheet from "@/components/SpaceEditSheet";
export default function Layout() {
 const [isLoading, setIsLoading] = useState(true);
 const [isAuthenticated, setIsAuthenticated] = useState(false);
 useEffect(() => {
  const checkToken = async () => {
   const token = await AsyncStorage.getItem("token");
   setIsAuthenticated(!!token);
   setIsLoading(false);
  };
  checkToken();
 }, []);
 if (isLoading) return null;
 return (
  <GestureHandlerRootView style={{ flex: 1 }}>
   <View style={{ flex: 1 }}>
    <Stack screenOptions={{ headerShown: false }}>
     {isAuthenticated ? (
      <Stack.Screen name="(tabs)" options={{ statusBarBackgroundColor: 'black' }} />
     ) : (
      <Stack.Screen name="auth" options={{ statusBarBackgroundColor: 'black' }} />
     )}
    </Stack>
   </View>
   <EventSheet />
   <CreateSheet />
   <CreateEventSheet />
```

```
      <CreateSpaceSheet />
      <SpaceEditSheet/>
      <Toasts
       defaultStyle={{
         view: {
          backgroundColor: "rgba(44, 44, 46, 0.85)",
          borderRadius: 12,
          shadowColor: "#000",
          shadowOffset: { width: 0, height: 4 },
          shadowOpacity: 0.4,
          shadowRadius: 6,
          padding: 14,
         },
         pressable: {
          backgroundColor: "rgba(44, 44, 46, 0.85)",
          borderRadius: 16,
          shadowColor: "#000",
          shadowOffset: { width: 0, height: 4 },
          shadowOpacity: 0.4,
          shadowRadius: 6,
         },
         text: {
          color: "#fff",
          fontSize: 16,
          fontWeight: "500",
         },
       }}
      />
    </GestureHandlerRootView>
  );
}
```

index.tsx

```
import { useCallback, useEffect, useState } from "react";
import { Stack, useRouter } from "expo-router";
import AsyncStorage from "@react-native-async-storage/async-storage";
import { View, ActivityIndicator } from "react-native";
import * as SplashScreen from 'expo-splash-screen';
import { useFonts } from "expo-font";
import TextBox from "@/components/TextBox";
import { useAuthStore } from "@/stores/authStore";
import AppHeaderLogo from "@/components/svgs/AppHeaderLogo";
import { useProfileStore } from "@/stores/profileStore";
export default function App() {
