“HyderateMate App – A Mobile Application Development Project”

An Project Document submitted in partial fulfilment

of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATIONTECHNOLOGY

Submitted by

GAYATHRI VADDI(A22126511181)

DEPARTMENT OF INFORMATION TECHNOLOGY

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)

(Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC A+)

Sangivalasa- 531162, Bheemunipatnam Mandal, Visakhapatnam Dist. (A.P).

(2025–2026)

ANILNEERUKONDAINSTITUTEOFTECHNOLOGYAND SCIENCES (UGC AUTONOMOUS)

(Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC A+)

Sangivalasa-531162,Bheemunipatnam Mandal,Visakhapatnam Dist.(A.P).

(2025–2026)

CERTIFICATE

Certified that the project documentation **"HydrateMate App – A Mobile Application Development Project"** is a bonafide work carried out by **Gayathri Vaddi** bearing Register No:**A22126511181**, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology** at **Anil Neerukonda Institute of Technology and Sciences (Affiliated to Andhra University), Visakhapatnam**, during the academic year **2025–2026**.

It is hereby certified that all corrections indicated for internal assessment have been duly incorporated in the document.

HEAD OF THE DEPARTMENT

Lecture In–Charge

Prof. M.Rekha Sundari
(Information Technology)

INDEX

AIM:

The aim of this project is to design and develop a mobile application using React Native that helps users track their daily water intake**,** receive hydration reminders**,** and monitor progress through charts and achievements. The app uses local storage**,** notifications**,** and modern UI components to enhance user engagement and health awareness.

DESCRIPTION:

HydrateMate is a React Native mobile application designed to encourage users to stay hydrated throughout the day. It allows users to log their daily water intake, set personal goals based on age and weight, and receive periodic reminders through local notifications.

The project is divided into multiple modules:

1. **User Setup Module** – Users enter their basic details such as name, age, weight, and daily schedule (wake-up and bedtime). The app then calculates a recommended daily hydration goal.
2. **Home Module** – Displays the user's daily progress, goal, and remaining water intake. Users can add water intake amounts manually or quickly log predefined quantities. A visual progress circle and motivational messages enhance engagement.
3. **History Module**– Shows a 7-day graphical representation of water intake using a **Line Chart**, enabling users to monitor trends and consistency.
4. **Achievements Module**– Awards badges for streaks and achievements such as "7-Day Streak" or "Overachiever." This element motivates continuous app usage.
5. **Settings Module** – Allows users to edit their profile, toggle **Dark Mode**, test notifications, reset data, and manage reminders.
6. **Notifications Module** – Uses **Expo Notifications** to send hydration reminders based on user-defined intervals and sleep/wake schedule.

Features Of HyderateMate App:

- Personalized hydration goal based on age and weight
- Real-time progress visualization
- Daily streaks and achievements
- Light & Dark mode
- Push notifications for reminders
- Persistent storage for offline use
- Modern, responsive UI design

Scope &Future Enhancement:

HydrateMate can be further enhanced by adding:

- Cloud synchronization using Firebase
- Voice and watch integration (smart watch reminders)
- Custom drink types (tea, coffee, juice tracking)
- Advanced analytics for weekly and monthly reports
- Social leader board for motivation

COMPONENTSUSED:

The development of the **HyderateMate App** required the integration of various software frameworks, hardware resources, and modular components. Each component played a significant role in ensuring smooth functionality, responsive design, and real-time gameplay.

A. Software Components

1. **React Native Framework**
    - AJavaScript-basedframeworkthatenablescross-platformmobileapplicationdevelopment.
    - Allows building native-like apps for **Android** and **iOS** using a single codebase.
    - Provides UI components, styling flexibility, and high performance suitable for mobile games.
2. **Expo CLI (Client + SDK)**
    - Simplifies React Native development by providing a managed workflow.
    - Helps run the app quickly on mobile devices using the **ExpoGo** app without requiring Android Studio or Xcode initially.
    - Provides APIs for device features like storage, sensors, and net working.
3. **React Navigation Library**
    - Enables smooth navigation between app screens such as Home, History, Settings, and Achievements.
4. **AsyncStorage**
    - Provides local persistent storage for user data, intake logs, and settings.
5. **Day.js Library:**
    - Light weight date manipulation library used for handling daily logs and streak calculations.
6. **react-native-chart-kit**
    - Renders the Interactive 7-day Hydration history Chart.
7. **Visual Studio Code(IDE)**
    - Development environment used for coding, debugging, and running the app.
    - Provides extensions for React Native development, Git integration, and syntax highlighting.
8. **Testing & Debugging Tools**
    - **ExpoGo Mobile App:** For running and testing the app instantly on Android/ iOS.
    - **Android Emulator / iOS Simulator:** For debugging in a controlled environment.
    - **Chrome Developer Tools:** For inspecting console logs, network requests, and debugging JavaScript.

B. Hardware Components

1. **Development Machine (Laptop/PC)**
    - **Processor:** Intel Core i3/i5 or equivalent(minimum)
    - **RAM:**4GB (8 GB recommended for smooth development)
    - **Storage:** At least 10GB free for SDKs, dependencies, and project files
    - **Operating System:** Windows10/11, Linux, or mac OS

2. **Mobile Device(for Testing &Deployment)**
    - **Android Device:**
        - Minimum2GBRAM,Android 8.0+
        - ExpoGo installed to test the app directly
    - **iOS Device (Optional):**
        - iOS12+with Expo Go for live testing

C. Functional Components(App Modules)

1. **App Provider & Context Management:**
    - Manages global state, including user profile, water intake logs, and settings. Data persists using AsyncStorage.
2. **Welcome Screen (User Setup):**
    - Collects User details and calculates recommended hydration goals dynamically.
3. **Home Screen:**
    - Displays daily progress with an animated water-fill visualization and progress percentage.
    - Users can log water intake and receive motivational prompts.
4. **History Screen:**
    - Displays a 7-day trend chart showing water intake data, allowing users to analyze their hydration consistency.
5. **Achievement Screen:**
    - Displays badges based on user streaks and milestones, adding gamification.
6. **Settings Screen:**
    - Provides user customization options like dark mode toggle, data reset, and notification testing.
7. **Notification Screen:**
    - Schedules reminders at regular intervals between the user's wake and sleep times using Expo Notifications.

D. Supporting Components

The Supporting Components in the **HyderateMate App** are essential elements that ensure smooth functionality, responsiveness, and a good user experience. These components complement the main modules by providing the UI, state management, and game logic utilities.

