Mobile Applications

Lecture 3 – hands-on assignment

PawCare – complete app navigation

Stack, Tabs, Drawer, Top Tabs, modals, deep links and route guards – React Navigation 7

Mateusz Pawełkiewicz

Version: October 2026

Expo SDK 58 · React Native 0.88 · React 19.3 · TypeScript

Code verified with tsc, ESLint and Jest; the app was run in the browser (react-native-web)

Goal

We replace state-based view switching with real navigation. The goal is to show **as many** React Navigation 7 features as possible in one realistic app: four kinds of navigators nested in each other, different screen presentations (card, modal, transparent modal, a sheet with detents), header options, deep links, route guards for signed-out users, a leave-guard for an unfinished booking and navigation from outside components.

Structure: **Root Stack** (sign-in / app + modals) → **Drawer** (side menu) → **Bottom Tabs** (Vets, Schedule, Visits, Profile) → inside the tabs a **Stack** of vets and **Material Top Tabs** with days.

> **Note**
> The finished code of this stage: the "code/l03" folder next to this document.

What we build

- the whole navigation map as TypeScript types (every `navigate` call and every parameter checked at compile time),
- route guards: a signed-out user sees only the welcome and sign-in screens (conditionally rendered screen groups),
- a Stack with different presentations: `card`, `modal` (booking), `transparentModal` ("Meet the vet" video), `formSheet` with `sheetAllowedDetents` (filters),
- headers: a large iOS title, a search bar in the header, a transparent header with blur, header buttons (share, favorite), a title computed from route params,
- bottom tabs with icons, a badge, history-based Back, a guest guard on the Visits tab (`tabPress` + `preventDefault`) and `popToTopOnBlur`,
- a Drawer with custom content (`drawerContent`), a sign-out item and opening from code (`DrawerActions`),
- top tabs with days (swipeable, lazy),
- passing a result back (filters → list) with `popTo` and merged params,
- a leave-guard for booking (`usePreventRemove`) with a confirmation dialog,
- deep links (`pawcare://vet/v1`, `https://pawcare.example.com/vet/v1`) with nested path config and a "Not found" screen,
- `navigation.preload` – the vet profile renders in the background as soon as a card is touched,
- navigation from outside components (`navigationRef`) and screen tracking (`onStateChange`).

What from the documentation we use in practice

Topic from the lecture / docs	Where in the code
NavigationContainer: `linking`, `theme`, `onReady`, `onStateChange`, `fallback`	`RootNavigator.tsx`, `navTheme.ts`, `linking.ts`
Native Stack: `Stack.Group`, `presentation` (modal, transparentModal, formSheet)	`RootNavigator.tsx`, `VetsStack.tsx`
formSheet: `sheetAllowedDetents`, `sheetGrabberVisible`, `sheetCornerRadius`	`VetsStack.tsx`

Topic from the lecture / docs	Where in the code
Header options: `headerLargeTitle`, `headerSearchBarOptions`, `headerTransparent`, `headerBlurEffect`, `headerRight`, `headerBackButtonDisplayMode`	`VetsStack.tsx`, `VetsScreen.tsx`, `VetHeaderActions.tsx`
`options` as a function, `getId`, `animation`, `gestureEnabled`	`RootNavigator.tsx`, `VetsStack.tsx`
Bottom Tabs: `tabBarIcon`, `tabBarBadge`, `backBehavior`, `popToTopOnBlur`, `listeners`/`screenListeners`, `animation: "shift"`	`HomeTabs.tsx`
Drawer: `drawerContent`, `DrawerItemList`, `DrawerItem`, `drawerType`, `swipeEdgeWidth`	`MainDrawer.tsx`
Material Top Tabs: `initialParams`, `lazy`, `swipeEnabled`	`ScheduleTopTabs.tsx`
Actions: nested `navigate`, `replace`, `popTo`, `goBack`, `reset` (`CommonActions`), `setOptions`, `preload`	screens in `src/screens/…`
Hooks: `usePreventRemove`, `useScrollToTop`, `useIsFocused`, `useNavigation`	`BookAppointmentScreen.tsx`, `VetsScreen.tsx`
Types: `NativeStackScreenProps`, `CompositeScreenProps`, `NavigatorScreenParams`, global `RootParamList`	`navigation/types.ts`
Deep linking: `prefixes`, nested `config`, `parse`/`stringify`, `*` (NotFound), `Linking.createURL`	`linking.ts`, `AboutScreen.tsx`
Route guards via conditional rendering of screens	`RootNavigator.tsx`, `AuthContext.tsx`
`createNavigationContainerRef` – navigation outside the component tree	`navigationRef.ts`

Before you start

Stage L02 completed.

Installing packages

React Navigation 7 with four navigator types and the native libraries they need (native screens, gestures, a pager for the top tabs):

Terminal

```
npx expo install @react-navigation/native @react-navigation/native-stack @react-navigation/bottom-tabs @react-navigation/drawer @react-navigation/material-top-tabs @react-navigation/elements react-native-screens react-native-safe-area-context react-native-gesture-handler react-native-reanimated react-native-worklets react-native-pager-view expo-linking
```

Watch out

Add `"scheme": "pawcare"` to `app.json` – without a scheme `pawcare://…` deep links do not work in a build (in Expo Go the addresses look like `exp://IP:8081/--/…`).

Files in this stage

The table lists every file that is created (NEW) or changed (CHANGED) in this stage. Changed files are printed IN FULL – simply replace their content.