  const [appIsReady, setAppIsReady] = useState(false);
  const router = useRouter();
  const { verifyToken, isAuthenticated } = useAuthStore();
  const [tokenChecked, setTokenChecked] = useState(false);
  const {profile, fetchProfile} = useProfileStore();
  useEffect(() => {
    async function prepare() {
      try {
        const token = await AsyncStorage.getItem("token");
        if (token) {
          await verifyToken({ token: JSON.parse(token) });
          await fetchProfile(JSON.parse(token));
          setTokenChecked(true);
        } else {
          setTokenChecked(true);
        }
        // await new Promise(resolve => setTimeout(resolve, 2000));
      } catch (e) {
        console.warn(e);
      } finally {
        setAppIsReady(true);
      }
    }
    prepare();
  }, []);
  useEffect(() => {
```

```
    if (tokenChecked && appIsReady) {
      router.replace(isAuthenticated ? "/(tabs)/home/screens/Upcoming" : "/auth/login");
    }
  }, [tokenChecked, isAuthenticated, appIsReady]);
  return (
    <View
      style={{ flex: 1, alignItems: 'center', justifyContent: 'center', backgroundColor: "black" }}>
      <AppHeaderLogo />
    </View>
  );
}
```

(tabs)/_layout.tsx:

```
import { Tabs } from "expo-router";
import { MaterialCommunityIcons } from "@expo/vector-icons";
import { ColorValue, Pressable, StyleSheet } from "react-native";
import AppHeader from "@/components/AppHeader";
import ProfileHeader from "@/components/ProfileHeader";
import { useSheetStore } from "@/stores/sheetStore";

export default function TabLayout() {
 const { openCreateSheet } = useSheetStore();
 const handleCreatePress = () => {
  openCreateSheet();
 };
 return (
  <>
   <Tabs
    screenOptions={{
     tabBarStyle: {
      backgroundColor: "#2f2f2f",
      borderTopWidth: 0,
      paddingTop: 8,
      paddingBottom: 7,
     },
     tabBarActiveTintColor: "white",
     tabBarInactiveTintColor: "#999",
     headerShown: true,
     tabBarShowLabel: false,
```

```
      header: () => <AppHeader />,
    }}
  >
    <Tabs.Screen
      name="home"
      options={{
        tabBarIcon: ({ color }: { color: ColorValue }) => (
          <MaterialCommunityIcons name="home-outline" size={26} color={color} />