1. **UI Components**
    - **Linear Gradient:** For aesthetic screen backgrounds.
    - **Animated Views:** For the progress wave effect.
    - **Ionicons:** Tab icons for visual clarity.
    - **Touchable Opacity & Switch:** For user interaction and toggling preferences.
2. **Hooks Used**
    - **useState** for local state.
    - **useEffect** for side effects like notifications and data loading.
    - **useRef** for animated values.

3. **Data Storage & retrieval**
    - use **AsyncStorage** to persist user info, intake logs, and theme settings.
4. **Chart Rendering**
    - **LineChart** is used to render intake data trends for the last 7 days.
5. **Dynamic Theming**
    - Light and Dark themes generated dynamically with gradient backgrounds and text color changes.
6. **Notification Scheduling**
    - Automatically schedules daily hydration reminders with customizable frequency.

PSEUDOCODE:

Module1: User Setup

```
START
IF user opens app for the first time THEN
    DISPLAY input fields:
      - Name
      - Age
      - Weight
      - Wake-up Time
      - Bedtime
    IF all fields are filled THEN
      CALCULATE dailyGoal = weight * 35 ml
      ADJUST goal based on age:
        IF age < 30 THEN goal = goal * 1.05
        IF age > 55 THEN goal = goal * 0.9
      SAVE user details (name, weight, age, wakeTime, bedTime, goal) to local storage
      NAVIGATE to Home Screen
    ELSE
      SHOW message "Please fill all required fields"
    ENDIF
ENDIF
END
```

Module 2: Home Screen

```
START
RETRIEVE user data and today's intake log from local storage
DISPLAY:
    - Daily Goal (goal in ml)
    - Current Intake (in ml)
    - Percentage progress (intake / goal * 100)
    - Animated water level visualization
USER inputs intake amount (in ml)
IF input is valid THEN
    ADD amount to today's intake
    SAVE updated intake log
    UPDATE progress display
    SHOW motivational message ("Great job!" or "Keep sipping!")
    IF total intake >= goal THEN
      SHOW alert " Goal Achieved!"
      TRIGGER celebration animation
    ENDIF
ELSE
```

```
    SHOW error message "Enter a valid number"
  ENDIF
  PROVIDE quick add button (+250 ml)
  PROVIDE reset today's data option
  END
```

Module 3: History Module

```
  START
  RETRIEVE last 7 days' intake logs from local storage
  FOR each day in the past 7 days DO
    FETCH intake amount
    IF no record exists THEN
      SET intake = 0
    ENDIF
    STORE values for chart plotting
  ENDFOR
  DISPLAY line chart showing intake trend for last 7 days
  DISPLAY daily values below chart
  END
```

Module 4: Achievements Module

```
  START
  COMPUTE current streak:
    SET streak = 0
    FOR each previous day in order
      IF intake >= goal THEN
        INCREMENT streak
      ELSE
        BREAK loop
      ENDIF
    ENDFOR
  INITIALIZE achievements = []
  IF streak >= 7 THEN ADD "7-Day Streak" to achievements
  IF streak >= 30 THEN ADD "30-Day Streak" to achievements
  IF any day intake >= 1.2 * goal THEN ADD "Overachiever" to achievements
  DISPLAY all achievements
  IF achievements list is empty THEN
    SHOW message "No achievements yet — keep going!"
  ENDIF
  END
```

Module 5: Settings Module

```
  START
  DISPLAY editable user profile:
    - Name
    - Age
    - Weight
    - Daily Goal
  ALLOW user to modify details
  IF "Save Changes" clicked THEN
    VALIDATE all inputs
    UPDATE stored user data
    SHOW message "Profile Updated Successfully"
  ENDIF
```

```
TOGGLE Dark Mode
   IF ON → Switch to dark theme
   IF OFF → Switch to light theme
TOGGLE Notifications
   IF ON → Enable periodic reminders
   IF OFF → Cancel all scheduled reminders
PROVIDE:
   - Test Reminder button (sends sample notification)
   - Reset Today's Data option
   - Clear All Data option (deletes all saved data)
END
```

Module 6: Notifications & Remainder Scheduling

```
START
IF notifications permission not granted THEN
   REQUEST permission from user
ENDIF
IF remindersEnabled = TRUE THEN
   CANCEL all previously scheduled notifications
   SET startTime = user.wakeTime
   SET endTime = user.bedTime
   SET interval = reminderInterval (default 120 mins)
   FOR every interval between startTime and endTime DO
      SCHEDULE notification:
         TITLE: "HydrateMate"
         BODY: "Time to drink water — tap to log quickly!"
         TRIGGER: repeating daily
      STORE notification ID
   ENDFOR
ELSE
   CANCEL all scheduled notifications
ENDIF
SAVE notification IDs to local storage
END
```

Module 7: Data Persistance & Theming

```
START
USE AsyncStorage for saving and retrieving:
   - User Profile
   - Intake Log
   - Settings (Dark Mode, Notifications)
   - Notification IDs
APPLY Theme:
   IF darkMode = TRUE THEN
      APPLY dark colors and gradients
   ELSE
      APPLY light colors and gradients
   ENDIF
END
```

CODE:

APP.JS

```
//----------------------IMPORTING---------------------//
import React, { useEffect, useState, createContext, useContext, useRef } from 'react';
import {
  View, Text, TextInput, TouchableOpacity, StyleSheet, Alert, ScrollView, Switch,
  Dimensions, Animated, Platform, SafeAreaView, StatusBar
} from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
import * as Notifications from 'expo-notifications';
import Constants from 'expo-constants';
import { NavigationContainer, DefaultTheme, DarkTheme } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Ionicons } from '@expo/vector-icons';
import dayjs from 'dayjs';
import { LineChart } from 'react-native-chart-kit';
import { LinearGradient } from 'expo-linear-gradient';

// -------------------- Context --------------------//
const AppContext = createContext();
function useApp() { return useContext(AppContext); }

// -------------------- Helper Functions --------------------//
const STORAGE_KEYS = {
  USER: 'HM_userData_v2',
  INTAKE_LOG: 'HM_intakeLog_v2',
  SETTINGS: 'HM_settings_v2',
  NOTIF_IDS: 'HM_notif_ids_v2'
};
function formatDate(date = new Date()) { return dayjs(date).format('YYYY-MM-DD'); }
async function loadJson(key, fallback) {
  try { const s = await AsyncStorage.getItem(key); return s ? JSON.parse(s) : fallback; }
  catch (e) { return fallback; }
}
async function saveJson(key, val) {
  try { await AsyncStorage.setItem(key, JSON.stringify(val)); }
  catch (e) { console.warn('save error', e); }
}

// -------------------- Theme --------------------//
```

```
function makeTheme(isDark) {
  if (!isDark) {
    return {
      background: '#f6fbff',
      surface: '#ffffff',
      card: 'rgba(255,255,255,0.95)',
      primary: '#0077b6',
      accent: '#00b4d8',
      text: '#083344',
      subtext: '#666',
      border: '#e6f0f4',
      inputBg: '#fff',
      wave: '#00bfff',
      topbar: '#05668d',
      statusBarStyle: 'dark-content',
      gradientLight: ['#e0f7ff', '#cfeefd'],
      gradientPrimary: ['#00b4d8', '#0077b6']
    };
  } else {
    return {
      background: '#071023',
      surface: '#0b1220',
      card: '#0f1724',
      primary: '#5dd6ff',
      accent: '#4fb6d9',
      text: '#e6f7ff',
      subtext: '#a9c7d7',
      border: '#122033',
      inputBg: '#071323',
      wave: '#2bb7ff',
      topbar: '#7ad0ff',
      statusBarStyle: 'light-content',
      gradientLight: ['#042033', '#063142'],
      gradientPrimary: ['#036b8f', '#024b6a']
    };
  }
}
// -------------------- Notifications Setup --------------------//
async function registerForPushNotificationsAsync() {
  if (!Constants.isDevice) {
    console.warn('Must use physical device for notifications');
```

```
    return false;
  }
  const { status: existingStatus } = await Notifications.getPermissionsAsync();
  let finalStatus = existingStatus;
  if (existingStatus !== 'granted') {
    const { status } = await Notifications.requestPermissionsAsync();
    finalStatus = status;
  }
  return finalStatus === 'granted';
}
async function cancelScheduledNotificationsAndClearStorage() {
  try {
    const ids = await loadJson(STORAGE_KEYS.NOTIF_IDS, []);
    if (Array.isArray(ids)) {
      await Promise.all(ids.map(id => Notifications.cancelScheduledNotificationAsync(id)));
    }
    await saveJson(STORAGE_KEYS.NOTIF_IDS, []);
  } catch (e) { console.warn('cancel notifs err', e); }
}
/**
 * scheduleReminders(user, settings)
 */
async function scheduleReminders(user, settings) {
  try {
    await cancelScheduledNotificationsAndClearStorage();
    if (!user || !settings || !settings.remindersEnabled) return [ ];
    const [wakeH, wakeM] = (user?.wakeTime || settings?.wakeTime || '07:00').split(':').map(Number);
    const [bedH, bedM] = (user?.bedTime || settings?.bedTime || '23:00').split(':').map(Number);
    const interval = Math.max(15, (settings.reminderIntervalMins || 120)); // min 15 min
    const wakeMinutes = (wakeH || 7) * 60 + (wakeM || 0);
    let bedMinutes = (bedH || 23) * 60 + (bedM || 0);
    if (bedMinutes <= wakeMinutes) bedMinutes += 24 * 60;
    const times = [];
    for (let t = wakeMinutes; t <= bedMinutes; t += interval) {
      const minutesOfDay = t % (24 * 60);
      const hour = Math.floor(minutesOfDay / 60);
      const minute = Math.floor(minutesOfDay % 60);
      times.push({ hour, minute });
    }
    if (times.length === 0) times.push({ hour: wakeH, minute: wakeM });
```

```
    const ok = await registerForPushNotificationsAsync();
    if (!ok) {
      console.warn('No push permission; skipping scheduling');
      return [];
    }
    const createdIds = [ ];
    for (const tm of times) {
      const id = await Notifications.scheduleNotificationAsync({
        content: {
          title: "HydrateMate",
          body: "Time to drink water — tap to log quickly!",
          data: { screen: 'Home' }
        },
        trigger: {
          hour: tm.hour,
          minute: tm.minute,
          repeats: true
        }
      });
      createdIds.push(id);
    }
    await saveJson(STORAGE_KEYS.NOTIF_IDS, createdIds);
    console.log('Scheduled reminders', createdIds.length);
    return createdIds;
  } catch (e) { console.warn('scheduleReminders err', e); return []; }
}
// ---------- App Provider ----------//
function AppProvider({ children }) {
  const [user, setUser] = useState(null);
  const [intakeLog, setIntakeLog] = useState([]);
  const [settings, setSettings] = useState({ darkMode: false, remindersEnabled: true, reminderIntervalMins:
    120, wakeTime: '07:00', bedTime: '23:00' });
  const [notifReady, setNotifReady] = useState(false);
  // load saved data
  useEffect(() => {
    (async () => {
      const savedUser = await loadJson(STORAGE_KEYS.USER, null);
      const savedLog = await loadJson(STORAGE_KEYS.INTAKE_LOG, []);
      const savedSettings = await loadJson(STORAGE_KEYS.SETTINGS, settings);
      setUser(savedUser);
      setIntakeLog(savedLog);
```

```
      setSettings(prev => ({ ...prev, ...(savedSettings || {}) }));
    })();
  }, []);
  useEffect(() => { saveJson(STORAGE_KEYS.USER, user); }, [user]);
  useEffect(() => { saveJson(STORAGE_KEYS.INTAKE_LOG, intakeLog); }, [intakeLog]);
  useEffect(() => { saveJson(STORAGE_KEYS.SETTINGS, settings); }, [settings]);
  useEffect(() => {
    (async () => {
      const ok = await registerForPushNotificationsAsync();
      setNotifReady(ok);
      if (settings.remindersEnabled) {
        await scheduleReminders(user, settings);
      } else {
        await cancelScheduledNotificationsAndClearStorage();
      }
    })();
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [user, settings.remindersEnabled, settings.reminderIntervalMins, settings.wakeTime, settings.bedTime]);
  const addIntake = (amount, type = 'water') => {
    if (!user) return;
    const today = formatDate();
    const cloned = [...intakeLog];
    const idx = cloned.findIndex(r => r.date === today);
    if (idx >= 0) cloned[idx].intake += amount;
    else cloned.push({ date: today, intake: amount });
    setIntakeLog(cloned);
  };
  const resetToday = () => {
    const today = formatDate();
    const cloned = intakeLog.filter(r => r.date !== today);
    setIntakeLog(cloned);
  };
  const theme = makeTheme(settings.darkMode);
  const value = { user, setUser, intakeLog, setIntakeLog, addIntake, resetToday, settings, setSettings,
    notifReady, theme };
  return <AppContext.Provider value={value}>{children}</AppContext.Provider>;
}
// -------------------- Small UI Helpers --------------------//
function TopBar({ title }) {
  const { theme } = useApp();
  return (
```

```
<View style={[styles.topbar, { backgroundColor: 'transparent' }]}>
<Text style={[styles.topbarTitle, { color: theme.topbar }]}>{title}</Text>
</View>
  );
}
// -------------------- Welcome Screen --------------------//
function WelcomeScreen({ navigation }) {
  const { setUser, setSettings, settings, theme } = useApp();
  const [name, setName] = useState('');
  const [weight, setWeight] = useState('');
  const [age, setAge] = useState('');
  const [wakeTime, setWakeTime] = useState('07:00');
  const [bedTime, setBedTime] = useState('23:00');
  const calculateWaterGoal = () => {
    const w = parseFloat(weight); const a = parseInt(age);
    if (!w || !a) return 2000;
    let ml = w * 35;
    if (a < 30) ml *= 1.05; else if (a > 55) ml *= 0.9;
    return Math.round(ml);
  };
  const onStart = async () => {
    if (!name || !weight || !age) { Alert.alert('Please fill required fields'); return; }
    const goal = calculateWaterGoal();
    const userObj = { name, weight: parseFloat(weight), age: parseInt(age), wakeTime, bedTime, goal,
     createdAt: new Date().toISOString() };
    setUser(userObj);
    setSettings(s => ({ ...s, wakeTime, bedTime }));
    await saveJson(STORAGE_KEYS.USER, userObj);
    if (navigation && navigation.replace) navigation.replace('MainTabs');
  };
  return (
<SafeAreaView style={{ flex: 1, backgroundColor: theme.background }}>
<StatusBar barStyle={theme.statusBarStyle} />
<LinearGradient colors={theme.gradientLight} style={styles.centered}>
<Text style={[styles.welcomeTitle, { color: theme.text }]}>Welcome to HydrateMate</Text>
<Text style={[styles.welcomeSubtitle, { color: theme.subtext }]}>Personalize your hydration plan</Text>
<View style={[styles.welcomeCard, { backgroundColor: theme.card }]}>
<TextInput
        placeholder="Name"
        placeholderTextColor={theme.subtext}
        style={[styles.input, { backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border
```

```
    }]}
        value={name} onChangeText={setName} />
<View style={{ flexDirection: 'row', justifyContent: 'space-between', width: '100%' }}>
<TextInput placeholder="Weight (kg)" placeholderTextColor={theme.subtext} style={[styles.input, { width: '48%', backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border }]} value={weight} onChangeText={setWeight} keyboardType="numeric" />
<TextInput placeholder="Age" placeholderTextColor={theme.subtext} style={[styles.input, { width: '48%', backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border }]} value={age} onChangeText={setAge} keyboardType="numeric" />
</View>
<View style={{ flexDirection: 'row', justifyContent: 'space-between', width: '100%' }}>
<TextInput placeholder="Wake-up (HH:MM)" placeholderTextColor={theme.subtext} style={[styles.input, { width: '48%', backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border }]} value={wakeTime} onChangeText={setWakeTime} />
<TextInput placeholder="Bedtime (HH:MM)" placeholderTextColor={theme.subtext} style={[styles.input, { width: '48%', backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border }]} value={bedTime} onChangeText={setBedTime} />
</View>
<Text style={[styles.help, { color: theme.subtext }]}>Recommended goal: <Text style={{ fontWeight: '700', color: theme.text }}>{calculateWaterGoal()} ml</Text></Text>
<TouchableOpacity style={styles.startBtn} onPress={onStart}>
<LinearGradient colors={theme.gradientPrimary} style={styles.btnGrad}>
<Text style={styles.btnText}>Get Started</Text>
</LinearGradient>
</TouchableOpacity>
</View>
</LinearGradient>
</SafeAreaView>
  );
}
// -------------------- Home Screen --------------------//
function HomeScreen() {
  const { user, intakeLog, addIntake, settings, theme } = useApp();
  const [input, setInput] = useState('');
  const [message, setMessage] = useState('');
  const [showConfetti, setShowConfetti] = useState(false);
  const today = formatDate();
  const todayEntry = intakeLog.find(r => r.date === today) || { date: today, intake: 0 };
  const intake = todayEntry.intake;
  const goal = user?.goal || 2000;
  const progress = Math.min(intake / goal, 1);

  const waveAnim = useRef(new Animated.Value(progress)).current;
```

```
  useEffect(() => {
    Animated.timing(waveAnim, { toValue: progress, duration: 900, useNativeDriver: false }).start();
  }, [progress]);
  const handleAdd = () => {
    const n = parseInt(input);
    if (!n || n <= 0) { Alert.alert('Enter positive ml'); return; }
    addIntake(n);
    setInput('');
    const tips = ['Keep sipping!', 'Great job!', 'Hydration boost!', 'Nice!'];
    setMessage(tips[Math.floor(Math.random() * tips.length)]);
    if (intake + n >= goal) {
      setShowConfetti(true);
      setTimeout(() => setShowConfetti(false), 4000);
      Alert.alert('□ Goal reached', 'You reached your daily water goal!');
    }
  };
  const waveHeight = waveAnim.interpolate({ inputRange: [0,1], outputRange: ['0%', '100%'] });
  return (
<ScrollView contentContainerStyle={[styles.screen, { backgroundColor: theme.background }]}>
<StatusBar barStyle={theme.statusBarStyle} />
<TopBar title={Hello, ${user?.name || 'Friend'}} />
<LinearGradient colors={[theme.surface, theme.background]} style={[styles.headerCard, { backgroundColor:
    theme.surface }]}>
<Text style={[styles.title, { color: theme.text }]}>Daily goal: <Text style={{ color: theme.primary }}>{goal}
    ml</Text></Text>
<Text style={[styles.sub, { color: theme.subtext }]}>Today: <Text style={{ fontWeight: '700', color:
    theme.text }}>{intake} ml</Text></Text>
</LinearGradient>
<View style={styles.progressCard}>
<View style={styles.circleContainer}>
<Text style={[styles.bigText, { color: theme.primary }]}>{Math.round(progress * 100)}%</Text>
<Text style={{ color: theme.subtext }}>{intake} / {goal} ml</Text>
</View>
<View style={[styles.waveWrapper, { borderColor: theme.border, backgroundColor: theme.card }]}>
<Animated.View style={[styles.waveFill, { height: waveHeight, backgroundColor: theme.wave }]} />
</View>
</View>
<View style={{ width: '100%', alignItems: 'center' }}>
<TextInput
      style={[styles.input, { backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border
    }]}
      placeholder="Add ml (e.g. 250)"
```

```
        placeholderTextColor={theme.subtext}
        value={input} onChangeText={setInput} keyboardType="numeric" />
<TouchableOpacity style={styles.primaryBtn} onPress={handleAdd}>
<LinearGradient colors={theme.gradientPrimary} style={styles.btnGrad}>
<Text style={styles.btnText}>Add</Text>
</LinearGradient>
</TouchableOpacity>
<TouchableOpacity style={[styles.secondaryBtn, { marginTop: 8, backgroundColor: theme.card, borderColor:
    theme.border }]} onPress={() => {