File	State	Description
src/auth/AuthContext.tsx	NEW	minimal sign-in state (extended in L04, L06, L10)
src/components/VetHeaderActions.tsx	NEW	buttons in the vet profile header
src/navigation/HomeTabs.tsx	NEW	bottom tabs
src/navigation/linking.ts	NEW	deep links: URL -> screen (with parameters)
src/navigation/MainDrawer.tsx	NEW	side menu with custom content
src/navigation/navigationRef.ts	NEW	navigation from outside components (e.g. notifications in L07)
src/navigation/navTheme.ts	NEW	React Navigation theme built from the app theme
src/navigation/RootNavigator.tsx	NEW	navigation container + root Stack with route guards
src/navigation/ScheduleTopTabs.tsx	NEW	top tabs (swipeable) with days
src/navigation/types.ts	NEW	the whole navigation map of the app, as types
src/navigation/VetsStack.tsx	NEW	Stack nested inside the "Vets" tab
src/screens/appointments/AppointmentDetailsScreen.tsx	NEW	visit details + nested navigation
src/screens/appointments/AppointmentsScreen.tsx	NEW	the list of booked visits
src/screens/auth/SignInScreen.tsx	NEW	simplified sign-in (full form in L05)
src/screens/auth/WelcomeScreen.tsx	NEW	welcome screen (signed-out users only)
src/screens/booking/BookAppointmentScreen.tsx	NEW	booking (modal) that guards against leaving
src/screens/drawer/AboutScreen.tsx	NEW	info + outgoing links (Linking)
src/screens/drawer/ClinicsScreen.tsx	NEW	the list of clinics (a side-menu screen)
src/screens/drawer/SettingsScreen.tsx	NEW	settings (working options come in L04)
src/screens/NotFoundScreen.tsx	NEW	unknown deep link
src/screens/profile/ProfileScreen.tsx	NEW	profile + examples of navigating between navigators
src/screens/schedule/ScheduleDayScreen.tsx	NEW	vets on duty on one day (top tab)
src/screens/vets/FiltersScreen.tsx	NEW	a sheet (formSheet) that passes the result back
src/screens/vets/VetDetailsScreen.tsx	NEW	vet profile + upcoming working days
src/screens/vets/VetsScreen.tsx	NEW	vets: search in the header, filters in a sheet
src/screens/vets/VetVideoScreen.tsx	NEW	transparent modal over the previous screen
src/state/AppointmentsContext.tsx	NEW	visits booked during the session (L04 replaces it)
app.json	CHANGED	

File	State	Description
App.tsx	CHANGED	stage L03: providers + navigation
src/components/VetCard.tsx	CHANGED	vet tile in the grid
src/components/VetDetailsModal.tsx	DELETE	
src/screens/BookingView.tsx	DELETE	
src/screens/ScheduleView.tsx	DELETE	
src/screens/VetsView.tsx	DELETE	

Step by step

1. Configuration: the deep link scheme

app.json

Replace the whole file

```
{
  "expo": {
    "name": "PawCare",
    "slug": "pawcare",
    "version": "1.0.0",
    "orientation": "default",
    "icon": "./assets/icon.png",
    "userInterfaceStyle": "automatic",
    "ios": {
      "supportsTablet": true
    },
    "android": {
      "adaptiveIcon": {
        "backgroundColor": "#0E9F8E",
        "foregroundImage": "./assets/android-icon-foreground.png"
      }
    },
    "plugins": [
      "expo-font"
    ],
    "scheme": "pawcare"
  }
}
```

2. Types of the whole navigation

Start with the types – the "map" of the app. Every navigator has its list of screens and params (`…ParamList`). A nested navigator is a param of its parent (`NavigatorScreenParams`). Screen prop types are composed bottom-up with `CompositeScreenProps`, so a screen inside a tab can navigate to a Root Stack screen. Declaring `ReactNavigation.RootParamList` makes `useNavigation()` aware of all screens.

src/navigation/types.ts

New file – the whole navigation map of the app, as types

```
// src/navigation/types.ts – the whole navigation map of the app, as types
import type { CompositeScreenProps, NavigatorScreenParams } from "@react-navigation/native";
import type { NativeStackScreenProps } from "@react-navigation/native-stack";
import type { BottomTabScreenProps } from "@react-navigation/bottom-tabs";
import type { DrawerScreenProps } from "@react-navigation/drawer";
import type { MaterialTopTabScreenProps } from "@react-navigation/material-top-tabs";
import type { Specialty } from "@/data/types";

// ── Level 1: root Stack ──────────────────────────────────────
export type RootStackParamList = {
  Welcome: undefined;
  SignIn: { email?: string } | undefined;
  Main: NavigatorScreenParams<DrawerParamList> | undefined;
  BookAppointment: { scheduleId: string };
  AppointmentDetails: { appointmentId: string };
  VetVideo: { vetId: string };
  NotFound: undefined;
};
```

```
// ── Level 2: Drawer (side menu) ──────────────────────────────
export type DrawerParamList = {
  HomeTabs: NavigatorScreenParams<TabParamList> | undefined;
  Clinics: undefined;
  Settings: undefined;
  About: undefined;
};

// ── Level 3: bottom tabs ─────────────────────────────────────
export type TabParamList = {
  VetsTab: NavigatorScreenParams<VetsStackParamList> | undefined;
  ScheduleTab: NavigatorScreenParams<ScheduleTopTabParamList> | undefined;
  AppointmentsTab: undefined;
  ProfileTab: undefined;
};

// ── Level 4a: Stack inside the "Vets" tab ────────────────────
export type VetFilters = { specialties?: Specialty[]; exoticOnly?: boolean };

export type VetsStackParamList = {
  Vets: VetFilters | undefined;
  VetDetails: { vetId: string };
  Filters: VetFilters | undefined;
};

// ── Level 4b: top tabs inside "Schedule" ─────────────────────
export type ScheduleTopTabParamList = {
  Today: { dayOffset: number };
  Tomorrow: { dayOffset: number };
  DayAfter: { dayOffset: number };
};

// ── Screen prop types (combine navigators from the bottom up) ────────
export type RootStackScreenProps<T extends keyof RootStackParamList> = NativeStackScreenProps<
  RootStackParamList,
  T
>;

export type DrawerScreenPropsOf<T extends keyof DrawerParamList> = CompositeScreenProps<
  DrawerScreenProps<DrawerParamList, T>,
  RootStackScreenProps<keyof RootStackParamList>
>;

export type TabScreenProps<T extends keyof TabParamList> = CompositeScreenProps<
  BottomTabScreenProps<TabParamList, T>,
  DrawerScreenPropsOf<keyof DrawerParamList>
>;

export type VetsScreenProps<T extends keyof VetsStackParamList> = CompositeScreenProps<
  NativeStackScreenProps<VetsStackParamList, T>,
  TabScreenProps<keyof TabParamList>
>;

export type ScheduleTopTabScreenProps<T extends keyof ScheduleTopTabParamList> =
  CompositeScreenProps<
    MaterialTopTabScreenProps<ScheduleTopTabParamList, T>,
    TabScreenProps<keyof TabParamList>
  >;

// Default type for useNavigation() without a generic
declare global {
  namespace ReactNavigation {
    // eslint-disable-next-line @typescript-eslint/no-empty-object-type
    interface RootParamList extends RootStackParamList {}
```

```
  }
}
```