        ),
      }}
    />
    <Tabs.Screen
      name="explore"
      options={{
        tabBarIcon: ({ color }: { color: ColorValue }) => (
          <MaterialCommunityIcons name="compass-outline" size={26} color={color} />
        ),
      }}
    />
    <Tabs.Screen
      name="create"
      options={{
        tabBarButton: () => (
          <Pressable style={styles.createButton} onPress={handleCreatePress}>
            <MaterialCommunityIcons name="plus-circle-outline" size={26} color={"#999999"} />
          </Pressable>
        ),
      }}
    />
    <Tabs.Screen
      name="notifications"
      options={{
        tabBarIcon: ({ color }: { color: ColorValue }) => (
          <MaterialCommunityIcons name="heart-outline" size={26} color={color} />
        ),
      }}
    />
    <Tabs.Screen
```

```
        name="profile"
        options={{
          tabBarIcon: ({ color }: { color: ColorValue }) => (
            <MaterialCommunityIcons name="account-outline" size={26} color={color} />
          ),
          header: ()=> <ProfileHeader/>
        }}
      />
      <Tabs.Screen
        name="space/[id]"
        options={{
          href: null,
        }}
      />
      <Tabs.Screen
        name="settings"
        options={{
          href: null,
        }}
      />
    </Tabs>
  </>
 );
}

const styles = StyleSheet.create({
 createButton: {
  // backgroundColor: "red",
  justifyContent: "center",
  alignItems: "center",
  marginTop: 6,
  // marginBottom: 25,
 },
});
```

EventSheet.tsx

```
import { useSheetStore } from '@/stores/sheetStore';
import BottomSheet, { BottomSheetScrollView, BottomSheetBackdrop } from '@gorhom/bottom-sheet';
import { useCallback, useEffect, useRef } from 'react';
import { StyleSheet, View } from 'react-native';
import EventPage from './EventPage';
const EventSheet = () => {
  const { isOpen, closeBottomSheet, selectedEvent } = useSheetStore();
  const bottomSheetRef = useRef<BottomSheet>(null);
  useEffect(() => {
    if (isOpen) {bottomSheetRef.current?.snapToIndex(0);} else {
      bottomSheetRef.current?.close();
    }
  }, [isOpen, selectedEvent]);
  const renderBackDrop = useCallback(
    (props: any) => (