        Alert.alert('Quick add', 'Add a quick 250 ml?', [{ text: 'Cancel' }, { text: 'Add', onPress: () =>
    addIntake(250) }]);
      }}>
<Text style={{ fontWeight: '600', color: theme.primary }}>Quick +250 ml</Text>
</TouchableOpacity>
      {message ? <Text style={[styles.help, { color: theme.subtext }]}>{message}</Text> : null}
      {showConfetti ? <Text style={{ marginTop: 8, fontSize: 28 }}>□</Text> : null}
<View style={[styles.summaryCard, { backgroundColor: theme.card, borderColor: theme.border }]}>
<Text style={{ fontWeight: '600', color: theme.text }}>Remaining: {Math.max(goal - intake, 0)} ml</Text>
<Text style={{ color: theme.subtext }}>Time left: {computeHoursLeft(user)}</Text>
</View>
</View>
</ScrollView>
  );
}
function computeHoursLeft(user) {
  try {
    if (!user) return '—';
    const now = dayjs();
    const [bh, bm] = (user.bedTime || '23:00').split(':').map(Number);
    let bed = dayjs().hour(bh).minute(bm);
    if (bed.isBefore(now)) bed = bed.add(1, 'day');
    const diff = bed.diff(now, 'hour');
    return diff > 0 ? ${diff} hrs : 'few hours';
  } catch (e) { return '—'; }
}
// ---------- History Screen ----------//
function HistoryScreen() {
  const { intakeLog, theme } = useApp();
  const last7Days = lastNDays(7);
  const last7 = last7Days.map(d => {
    const r = intakeLog.find(i => i.date === d) || { date: d, intake: 0 };
```

```
    return r.intake;
  });
  const labels = last7Days.map(d => dayjs(d).format('DD'));
  const chartConfig = {
    backgroundGradientFrom: theme.surface,
    backgroundGradientTo: theme.surface,
    decimalPlaces: 0,
    color: (opacity = 1) => ${hexToRgba(theme.primary, opacity)},
    labelColor: (opacity = 1) => ${hexToRgba(theme.subtext, opacity)},
    propsForDots: { r: '4', strokeWidth: '2' },
  };
  return (
<ScrollView contentContainerStyle={[styles.screen, { backgroundColor: theme.background }]}>
<StatusBar barStyle={theme.statusBarStyle} />
<TopBar title="History" />
<Text style={[styles.title, { fontSize: 20, color: theme.text }]}>Last 7 days</Text>
<LineChart
        data={{ labels, datasets: [{ data: last7 }] }}
        width={Dimensions.get('window').width - 32}
        height={220}
        chartConfig={chartConfig}
        bezier
        style={{ borderRadius: 12, marginTop: 8, backgroundColor: theme.surface }}
      />
<View style={{ marginTop: 16, width: '100%', alignItems: 'center' }}>
        {last7Days.slice().reverse().map(d => {
          const intake = intakeLog.find(i => i.date === d)?.intake || 0;
          return (
<View key={d} style={[styles.historyRow, { borderColor: theme.border, backgroundColor: theme.surface
    }]}>
<Text style={{ color: theme.text }}>{dayjs(d).format('ddd DD MMM')}</Text>
<Text style={{ fontWeight: '700', color: theme.primary }}>{intake} ml</Text>
</View>
          );
        })}
</View>
</ScrollView>
  );
}
function lastNDays(n) {
  const arr = [];
```

```
  for (let i = n - 1; i >= 0; i--) arr.push(dayjs().subtract(i, 'day').format('YYYY-MM-DD'));
  return arr;
}
function hexToRgba(hex, opacity) {
  // simple helper: converts #rrggbb to rgba(...)
  const h = hex.replace('#', '');
  const r = parseInt(h.substring(0,2),16), g = parseInt(h.substring(2,4),16), b = parseInt(h.substring(4,6),16);
  return rgba(${r}, ${g}, ${b}, ${opacity});
}
// ---------- Settings Screen ----------//
function SettingsScreen() {
  const { user, setUser, settings, setSettings, resetToday, setIntakeLog, theme } = useApp();
  const [editableUser, setEditableUser] = useState(user || {});
  const [customGoal, setCustomGoal] = useState('');
  useEffect(() => {
    setEditableUser(user || {});
    setCustomGoal(user?.goal?.toString() || '');
  }, [user]);
  const saveUserInfo = async () => {
    if (!editableUser.name || !editableUser.weight || !editableUser.age) {
      Alert.alert('⚠ Please fill all fields');
      return;
    }
    const updated = {
      ...editableUser,
      weight: parseFloat(editableUser.weight),
      age: parseInt(editableUser.age),
      goal: parseInt(customGoal) || editableUser.goal || 2000,
    };
    setUser(updated);
    await saveJson(STORAGE_KEYS.USER, updated);
    Alert.alert('✅ Profile updated successfully!');
  };
  const toggleDark = () => setSettings({ ...settings, darkMode: !settings.darkMode });
  const clearAllData = () => {
    Alert.alert('□ Clear All Data', 'Are you sure you want to delete all saved data?', [
      { text: 'Cancel', style: 'cancel' },
      {
        text: 'Clear',
        style: 'destructive',
        onPress: async () => {
```

```
          await AsyncStorage.clear();
          setIntakeLog([]);
          setUser(null);
          Alert.alert('✅ All data cleared');
        },
      },
    ]);
  };
  const sendTestReminder = async () => {
    await Notifications.scheduleNotificationAsync({
      content: {
        title: '💧 Hydration Reminder',
        body: 'Time to take a sip! Stay hydrated 💦',
      },
      trigger: null,
    });
    Alert.alert('✅ Test reminder sent!');
  };
  const initials = editableUser.name
    ? editableUser.name.split(' ').map(w => w[0]).join('').toUpperCase()
    : '?';
  return (
<LinearGradient colors={theme.gradientLight} style={{ flex: 1 }}>
<ScrollView contentContainerStyle={[styles.screen, { padding: 20, backgroundColor: theme.background }]}>
<StatusBar barStyle={theme.statusBarStyle} />
<View style={{ alignItems: 'center', marginBottom: 20 }}>
<View style={{
          width: 90, height: 90, borderRadius: 45, backgroundColor: theme.primary, justifyContent: 'center',
    alignItems: 'center',
          shadowColor: '#000', shadowOpacity: 0.25, shadowRadius: 6, elevation: 6,
        }}>
<Text style={{ fontSize: 36, color: '#fff', fontWeight: 'bold' }}>{initials}</Text>
</View>
<Text style={{ fontSize: 22, marginTop: 10, color: theme.text, fontWeight: '600' }}>
          {editableUser.name || 'Your Name'}
</Text>
</View>

<View style={[styles.card, { backgroundColor: theme.card, borderColor: theme.border }]}>
<Text style={[styles.sectionTitle, { color: theme.primary }]}>👤 Profile Information</Text>
<TextInput style={[styles.input, { backgroundColor: theme.inputBg, color: theme.text, borderColor:
```

```
theme.border }]} placeholder="Name" placeholderTextColor={theme.subtext} value={editableUser.name || ''} onChangeText={(v) => setEditableUser({ ...editableUser, name: v })} />
<TextInput style={[styles.input, { backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border }]} placeholder="Age" placeholderTextColor={theme.subtext} keyboardType="numeric" value={editableUser.age?.toString() || ''} onChangeText={(v) => setEditableUser({ ...editableUser, age: v })} />
<TextInput style={[styles.input, { backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border }]} placeholder="Weight (kg)" placeholderTextColor={theme.subtext} keyboardType="numeric" value={editableUser.weight?.toString() || ''} onChangeText={(v) => setEditableUser({ ...editableUser, weight: v })} />
<TextInput style={[styles.input, { backgroundColor: theme.inputBg, color: theme.text, borderColor: theme.border }]} placeholder="Daily Goal (ml)" placeholderTextColor={theme.subtext} keyboardType="numeric" value={customGoal} onChangeText={setCustomGoal} />
<TouchableOpacity style={styles.primaryBtn} onPress={saveUserInfo}>
<LinearGradient colors={theme.gradientPrimary} style={styles.btnGrad}>
<Text style={styles.btnText}>□ Save Changes</Text>
</LinearGradient>
</TouchableOpacity>
</View>
<View style={[styles.card, { backgroundColor: theme.card, borderColor: theme.border }]}>
<Text style={[styles.sectionTitle, { color: theme.primary }]}>□ Preferences</Text>
<View style={styles.settingRow}>
<Text style={[styles.settingLabel, { color: theme.text }]}>Dark Mode</Text>
<Switch value={settings.darkMode} onValueChange={toggleDark} />
</View>
<View style={styles.settingRow}>
<Text style={[styles.settingLabel, { color: theme.text }]}>Reminders</Text>
<Switch
          value={settings.remindersEnabled}
          onValueChange={() =>
            setSettings({ ...settings, remindersEnabled: !settings.remindersEnabled })
          }
        />
</View>
<TouchableOpacity style={[styles.secondaryBtn, { marginTop: 10, backgroundColor: theme.card, borderColor: theme.border }]} onPress={sendTestReminder}>
<Text style={{ color: theme.primary, fontWeight: '600' }}>□ Test Reminder</Text>
</TouchableOpacity>
</View>

<View style={[styles.card, { backgroundColor: theme.card, borderColor: theme.border }]}>
<Text style={[styles.sectionTitle, { color: theme.primary }]}>□ Actions</Text>
<TouchableOpacity style={[styles.secondaryBtn, { backgroundColor: '#e0f7fa' }]} onPress={resetToday}>
```

```
<Text style={{ color: '#00796b', fontWeight: '600' }}> Reset Today</Text>
</TouchableOpacity>
<TouchableOpacity style={[styles.secondaryBtn, { backgroundColor: '#ffe5e5' }]} onPress={clearAllData}>
<Text style={{ color: '#c62828', fontWeight: '600' }}> Clear All Data</Text>
</TouchableOpacity>
</View>
<Text style={{ textAlign: 'center', color: theme.subtext, marginTop: 20 }}>
        HydrateMate | Stay Fresh, Stay Hydrated
</Text>
</ScrollView>
</LinearGradient>
  );
}
// -------------------- Achievements Screen --------------------//
function AchievementsScreen() {
  const { intakeLog, theme } = useApp();
  const streak = computeStreak(intakeLog);
  const achievements = [];
  if (streak >= 7) achievements.push('7-day Streak');
  if (streak >= 30) achievements.push('30-day Streak');
  const overachiever = intakeLog.some(d => d.intake >= 1.2 * (loadTodayGoalFromStorageSync() || 2000));
  if (overachiever) achievements.push('Overachiever');
  return (
<ScrollView contentContainerStyle={[styles.screen, { backgroundColor: theme.background }]}>
<StatusBar barStyle={theme.statusBarStyle} />
<TopBar title="Achievements" />
<Text style={[styles.title, { color: theme.text }]}>Achievements</Text>
     {achievements.length === 0 ? <Text style={[styles.help, { color: theme.subtext }]}>No achievements yet
    — keep going!</Text> : (
       achievements.map((a,i) => (
<View key={i} style={[styles.achRow, { backgroundColor: theme.card, borderColor: theme.border
    }]}><Text style={{ fontWeight: '600', color: theme.text }}> {a}</Text></View>
       ))
     )}
</ScrollView>
  );
}
function computeStreak(log) {
  const goal = loadTodayGoalFromStorageSync() || 2000;
  let streak = 0;
  for (let i = 0; i < 365; i++) {
```

```
    const d = dayjs().subtract(i, 'day').format('YYYY-MM-DD');
    const entry = log.find(r => r.date === d);
    if (entry && entry.intake >= goal) streak++; else break;
  }
  return streak;
}
function loadTodayGoalFromStorageSync() { return 2000; }
// -------------------- Navigation --------------------//
const Tab = createBottomTabNavigator();
export default function App() {
  return (
<AppProvider>
<MainApp />
</AppProvider>
  );
}
function MainApp() {
  const { user, settings, theme } = useApp();
  const [ready, setReady] = useState(false);
  useEffect(() => {
    (async () => {
      const ok = await registerForPushNotificationsAsync();
      if (ok) console.log('Notifications ready');
      setReady(true);
    })();
  }, []);
  if (!ready) return null;
  // Use react-navigation built-in themes for nav chrome, keep consistent with our theme choice
  const navTheme = settings.darkMode ? DarkTheme : DefaultTheme;
  return (
<NavigationContainer theme={navTheme}>
      {user ? (
<Tab.Navigator screenOptions={({ route }) => ({
          headerShown: false,
          tabBarActiveTintColor: theme.primary,
          tabBarStyle: { paddingVertical: Platform.OS === 'ios' ? 8 : 4, height: 60, backgroundColor: theme.card
    },
          tabBarIcon: ({ color, size }) => {
            let name = 'water-outline';
            if (route.name === 'Home') name = 'water-outline';
            if (route.name === 'History') name = 'stats-chart-outline';
```

```
          if (route.name === 'Settings') name = 'settings-outline';
          if (route.name === 'Achievements') name = 'trophy-outline';
          return <Ionicons name={name} size={size} color={color} />;
        }
      })}>
<Tab.Screen name="Home" component={HomeScreen} />
<Tab.Screen name="History" component={HistoryScreen} />
<Tab.Screen name="Achievements" component={AchievementsScreen} />
<Tab.Screen name="Settings" component={SettingsScreen} />
</Tab.Navigator>
    ) : (
<WelcomeScreen navigation={{ replace: () => setReady(true) }} />
    )}
</NavigationContainer>
  );
}
// -------------------- Styles --------------------//
const styles = StyleSheet.create({
  centered: {
    flexGrow: 1,
    alignItems: 'center',
    justifyContent: 'center',
    padding: 20
  },
  screen: {
    flexGrow: 1,
    padding: 16,
    alignItems: 'center',
  },
  topbar: {
    width: '100%',
    paddingVertical: 12,
    paddingHorizontal: 8,
    alignItems: 'center',
    marginBottom: 6
  },
  topbarTitle: {
    fontSize: 22,
    fontWeight: '700',
  },
  welcomeTitle: {
```

```
  fontSize: 32,
  fontWeight: '800',
  marginBottom: 6,
  textAlign: 'center'
},
welcomeSubtitle: {
  marginBottom: 12
},
welcomeCard: {
  marginTop: 8,
  width: '100%',
  borderRadius: 16,
  padding: 16,
  alignItems: 'center',
  shadowColor: '#000',
  shadowOpacity: 0.08,
  shadowRadius: 8,
  elevation: 6
},
headerCard: {
  width: '100%',
  padding: 14,
  borderRadius: 12,
  alignItems: 'center',
  marginBottom: 12,
  shadowColor: '#000',
  shadowOpacity: 0.06,
  shadowRadius: 6,
  elevation: 3,
},
// ---- Text Styles ----//
title: {
  fontSize: 20,
  fontWeight: '700',
  marginVertical: 6,
  textAlign: 'center'
},
sub: {
  fontSize: 14,
  marginBottom: 8
},
```

```
help: {
  marginTop: 8
},
// ---- Input Fields ----//
input: {
  borderWidth: 1,
  borderRadius: 12,
  padding: 10,
  width: 300,
  marginTop: 10,
  textAlign: 'center',
  shadowColor: '#000',
  shadowOpacity: 0.03,
  shadowRadius: 2,
  elevation: 2
},
// ---- Buttons ----//
primaryBtn: {
  marginTop: 12,
  width: 220,
  borderRadius: 12,
  overflow: 'hidden',
  elevation: 3
},
btnGrad: {
  paddingVertical: 12,
  alignItems: 'center',
  justifyContent: 'center'
},
btnText: {
  color: '#fff',
  fontWeight: '700',
  fontSize: 16
},
startBtn: {
  marginTop: 12,
  width: '100%',
  borderRadius: 12,
  overflow: 'hidden'
},
secondaryBtn: {
```