3. Sign-in state and session visits (Context)

For now we only need "signed in / guest / signed out" – a simple `AuthContext`. Visits booked during the session live in `AppointmentsContext` (replaced by a store in lecture 4 and by the API in lecture 6).

src/auth/AuthContext.tsx

New file – minimal sign-in state (extended in L04, L06, L10)

```tsx
// src/auth/AuthContext.tsx – minimal sign-in state (extended in L04, L06, L10)
import { createContext, useContext, useMemo, useState, type ReactNode } from "react";

export type User = { name: string; email?: string; isGuest: boolean };

type AuthContextValue = {
  user: User | null;
  signIn: (user: User) => void;
  signOut: () => void;
};

const AuthContext = createContext<AuthContextValue | null>(null);

export function AuthProvider({ children }: { children: ReactNode }) {
  const [user, setUser] = useState<User | null>(null);

  const value = useMemo(
    () => ({ user, signIn: (u: User) => setUser(u), signOut: () => setUser(null) }),
    [user],
  );

  return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>;
}

export function useAuth() {
  const ctx = useContext(AuthContext);
  if (!ctx) throw new Error("useAuth must be used inside <AuthProvider>");
  return ctx;
}
```

src/state/AppointmentsContext.tsx

New file – visits booked during the session (L04 replaces it)

```tsx
// src/state/AppointmentsContext.tsx – visits booked during the session (L04 replaces it)
import { createContext, useContext, useMemo, useState, type ReactNode } from "react";

export type Appointment = {
  id: string;
  scheduleId: string;
  slot: string; // ISO start time
  serviceId: string;
  total: number;
  createdAt: string;
};

type AppointmentsContextValue = {
  appointments: Appointment[];
  addAppointment: (data: Omit<Appointment, "id" | "createdAt">) => Appointment;
};
```

```
const AppointmentsContext = createContext<AppointmentsContextValue | null>(null);

export function AppointmentsProvider({ children }: { children: ReactNode }) {
  const [appointments, setAppointments] = useState<Appointment[]>([]);

  const value = useMemo<AppointmentsContextValue>(
    () => ({
      appointments,
      addAppointment: (data) => {
        const appointment: Appointment = {
          ...data,
          id: `A${Date.now().toString(36).toUpperCase()}`,
          createdAt: new Date().toISOString(),
        };
        setAppointments((prev) => [appointment, ...prev]);
        return appointment;
      },
    }),
    [appointments],
  );

  return <AppointmentsContext.Provider value={value}>{children}</AppointmentsContext.Provider>;
}

export function useAppointments() {
  const ctx = useContext(AppointmentsContext);
  if (!ctx) throw new Error("useAppointments must be used inside <AppointmentsProvider>");
  return ctx;
}
```

4. Navigation theme and the navigator ref

src/navigation/navTheme.ts

New file – React Navigation theme built from the app theme

```
// src/navigation/navTheme.ts – React Navigation theme built from the app theme
import { DarkTheme, DefaultTheme, type Theme } from "@react-navigation/native";
import { fonts, type AppTheme } from "@/theme/theme";

export function toNavigationTheme(theme: AppTheme): Theme {
  const base = theme.dark ? DarkTheme : DefaultTheme;
  return {
    ...base,
    colors: {
      ...base.colors,
      primary: theme.colors.primary, // active elements (e.g. the selected tab)
      background: theme.colors.background, // screen background
      card: theme.colors.surface, // headers, tab bar, drawer
      text: theme.colors.text,
      border: theme.colors.border,
      notification: theme.colors.danger, // badge color
    },
    // In v7 the theme also contains the fonts used by headers and tabs
    fonts: {
      regular: { fontFamily: fonts.regular, fontWeight: "400" },
      medium: { fontFamily: fonts.medium, fontWeight: "500" },
      bold: { fontFamily: fonts.bold, fontWeight: "700" },
      heavy: { fontFamily: fonts.black, fontWeight: "900" },
    },
  };
}
```

src/navigation/navigationRef.ts

New file – navigation from outside components (e.g. notifications in L07)

```
// src/navigation/navigationRef.ts – navigation from outside components (e.g. notifications in
L07)
import { createNavigationContainerRef } from "@react-navigation/native";
import type { RootStackParamList } from "./types";

export const navigationRef = createNavigationContainerRef<RootStackParamList>();

// Safe call: the navigator might not be ready yet
export function navigateToAppointment(appointmentId: string) {
  if (navigationRef.isReady()) {
    navigationRef.navigate("AppointmentDetails", { appointmentId });
  }
}
```

5. The vets Stack in the "Vets" tab

The list has a large iOS title and a search bar in the header. The vet profile has a transparent, blurred header with buttons (share, favorite). `getId` makes another vet a new screen instead of replacing the params. Filters open as a **sheet** (`formSheet`) with two detents and a grabber.

src/navigation/VetsStack.tsx

New file – Stack nested inside the "Vets" tab

```
// src/navigation/VetsStack.tsx – Stack nested inside the "Vets" tab
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import { DrawerToggleButton } from "@react-navigation/drawer";
import { useAppTheme } from "@/theme/theme";
import type { VetsStackParamList } from "./types";
import { VetsScreen } from "@/screens/vets/VetsScreen";
import { VetDetailsScreen } from "@/screens/vets/VetDetailsScreen";
import { FiltersScreen } from "@/screens/vets/FiltersScreen";
import { VetHeaderActions } from "@/components/VetHeaderActions";

const Stack = createNativeStackNavigator<VetsStackParamList>();

export function VetsStack() {
  const theme = useAppTheme();
  return (
    <Stack.Navigator
      screenOptions={{
        headerTintColor: theme.colors.primary,
        headerBackButtonDisplayMode: "minimal",
        headerTitleAlign: "center", // Android: centered title (iOS – always)
      }}
    >
      <Stack.Screen
        name="Vets"
        component={VetsScreen}
        options={{
          title: "Our vets",
          headerLargeTitle: true, // iOS: large title that collapses while scrolling
          headerLeft: () => <DrawerToggleButton tintColor={theme.colors.primary} />,
        }}
      />
      <Stack.Screen
        name="VetDetails"
        component={VetDetailsScreen}
        // A unique id: another vet is a new screen, not an update of the parameters
        getId={({ params }) => params.vetId}
        // options as a function – access to the route (screen parameters)
```

```
        options={({ route }) => ({
          title: "",
          headerTransparent: true, // the content goes under the header
          headerBlurEffect: "systemChromeMaterial", // iOS: blurred header background
          headerTintColor: "#FFFFFF",
          headerRight: () => <VetHeaderActions vetId={route.params.vetId} />,
        })}
      />
      <Stack.Screen
        name="Filters"
        component={FiltersScreen}
        options={{
          title: "Filters",
          presentation: "formSheet", // a sheet from the bottom
          sheetAllowedDetents: [0.5, 1], // half or full screen
          sheetGrabberVisible: true, // drag handle
          sheetCornerRadius: 24,
        }}
      />
    </Stack.Navigator>
  );
}
```