      <BottomSheetBackdrop
        {...props}
        pressBehavior="close"
        appearsOnIndex={0}
        disappearsOnIndex={-1}
      />
    ),
    [closeBottomSheet]
  );
  return (
    <BottomSheet
      ref={bottomSheetRef}
      index={-1}
      snapPoints={['88%']}
      enablePanDownToClose={true}
      animateOnMount={true}
      bottomInset={0}
      enableDynamicSizing={false}
      onChange={(index) => {
        if (index === -1) {
          closeBottomSheet();
        }
```

```
        }}
        handleStyle={styles.handle}
        backgroundStyle={styles.background}
        handleIndicatorStyle={styles.indicator}
        backdropComponent={renderBackDrop}
      >
        <BottomSheetScrollView
          style={styles.contentContainer}
          showsHorizontalScrollIndicator={false}
          showsVerticalScrollIndicator={false}
        >
          <View>
            <EventPage event={selectedEvent} />
          </View>
        </BottomSheetScrollView>
      </BottomSheet>
    );
};
export default EventSheet;
const styles = StyleSheet.create({
  handle: {
    borderTopLeftRadius: 30,
    borderTopRightRadius: 30,
  },
  background: {
    backgroundColor: '#1e1e1e',
    borderTopLeftRadius: 30,
    borderTopRightRadius: 30,
  },
});
```

8. RESULT/OUTPUT SCREENS

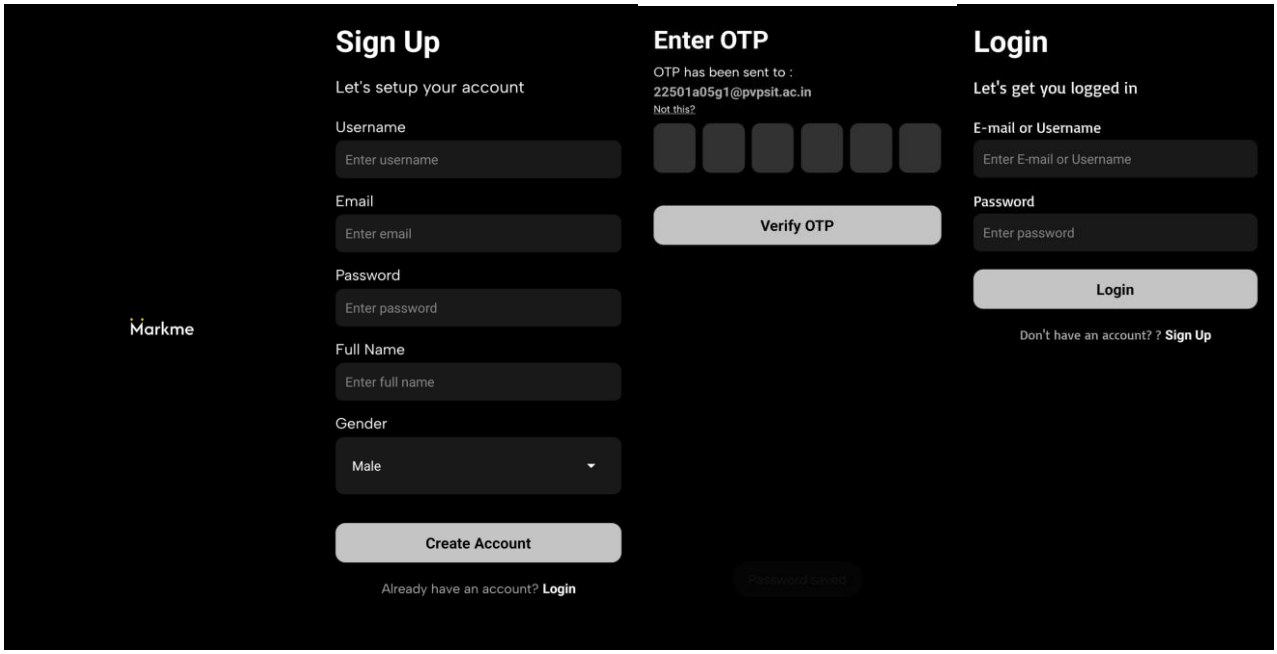


Figure 1: Sign Up and Login - These are signup and login pages. The signup page allows users to create an account and use the app, while the login page with password authentication.

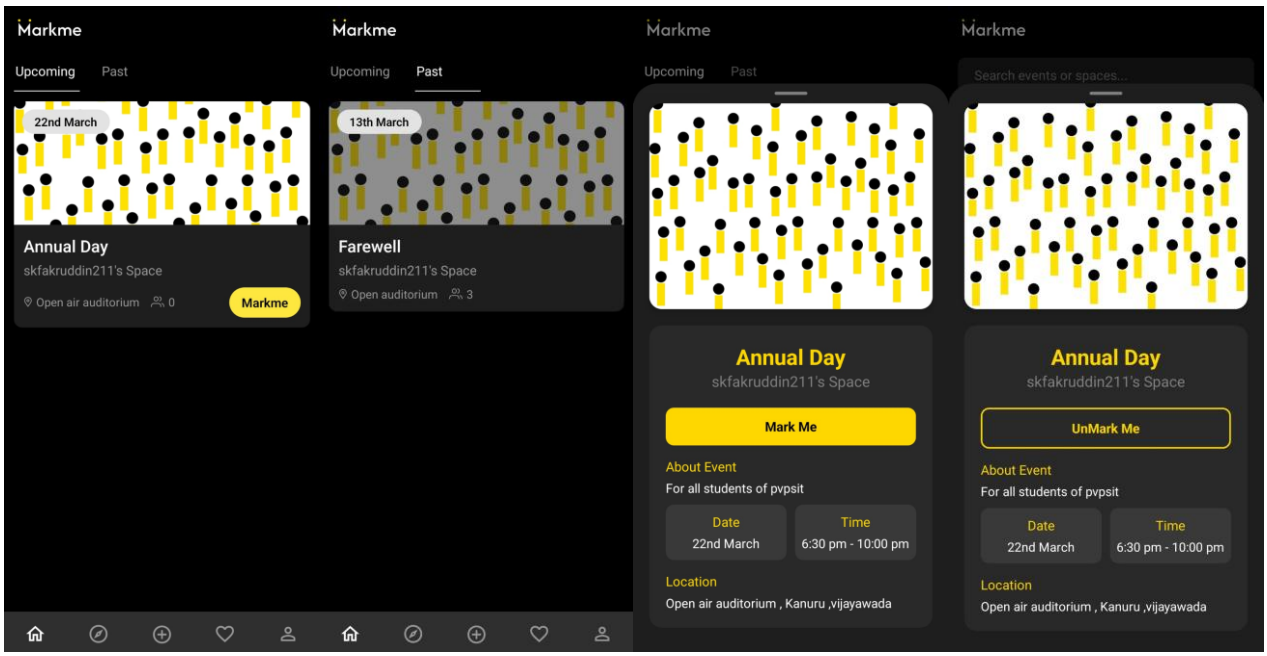


Figure 2: Home Page: This is the home page where upcoming and past events are displayed. Event card contains details e.g. the space it belongs to, the number of attendees, the location, and the date and time of the event. Users can join an event by clicking the 'MarkMe' button. If they change their mind, they can use the 'UnmarkMe' button to withdraw from the event.

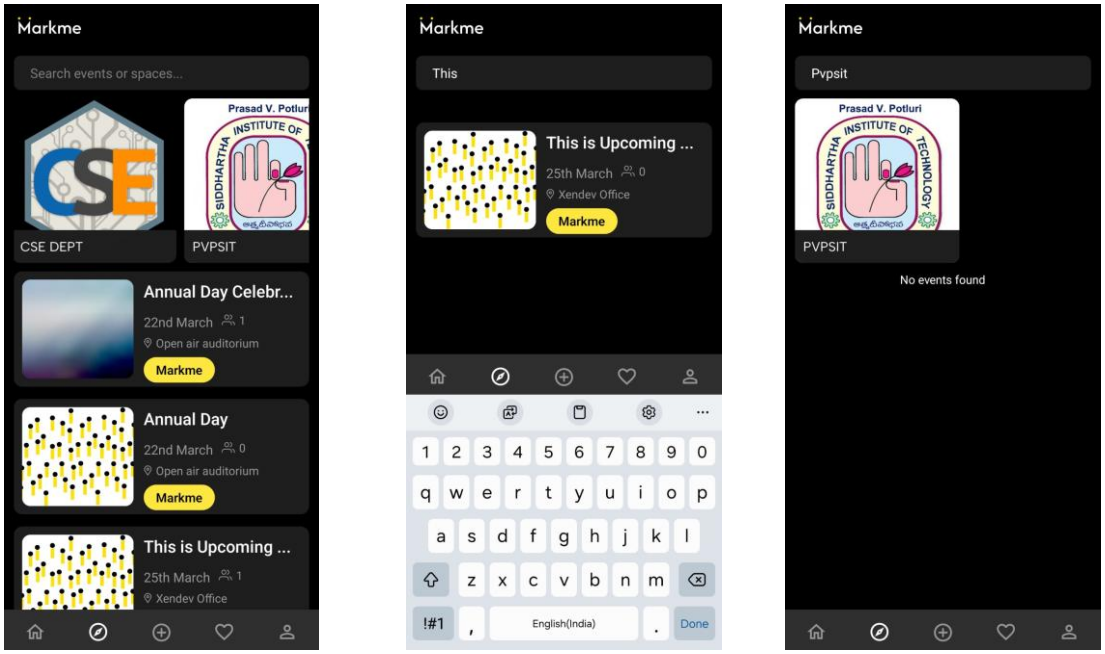


Figure 3: Explore Page - The Explore Page of MarkMe allows users to browse and search for events and spaces. It features a search bar for filtering, event cards displaying key details like name, date, location, and attendees.