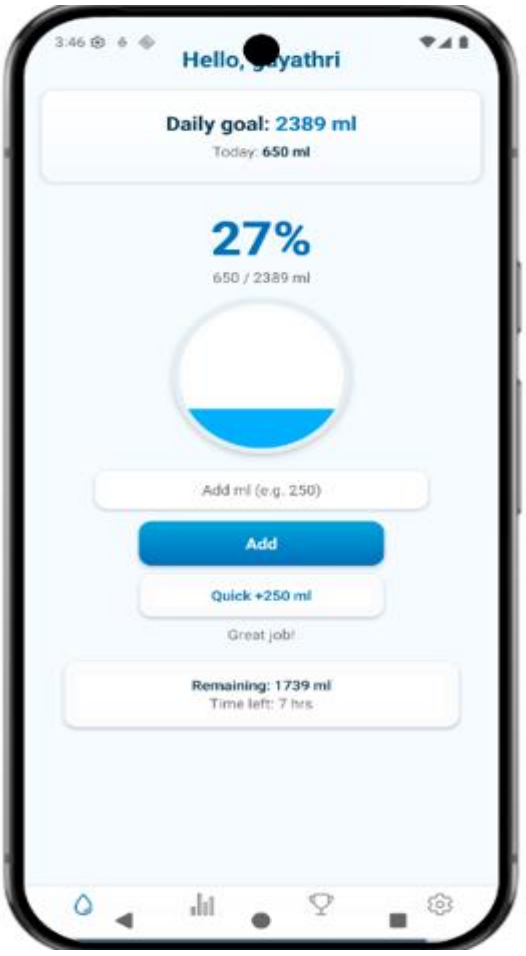
```
  padding: 12,
  borderRadius: 12,
  borderWidth: 1,
  marginTop: 10,
  alignItems: 'center',
  width: 220,
  shadowColor: '#000',
  shadowOpacity: 0.02,
  shadowRadius: 2,
  elevation: 2
},
// ---- Progress & Summary ----//
progressCard: {
  width: '100%',
  alignItems: 'center',
  marginTop: 12,
  marginBottom: 6
},
circleContainer: {
  alignItems: 'center',
  marginBottom: 12
},
bigText: {
  fontSize: 48,
  fontWeight: '800',
},
waveWrapper: {
  width: 160,
  height: 160,
  borderRadius: 80,
  overflow: 'hidden',
  borderWidth: 6,
  shadowColor: '#000',
  shadowOpacity: 0.06,
  shadowRadius: 8,
  elevation: 4
},
waveFill: {
  width: '100%',
  position: 'absolute',
  bottom: 0
```