src/components/VetHeaderActions.tsx

New file – buttons in the vet profile header

```
// src/components/VetHeaderActions.tsx – buttons in the vet profile header
import { useState } from "react";
import { Share, StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { HeaderButton } from "@react-navigation/elements";
import { findVet } from "@/data/mockData";

// The component is passed in the navigator `options` (not via setOptions from the screen),
// so it also works with navigation.preload() – a preloaded screen cannot call setOptions.
export function VetHeaderActions({ vetId }: { vetId: string }) {
  const vet = findVet(vetId);
  const [favorite, setFavorite] = useState(false); // moved to a global store in L04

  return (
    <View style={styles.row}>
      <HeaderButton
        accessibilityLabel="Share"
        onPress={() =>
          vet && Share.share({ message: `${vet.name} at PawCare: pawcare://vet/${vet.id}` })
        }
      >
        <Ionicons name="share-outline" size={22} color="#FFFFFF" />
      </HeaderButton>
      <HeaderButton
        accessibilityLabel={favorite ? "Remove from favorites" : "Add to favorites"}
        onPress={() => setFavorite((f) => !f)}
      >
        <Ionicons name={favorite ? "heart" : "heart-outline"} size={22} color="#FFFFFF" />
      </HeaderButton>
    </View>
  );
}

const styles = StyleSheet.create({ row: { flexDirection: "row" } });
```

src/components/VetCard.tsx

Replace the whole file – vet tile in the grid

```tsx
// src/components/VetCard.tsx – vet tile in the grid
import { memo } from "react";
import { Platform, Pressable, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import type { Vet } from "@/data/types";
import { radius, spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";
import { RatingStars } from "./RatingStars";
import { SpeciesIcons } from "./SpeciesIcons";

// A tiny "blurhash" – blurry placeholder visible before the image downloads
const BLURHASH = "L5H2EC=PM+yV0g-mq.wG9c010J}I";

type Props = {
  vet: Vet;
  onPress: (vet: Vet) => void;
  onPressIn?: (vet: Vet) => void;
  onLongPress?: (vet: Vet) => void;
};

function VetCardComponent({ vet, onPress, onPressIn, onLongPress }: Props) {
  const theme = useAppTheme();
  return (
    <Pressable
      onPress={() => onPress(vet)}
      onPressIn={() => onPressIn?.(vet)}
      onLongPress={() => onLongPress?.(vet)}
      delayLongPress={350}
      android_ripple={{ color: "rgba(0,0,0,0.12)", foreground: true }}
      // A function in `style` receives { pressed } – press feedback on iOS
      style={({ pressed }) => [
        styles.card,
        { backgroundColor: theme.colors.surface, borderColor: theme.colors.border },
        pressed && Platform.OS === "ios" && styles.pressed,
      ]}
      accessibilityRole="button"
      accessibilityLabel={`${vet.name}, ${vet.specialty}`}
      accessibilityHint="Opens the vet profile"
    >
      <Image
        source={vet.photoUrl}
        placeholder={{ blurhash: BLURHASH }}
        contentFit="cover"
        transition={250}
        style={styles.photo}
        accessibilityIgnoresInvertColors
      />
      <View style={styles.info}>
        <AppText variant="subtitle" numberOfLines={1}>
          {vet.name}
        </AppText>
        <AppText variant="caption" muted numberOfLines={1}>
          {vet.specialty}
        </AppText>
        <SpeciesIcons species={vet.species} size={14} />
        <RatingStars rating={vet.rating} size={12} />
      </View>
      {vet.isNew && (
        <View style={[styles.ribbon, { backgroundColor: theme.colors.primary }]}>
          <AppText variant="label" color={theme.colors.onPrimary}>
            New
          </AppText>
        </View>
      )}
    </Pressable>
  );
```

```
}

// memo: the card re-renders only when its props change
export const VetCard = memo(VetCardComponent);

const styles = StyleSheet.create({
  card: {
    flex: 1,
    borderRadius: radius.lg,
    borderWidth: StyleSheet.hairlineWidth, // the thinnest line on the given screen
    overflow: "hidden", // clips the image and ripple to the rounded corners
    boxShadow: "0 4px 14px rgba(0, 0, 0, 0.10)",
  },
  pressed: { opacity: 0.85, transform: [{ scale: 0.98 }] },
  photo: { width: "100%", aspectRatio: 1 },
  info: { padding: spacing.md, gap: 6 },
  ribbon: {
    position: "absolute",
    top: spacing.sm,
    left: spacing.sm,
    paddingHorizontal: spacing.sm,
    paddingVertical: 3,
    borderRadius: radius.sm,
  },
});
```

6. Vet screens

`VetsScreen` sets the header search bar with `navigation.setOptions` in `useLayoutEffect` and scrolls to the top when the tab is pressed again (`useScrollToTop`); touching a card preloads the profile. `FiltersScreen` returns its result with `navigation.popTo("Vets", filters, { merge: true })`. `VetVideoScreen` is a transparent modal – the previous screen is visible underneath.