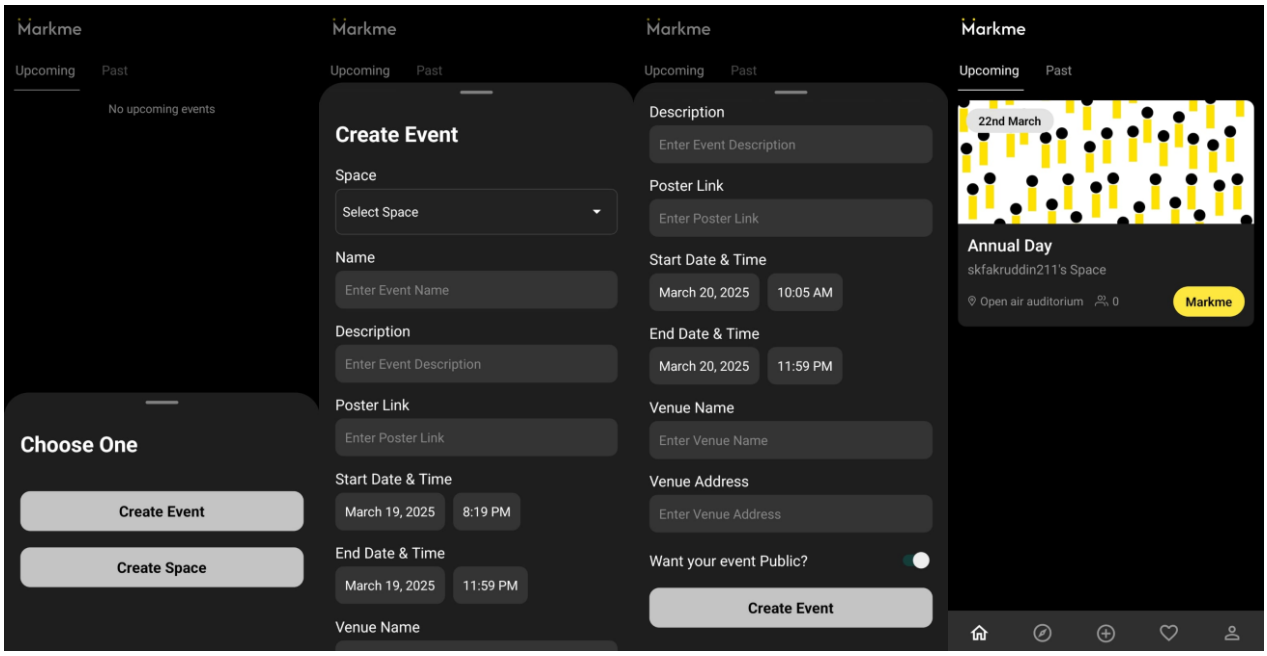


Figure 4: Creating Events - This page allows users to create events.First, the app prompts the user to choose whether they want to create a space or an event. If the 'Create Event' option is chosen, the app displays a card where the user must fill in the required text boxes and click the 'Create Event' button to successfully create the event.