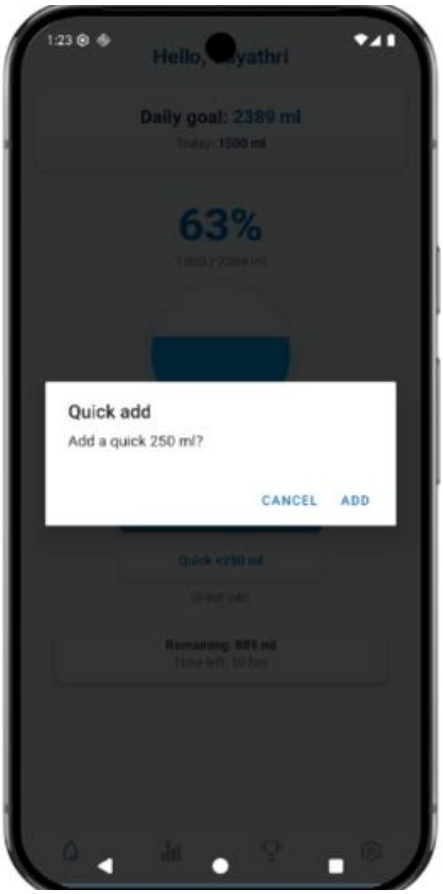
```
},
summaryCard: {
  marginTop: 18,
  padding: 14,
  borderRadius: 12,
  borderWidth: 1,
  width: '90%',
  alignItems: 'center',
  shadowColor: '#000',
  shadowOpacity: 0.05,
  shadowRadius: 3,
  elevation: 2
},
// ---- History & Achievement ----//
historyRow: {
  flexDirection: 'row',
  justifyContent: 'space-between',
  paddingVertical: 10,
  borderBottomWidth: 0.3,
  width: Dimensions.get('window').width - 32
},
achRow: {
  padding: 12,
  borderRadius: 12,
  borderWidth: 1,
  marginTop: 8,
  width: Dimensions.get('window').width - 32,
  shadowColor: '#000',
  shadowOpacity: 0.03,
  shadowRadius: 3,
  elevation: 2
},
// ---- Settings Cards ----//
card: {
  borderRadius: 18,
  padding: 16,
  marginBottom: 20,
  width: '95%',
  shadowColor: '#000',
  shadowOpacity: 0.12,
  shadowRadius: 6,
```