src/screens/vets/VetsScreen.tsx

New file – vets: search in the header, filters in a sheet

```
// src/screens/vets/VetsScreen.tsx – vets: search in the header, filters in a sheet
import { useCallback, useLayoutEffect, useMemo, useRef, useState } from "react";
import { FlatList, Pressable, StyleSheet, View, useWindowDimensions } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { useScrollToTop } from "@react-navigation/native";
import { VETS } from "@/data/mockData";
import type { Vet } from "@/data/types";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { FeaturedCarousel } from "@/components/FeaturedCarousel";
import { VetCard } from "@/components/VetCard";
import type { VetsScreenProps } from "@/navigation/types";
import { spacing, useAppTheme } from "@/theme/theme";

export function VetsScreen({ navigation, route }: VetsScreenProps<"Vets">) {
  const theme = useAppTheme();
  const { width } = useWindowDimensions();
  const [query, setQuery] = useState("");
  const listRef = useRef<FlatList<Vet>>(null);

  // Tapping the active tab again scrolls the list to the top
  useScrollToTop(listRef);

  // Filters live in the route params – the Filters screen sets them
  // useMemo: without it "?? []" would create a new array on every render
  const specialties = useMemo(() => route.params?.specialties ?? [],
[route.params?.specialties]);
```

```
  const exoticOnly = route.params?.exoticOnly ?? false;
  const activeFilters = specialties.length + (exoticOnly ? 1 : 0);

  // Header options that depend on the screen state are set with setOptions
  useLayoutEffect(() => {
    navigation.setOptions({
      headerSearchBarOptions: {
        placeholder: "Search vets",
        onChangeText: (e) => setQuery(e.nativeEvent.text),
        hideWhenScrolling: true,
      },
      headerRight: () => (
        <Pressable
          onPress={() => navigation.navigate("Filters", { specialties, exoticOnly })}
          hitSlop={10}
          accessibilityRole="button"
          accessibilityLabel={`Filters, active: ${activeFilters}`}
          style={styles.filterButton}
        >
          <Ionicons name="options-outline" size={22} color={theme.colors.primary} />
          {activeFilters > 0 && (
            <View style={[styles.badge, { backgroundColor: theme.colors.primary }]}>
              <AppText variant="caption" color={theme.colors.onPrimary}>
                {activeFilters}
              </AppText>
            </View>
          )}
        </Pressable>
      ),
    });
  }, [navigation, specialties, exoticOnly, activeFilters, theme]);

  const numColumns = width >= 900 ? 4 : width >= 600 ? 3 : 2;
  const filtered = useMemo(
    () =>
      VETS.filter((v) => v.name.toLowerCase().includes(query.trim().toLowerCase()))
        .filter((v) => specialties.length === 0 || specialties.includes(v.specialty))
        .filter((v) => !exoticOnly || v.species.some((s) => s === "bird" || s === "reptile")),
    [query, specialties, exoticOnly],
  );

  const openVet = useCallback(
    (vet: Vet) => navigation.navigate("VetDetails", { vetId: vet.id }),
    [navigation],
  );
  // preload (v7): the details screen renders in the background as soon as the card is touched
  const preloadVet = useCallback(
    (vet: Vet) => navigation.preload("VetDetails", { vetId: vet.id }),
    [navigation],
  );

  return (
    <FlatList
      ref={listRef}
      key={numColumns}
      data={filtered}
      keyExtractor={(item) => item.id}
      numColumns={numColumns}
      // iOS: the list cooperates with the large title and the search bar
      contentInsetAdjustmentBehavior="automatic"
      renderItem={({ item }) => (
        <View style={[styles.cell, { maxWidth: `${100 / numColumns}%` }]}>
          <VetCard vet={item} onPress={openVet} onPressIn={preloadVet} />
        </View>
      )}
      ListHeaderComponent={
```

```
        query === "" ? (
          <View style={styles.header}>
            <FeaturedCarousel vets={VETS.filter((v) => v.isNew)} onPress={openVet} />
            <AppText variant="label" muted style={styles.count}>
              Vets: {filtered.length}
            </AppText>
          </View>
        ) : undefined
      }
      ListEmptyComponent={
        <EmptyState title="No vets found" message="Change the search or filters." />
      }
      columnWrapperStyle={styles.columns}
      contentContainerStyle={styles.content}
      keyboardDismissMode="on-drag"
    />
  );
}

const styles = StyleSheet.create({
  header: { gap: spacing.md, paddingTop: spacing.md, marginBottom: spacing.md },
  count: { marginHorizontal: spacing.lg },
  content: { paddingBottom: spacing.xxl },
  columns: { paddingHorizontal: spacing.lg - spacing.sm / 2 },
  cell: { flex: 1, padding: spacing.sm / 2, paddingBottom: spacing.md },
  filterButton: { padding: 4 },
  badge: {
    position: "absolute",
    top: -4,
    right: -6,
    minWidth: 18,
    height: 18,
    borderRadius: 9,
    alignItems: "center",
    justifyContent: "center",
    paddingHorizontal: 4,
  },
});
```

src/screens/vets/FiltersScreen.tsx

New file – a sheet (formSheet) that passes the result back

```
// src/screens/vets/FiltersScreen.tsx – a sheet (formSheet) that passes the result back
import { useState } from "react";
import { Pressable, StyleSheet, Switch, View } from "react-native";
import { SPECIALTIES } from "@/data/mockData";
import type { Specialty } from "@/data/types";
import { AppText } from "@/components/AppText";
import { SpecialtyChips } from "@/components/SpecialtyChips";
import type { VetsScreenProps } from "@/navigation/types";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function FiltersScreen({ navigation, route }: VetsScreenProps<"Filters">) {
  const theme = useAppTheme();
  const [specialties, setSpecialties] = useState<Specialty[]>(route.params?.specialties ?? []);
  const [exoticOnly, setExoticOnly] = useState(route.params?.exoticOnly ?? false);

  const toggle = (s: Specialty) =>
    setSpecialties((prev) => (prev.includes(s) ? prev.filter((x) => x !== s) : [...prev, s]));

  // popTo + merge: go back to the Vets screen and merge the new params with the current ones
  const apply = () => navigation.popTo("Vets", { specialties, exoticOnly }, { merge: true });

  return (
    <View style={styles.container}>
```

```
      <AppText variant="label" muted>
        Specialties
      </AppText>
      <View style={styles.chips}>
        <SpecialtyChips specialties={SPECIALTIES} selected={specialties} onToggle={toggle} />
      </View>
      <View style={styles.row}>
        <AppText>Birds & reptiles only</AppText>
        <Switch value={exoticOnly} onValueChange={setExoticOnly} />
      </View>
      <View style={styles.actions}>
        <Pressable
          style={[styles.button, { borderColor: theme.colors.border }]}
          onPress={() => {
            setSpecialties([]);
            setExoticOnly(false);
          }}
        >
          <AppText>Clear</AppText>
        </Pressable>
        <Pressable
          style={[styles.button, { backgroundColor: theme.colors.primary }]}
          onPress={apply}
        >
          <AppText color={theme.colors.onPrimary}>Apply</AppText>
        </Pressable>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { padding: spacing.lg, gap: spacing.lg },
  chips: { marginHorizontal: -spacing.lg },
  row: { flexDirection: "row", justifyContent: "space-between", alignItems: "center" },
  actions: { flexDirection: "row", gap: spacing.md },
  button: {
    flex: 1,
    height: 48,
    borderRadius: radius.md,
    borderWidth: 1,
    borderColor: "transparent",
    alignItems: "center",
    justifyContent: "center",
  },
});
```