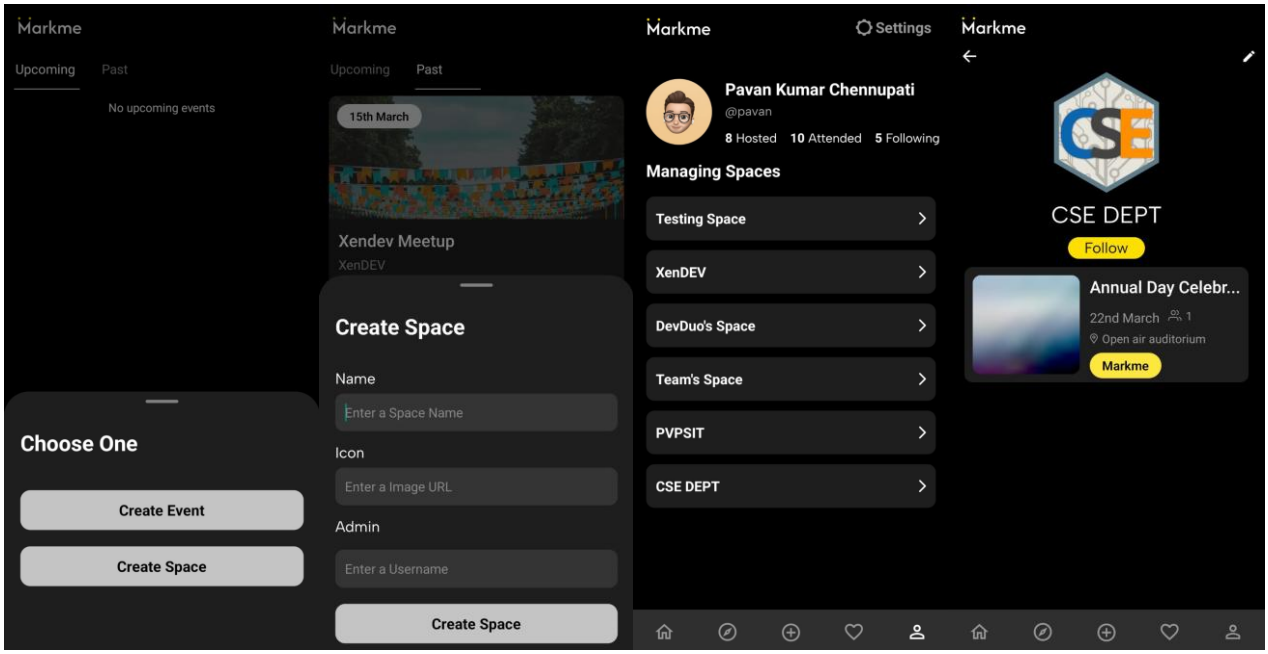


Figure 5: Create Space - The Create Space feature in MarkMe allows users to establish dedicated spaces for event organization and community interactions. Users can provide a name, icon, and admin details to personalize their space.

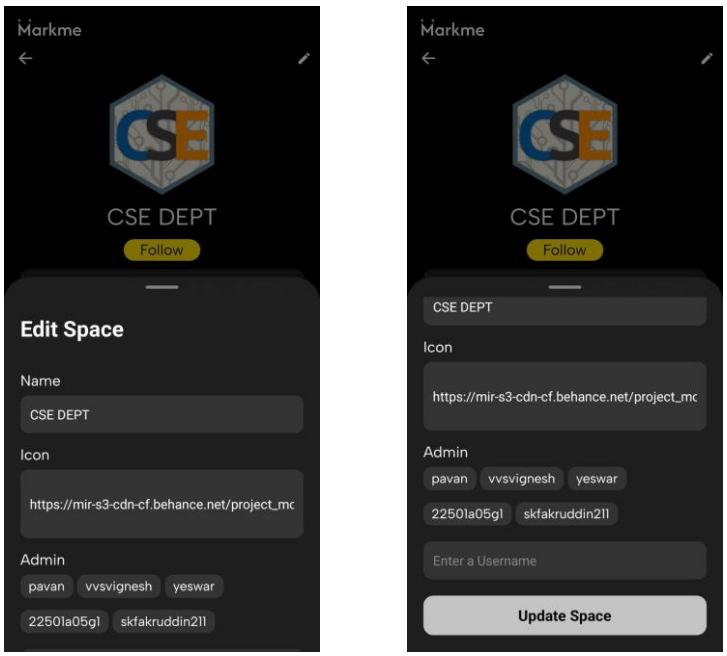


Figure 6: Edit Space Details - Space admins can edit space details such as the name, icon, and manage space admins by adding or removing them.