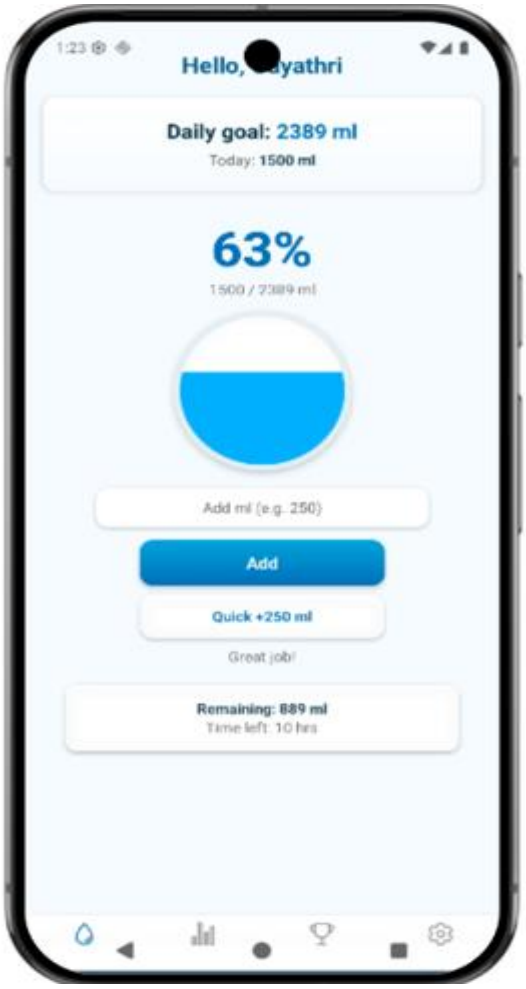
```
    elevation: 4,
    borderWidth: 1
  },
  sectionTitle: {
    fontSize: 18,
    fontWeight: '700',
    marginBottom: 12,
  },
  settingRow: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    paddingVertical: 10,
    borderBottomWidth: 0.5,
  },
  settingLabel: {
    fontSize: 16,
  },
});
```