src/screens/vets/VetDetailsScreen.tsx

New file – vet profile + upcoming working days

```
// src/screens/vets/VetDetailsScreen.tsx – vet profile + upcoming working days
import { Pressable, ScrollView, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import { LinearGradient } from "expo-linear-gradient";
import { Ionicons } from "@expo/vector-icons";
import { SCHEDULES, findClinic, findVet } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { RatingStars } from "@/components/RatingStars";
import { ScheduleRow } from "@/components/ScheduleRow";
import { SpeciesIcons } from "@/components/SpeciesIcons";
import type { VetsScreenProps } from "@/navigation/types";
import { formatDayLabel, formatYears } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function VetDetailsScreen({ navigation, route }: VetsScreenProps<"VetDetails">) {
```

```
  const theme = useAppTheme();
  const vet = findVet(route.params.vetId);

  if (!vet) {
    return <EmptyState title="Vet not found" message="The vet may no longer work with us." />;
  }

  const schedules = SCHEDULES.filter((s) => s.vetId === vet.id);
  const clinic = findClinic(vet.clinicId);

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <View>
        <Image source={vet.coverUrl} style={styles.cover} contentFit="cover" />
        <LinearGradient
          colors={["rgba(0,0,0,0.6)", "transparent", theme.colors.background]}
          style={StyleSheet.absoluteFill}
        />
      </View>
      <View style={styles.body}>
        <AppText variant="hero">{vet.name}</AppText>
        <AppText variant="subtitle" color={theme.colors.primary}>
          {vet.specialty}
        </AppText>
        <View style={styles.meta}>
          <RatingStars rating={vet.rating} reviews={vet.reviews} />
          <SpeciesIcons species={vet.species} />
        </View>
        <AppText muted>
          {formatYears(vet.experienceYears)} of experience · {clinic?.name}
        </AppText>
        <AppText>{vet.bio}</AppText>

        <Pressable
          // The VetVideo screen lives in the root Stack – navigate "looks" in parent navigators
          onPress={() => navigation.navigate("VetVideo", { vetId: vet.id })}
          style={[styles.video, { borderColor: theme.colors.primary }]}
        >
          <Ionicons name="play-circle" size={22} color={theme.colors.primary} />
          <AppText variant="subtitle" color={theme.colors.primary}>
            Meet {vet.name.split(" ")[1]}
          </AppText>
        </Pressable>

        <AppText variant="title">Working days</AppText>
      </View>
      {schedules.map((s) =>
        clinic ? (
          <View key={s.id} style={styles.schedule}>
            <AppText variant="label" muted style={styles.day}>
              {formatDayLabel(s.date)}
            </AppText>
            <ScheduleRow
              schedule={s}
              vet={vet}
              clinic={clinic}
              onPress={() => navigation.navigate("BookAppointment", { scheduleId: s.id })}
            />
          </View>
        ) : null,
      )}
    </ScrollView>
  );
}

const styles = StyleSheet.create({
```

```
  content: { paddingBottom: spacing.xxl },
  cover: { width: "100%", aspectRatio: 4 / 3 },
  body: { padding: spacing.lg, gap: spacing.md },
  meta: { flexDirection: "row", alignItems: "center", gap: spacing.lg, flexWrap: "wrap" },
  video: {
    flexDirection: "row",
    alignItems: "center",
    alignSelf: "flex-start",
    gap: spacing.sm,
    paddingHorizontal: spacing.lg,
    paddingVertical: spacing.sm,
    borderWidth: 1.5,
    borderRadius: radius.pill,
  },
  schedule: { marginBottom: spacing.sm },
  day: { marginHorizontal: spacing.lg, marginBottom: spacing.sm },
});
```

src/screens/vets/VetVideoScreen.tsx

New file – transparent modal over the previous screen

```
// src/screens/vets/VetVideoScreen.tsx – transparent modal over the previous screen
import { Pressable, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import { Ionicons } from "@expo/vector-icons";
import { findVet } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import type { RootStackScreenProps } from "@/navigation/types";
import { radius, spacing } from "@/theme/theme";

export function VetVideoScreen({ navigation, route }: RootStackScreenProps<"VetVideo">) {
  const vet = findVet(route.params.vetId);

  return (
    <View style={styles.overlay}>
      {/* Tapping the background closes the modal */}
      <Pressable style={StyleSheet.absoluteFill} onPress={() => navigation.goBack()} />
      <View style={styles.card}>
        <Image source={vet?.coverUrl} style={styles.video} contentFit="cover" />
        <View style={styles.play}>
          <Ionicons name="play" size={36} color="#FFFFFF" />
        </View>
        <AppText variant="subtitle" color="#FFFFFF" style={styles.title}>
          {vet ? `Meet ${vet.name}` : "Video"}
        </AppText>
        <AppText variant="caption" color="#D1D5DB" style={styles.title}>
          The video player comes in L07 – for now we show a modal presentation.
        </AppText>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  overlay: {
    flex: 1,
    backgroundColor: "rgba(0,0,0,0.7)",
    justifyContent: "center",
    padding: spacing.lg,
  },
  card: {
    backgroundColor: "#0F1716",
    borderRadius: radius.lg,
    overflow: "hidden",
    paddingBottom: spacing.md,
```

```
  },
  video: { width: "100%", aspectRatio: 16 / 9 },
  play: {
    position: "absolute",
    top: 0,
    left: 0,
    right: 0,
    aspectRatio: 16 / 9,
    alignItems: "center",
    justifyContent: "center",
  },
  title: { paddingHorizontal: spacing.lg, paddingTop: spacing.md },
});
```

7. Top tabs with days

Three tabs (Today, Tomorrow, Day after) reuse one screen component with different `initialParams`. Tabs can be swiped; screens load lazily.