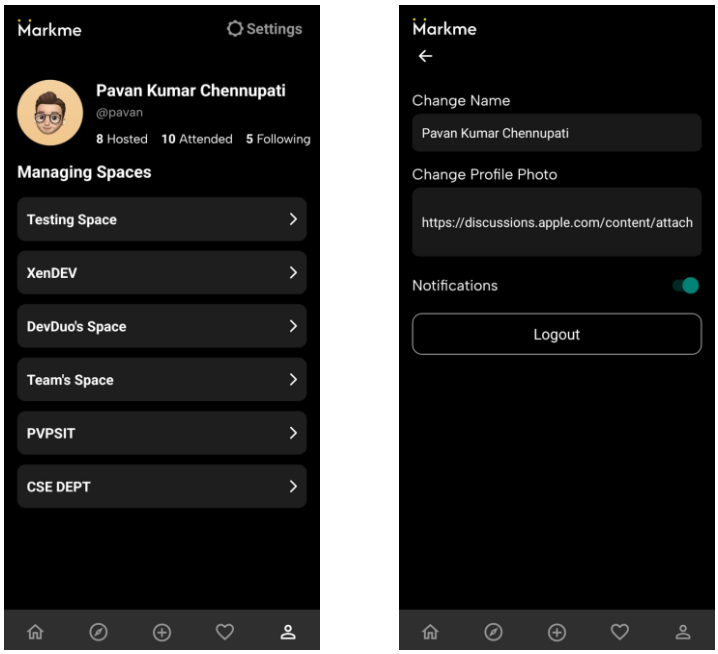


Figure 7: Settings - When the user clicks on their profile and navigates to the settings, they can update their name, change their profile photo, and modify their notification preferences.

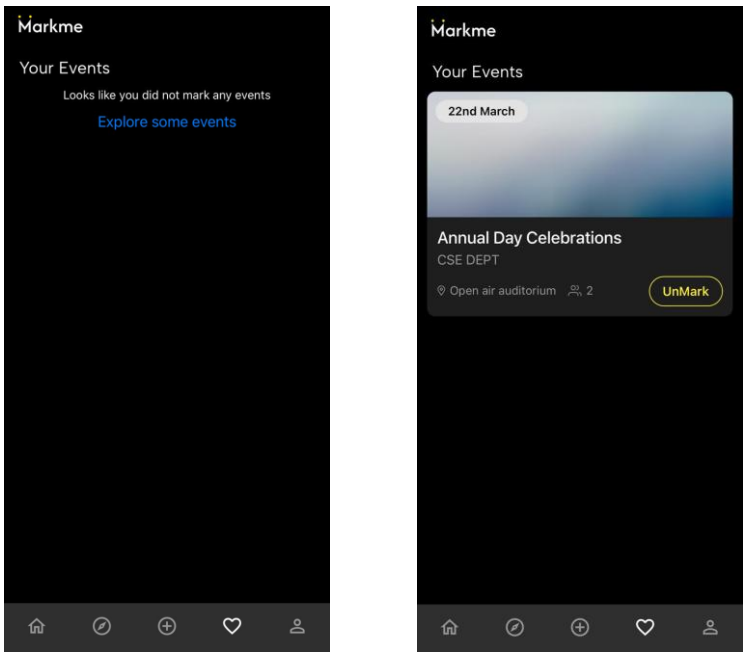


Figure 8: Your Events: On this page, the events the user has marked are displayed. They can also unmark events from this page.

9. CONCLUSION & FUTURE ENHANCEMENTS

Mark Me is a seamless and efficient event management application designed to simplify event and attendee management. By offering a user-friendly interface and robust functionality, it enables organizers to monitor attendance, verify participation, and streamline event workflows effortlessly. The system ensures a secure and accessible experience for both hosts and participants.

Built with React Native (TypeScript) and Expo, along with Node.js, Express, and MongoDB, Mark Me delivers a highly responsive and scalable experience. The streamlined event tracking enhances efficiency, making it an ideal solution for educational institutions, corporate events, and community gatherings.

Future enhancements will focus on offline mode support, advanced analytics for attendance trends, role-based access controls, and integration with third-party calendar systems. Features like push notifications for reminders, AI-driven attendance insights, and enhanced accessibility options will further improve usability. These improvements will position Mark Me as a powerful tool for event management application.

10. REFERENCES (WEB SITE URLS)

GitHub repository links:

https://github.com/pavancos/markme
https://github.com/vigneshvaranasi/markmeEngine

1. Expo Docs - https://expo.dev

2. Gorhom bottom sheet Docs - https://gorhom.dev/react-native-bottom-sheet/

3. Expo Tabs - https://docs.expo.dev/router/advanced/tabs

4. Mongoose Docs - https://mongoosejs.com

5. Eas Docs - https://docs.expo.dev/build/introduction

6. Android Studio Docs - https://developer.android.com/develop