OUTPUT:

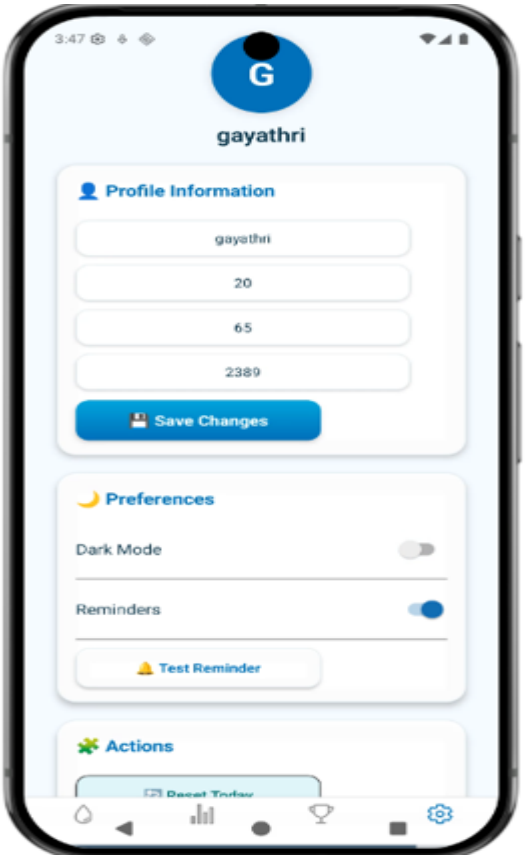

Welcome to HydrateMate
Personalize your hydration plan
gayathri
65
20
07:00
23:30
Recommended goal: 2389 ml
Get Started


Daily goal: 2389 ml
Today: 650 ml
27%
650 / 2389 ml
Add ml (e.g. 250)
Add
Quick +250 ml
Great job!
Remaining: 1739 ml
Time left: 7 hrs


Daily goal: 2389 ml
63%
Quick add
Add a quick 250 ml?
CANCEL
ADD


Hello, ayathri
Daily goal: 2389 ml
Today: 1500 ml
63%
1500 / 2389 ml
Add ml (e.g. 250)
Add
Quick +250 ml
Great job!
Remaining: 889 ml
Time left: 10 hrs


Last 7 days
650
488
325
163
0
06
07
08
09
10
11
12
Sun 12 Oct 650 ml
Sat 11 Oct 0 ml
Fri 10 Oct 0 ml
Thu 09 Oct 0 ml
Wed 08 Oct 0 ml
Tue 07 Oct 0 ml
Mon 06 Oct 0 ml


G
gayathri
Profile Information
gayathri
20
65
2389
Save Changes
Preferences
Dark Mode
Reminders
Test Reminder
Actions