src/navigation/ScheduleTopTabs.tsx

New file – top tabs (swipeable) with days

```
// src/navigation/ScheduleTopTabs.tsx – top tabs (swipeable) with days
import { ActivityIndicator } from "react-native";
import { createMaterialTopTabNavigator } from "@react-navigation/material-top-tabs";
import { fonts, useAppTheme } from "@/theme/theme";
import type { ScheduleTopTabParamList } from "./types";
import { ScheduleDayScreen } from "@/screens/schedule/ScheduleDayScreen";

const TopTab = createMaterialTopTabNavigator<ScheduleTopTabParamList>();

export function ScheduleTopTabs() {
  const theme = useAppTheme();
  return (
    <TopTab.Navigator
      screenOptions={{
        tabBarActiveTintColor: theme.colors.primary,
        tabBarInactiveTintColor: theme.colors.textMuted,
        tabBarIndicatorStyle: { backgroundColor: theme.colors.primary, height: 3 },
        tabBarLabelStyle: { fontFamily: fonts.semibold, textTransform: "none" },
        swipeEnabled: true, // switch tabs with a swipe
        lazy: true, // a screen is created only when first visited
        lazyPlaceholder: () => <ActivityIndicator style={{ marginTop: 32 }} />,
      }}
    >
      <TopTab.Screen
        name="Today"
        component={ScheduleDayScreen}
        initialParams={{ dayOffset: 0 }} // the initial parameter of the screen
        options={{ title: "Today" }}
      />
      <TopTab.Screen
        name="Tomorrow"
        component={ScheduleDayScreen}
        initialParams={{ dayOffset: 1 }}
        options={{ title: "Tomorrow" }}
      />
      <TopTab.Screen
        name="DayAfter"
        component={ScheduleDayScreen}
        initialParams={{ dayOffset: 2 }}
        options={{ title: "Day after" }}
      />
    </TopTab.Navigator>
  );
```

```
}
```

src/screens/schedule/ScheduleDayScreen.tsx

New file – vets on duty on one day (top tab)

```
// src/screens/schedule/ScheduleDayScreen.tsx – vets on duty on one day (top tab)
import { useMemo } from "react";
import { FlatList, StyleSheet, View } from "react-native";
import { useIsFocused } from "@react-navigation/native";
import { SCHEDULES, findClinic, findVet } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { ScheduleRow } from "@/components/ScheduleRow";
import type { ScheduleTopTabParamList, ScheduleTopTabScreenProps } from "@/navigation/types";
import { formatDayLabel, startOfDay } from "@/utils/format";
import { spacing } from "@/theme/theme";

type Props = ScheduleTopTabScreenProps<keyof ScheduleTopTabParamList>;

export function ScheduleDayScreen({ navigation, route }: Props) {
  const { dayOffset } = route.params;
  const isFocused = useIsFocused(); // is this tab currently visible

  const day = useMemo(() => {
    const d = startOfDay(new Date());
    d.setDate(d.getDate() + dayOffset);
    return d;
  }, [dayOffset]);

  const schedules = useMemo(
    () => SCHEDULES.filter((s) => new Date(s.date).getTime() === day.getTime()),
    [day],
  );

  return (
    <FlatList
      data={schedules}
      keyExtractor={(s) => s.id}
      renderItem={({ item }) => {
        const vet = findVet(item.vetId);
        const clinic = findClinic(item.clinicId);
        return vet && clinic ? (
          <ScheduleRow
            schedule={item}
            vet={vet}
            clinic={clinic}
            onPress={() => navigation.navigate("BookAppointment", { scheduleId: item.id })}
          />
        ) : null;
      }}
      ListHeaderComponent={
        <View style={styles.header}>
          <AppText variant="title">{formatDayLabel(day.toISOString())}</AppText>
          {isFocused && (
            <AppText variant="caption" muted>
              Swipe sideways to change the day
            </AppText>
          )}
        </View>
      }
      ItemSeparatorComponent={() => <View style={styles.separator} />}
      ListEmptyComponent={<EmptyState icon="calendar-outline" title="No vets on duty" />}
      contentContainerStyle={styles.content}
    />
  );
```

```
}

const styles = StyleSheet.create({
  header: { padding: spacing.lg, gap: 2 },
  separator: { height: spacing.sm },
  content: { paddingBottom: spacing.xxl },
});
```

8. Visits and profile

`AppointmentDetailsScreen` navigates to a nested screen in another tab (the full path `navigate("Main", { screen: …, params: … })`) and resets the history with `CommonActions.reset`. `ProfileScreen` collects examples of moving between navigators, including `DrawerActions.openDrawer()`.

src/screens/appointments/AppointmentsScreen.tsx

New file – the list of booked visits

```
// src/screens/appointments/AppointmentsScreen.tsx – the list of booked visits
import { useState } from "react";
import { FlatList, Pressable, StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { SCHEDULES, findService, findVet } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { SegmentedControl } from "@/components/SegmentedControl";
import type { TabScreenProps } from "@/navigation/types";
import { useAppointments } from "@/state/AppointmentsContext";
import { formatDayLabel, formatPrice, formatTime } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

type Filter = "upcoming" | "past";

export function AppointmentsScreen({ navigation }: TabScreenProps<"AppointmentsTab">) {
  const theme = useAppTheme();
  const { appointments } = useAppointments();
  const [filter, setFilter] = useState<Filter>("upcoming");

  // Read the time once (lazy initializer) – rendering must be "pure"
  const [now] = useState(() => Date.now());
  const visible = appointments.filter((a) => {
    const isPast = new Date(a.slot).getTime() < now;
    return filter === "past" ? isPast : !isPast;
  });

  return (
    <View style={styles.flex}>
      <View style={styles.segmented}>
        <SegmentedControl<Filter>
          options={[
            { value: "upcoming", label: "Upcoming" },
            { value: "past", label: "Past" },
          ]}
          value={filter}
          onChange={setFilter}
        />
      </View>
      <FlatList
        data={visible}
        keyExtractor={(a) => a.id}
        contentContainerStyle={styles.content}
        ItemSeparatorComponent={() => <View style={styles.separator} />}
        ListEmptyComponent={<EmptyState icon="paw-outline" title="No visits" />}
```

```
        renderItem={({ item }) => {
          const schedule = SCHEDULES.find((s) => s.id === item.scheduleId);
          const vet = schedule && findVet(schedule.vetId);
          const service = findService(item.serviceId);
          return (
            <Pressable
              onPress={() => navigation.navigate("AppointmentDetails", { appointmentId: item.id
})}
              style={[styles.card, { backgroundColor: theme.colors.surface }]}
            >
              <Ionicons name="paw" size={28} color={theme.colors.primary} />
              <View style={styles.flex}>
                <AppText variant="subtitle">{service?.name}</AppText>
                <AppText variant="caption" muted>
                  {formatDayLabel(item.slot)}, {formatTime(item.slot)} · {vet?.name}
                </AppText>
              </View>
              <AppText variant="subtitle">{formatPrice(item.total)}</AppText>
            </Pressable>
          );
        }}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  segmented: { paddingVertical: spacing.md },
  content: { paddingHorizontal: spacing.lg, paddingBottom: spacing.xxl },
  separator: { height: spacing.sm },
  card: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.lg,
    borderRadius: radius.md,
  },
});
```

src/screens/appointments/AppointmentDetailsScreen.tsx

New file – visit details + nested navigation

```
// src/screens/appointments/AppointmentDetailsScreen.tsx – visit details + nested navigation
import { Pressable, ScrollView, StyleSheet, View } from "react-native";
import { CommonActions } from "@react-navigation/native";
import { SCHEDULES, findClinic, findService, findVet } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import type { RootStackScreenProps } from "@/navigation/types";
import { useAppointments } from "@/state/AppointmentsContext";
import { formatDayLabel, formatPrice, formatTime } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function AppointmentDetailsScreen({
  navigation,
  route,
}: RootStackScreenProps<"AppointmentDetails">) {
  const theme = useAppTheme();
  const { appointments } = useAppointments();
  const appointment = appointments.find((a) => a.id === route.params.appointmentId);
  const schedule = appointment && SCHEDULES.find((s) => s.id === appointment.scheduleId);
  const vet = schedule && findVet(schedule.vetId);
  const clinic = schedule && findClinic(schedule.clinicId);
  const service = appointment && findService(appointment.serviceId);
```

```
  if (!appointment || !vet || !clinic || !service) {
    return <EmptyState icon="paw-outline" title="Visit not found" />;
  }

  // A full path to a screen in a deeply nested navigator
  const openVet = () =>
    navigation.navigate("Main", {
      screen: "HomeTabs",
      params: { screen: "VetsTab", params: { screen: "VetDetails", params: { vetId: vet.id } } }
},
    });

  // reset: a new history – the visit list as the only screen underneath
  const goToVisits = () =>
    navigation.dispatch(
      CommonActions.reset({
        index: 0,
        routes: [
          { name: "Main", params: { screen: "HomeTabs", params: { screen: "AppointmentsTab" } } }
},
        ],
      }),
    );

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <View style={[styles.card, { backgroundColor: theme.colors.surface }]}>
        <AppText variant="label" color={theme.colors.primary}>
          Visit {appointment.id}
        </AppText>
        <AppText variant="hero">{service.name}</AppText>
        <Row
          label="When"
          value={`${formatDayLabel(appointment.slot)}, ${formatTime(appointment.slot)}`}
        />
        <Row label="Vet" value={vet.name} />
        <Row label="Clinic" value={`${clinic.name}, ${clinic.address}`} />
        <Row label="Price" value={formatPrice(appointment.total)} />
        <AppText variant="caption" muted>
          A check-in QR code is added in L07.
        </AppText>
      </View>
      <Pressable style={[styles.button, { borderColor: theme.colors.primary }]}
onPress={openVet}>
        <AppText color={theme.colors.primary}>Vet profile</AppText>
      </Pressable>
      <Pressable
        style={[styles.button, { backgroundColor: theme.colors.primary }]}
        onPress={goToVisits}
      >
        <AppText color={theme.colors.onPrimary}>My visits</AppText>
      </Pressable>
    </ScrollView>
  );
}

function Row({ label, value }: { label: string; value: string }) {
  return (
    <View style={styles.row}>
      <AppText muted>{label}</AppText>
      <AppText variant="subtitle" style={styles.value}>
        {value}
      </AppText>
    </View>
  );
```

```
}

const styles = StyleSheet.create({
  content: { padding: spacing.lg, gap: spacing.md },
  card: { padding: spacing.xl, borderRadius: radius.lg, gap: spacing.md },
  row: { flexDirection: "row", justifyContent: "space-between", gap: spacing.md },
  value: { flexShrink: 1, textAlign: "right" },
  button: {
    height: 50,
    borderRadius: radius.md,
    borderWidth: 1.5,
    borderColor: "transparent",
    alignItems: "center",
    justifyContent: "center",
  },
});
```

src/screens/profile/ProfileScreen.tsx

New file – profile + examples of navigating between navigators

```
// src/screens/profile/ProfileScreen.tsx – profile + examples of navigating between navigators
import { Pressable, ScrollView, StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { DrawerActions } from "@react-navigation/native";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import type { TabScreenProps } from "@/navigation/types";
import { useAppointments } from "@/state/AppointmentsContext";
import { radius, spacing, useAppTheme } from "@/theme/theme";

type IconName = keyof typeof Ionicons.glyphMap;

export function ProfileScreen({ navigation }: TabScreenProps<"ProfileTab">) {
  const theme = useAppTheme();
  const { user, signOut } = useAuth();
  const { appointments } = useAppointments();

  const items: { icon: IconName; label: string; onPress: () => void }[] = [
    // Another tab of the same navigator
    {
      icon: "paw-outline",
      label: `My visits (${appointments.length})`,
      onPress: () => navigation.navigate("AppointmentsTab"),
    },
    // A screen of the parent navigator (Drawer) – navigate goes up the tree
    { icon: "settings-outline", label: "Settings", onPress: () =>
navigation.navigate("Settings") },
    // An action of the parent navigator sent with dispatch
    {
      icon: "menu-outline",
      label: "Open menu",
      onPress: () => navigation.dispatch(DrawerActions.openDrawer()),
    },
    // A nested screen in another tab
    {
      icon: "calendar-outline",
      label: "Tomorrow's schedule",
      onPress: () =>
        navigation.navigate("ScheduleTab", { screen: "Tomorrow", params: { dayOffset: 1 } }),
    },
  ];

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <View style={styles.header}>
```

```
        <View style={[styles.avatar, { backgroundColor: theme.colors.primary }]}>
          <AppText variant="hero" color={theme.colors.onPrimary}>
            {user?.name.charAt(0).toUpperCase()}
          </AppText>
        </View>
        <AppText variant="title">{user?.name}</AppText>
        <AppText muted>{user?.isGuest ? "Guest account" : user?.email}</AppText>
      </View>
      {items.map((item) => (
        <Pressable
          key={item.label}
          onPress={item.onPress}
          style={[styles.item, { backgroundColor: theme.colors.surface }]}
        >
          <Ionicons name={item.icon} size={22} color={theme.colors.primary} />
          <AppText style={styles.flex}>{item.label}</AppText>
          <Ionicons name="chevron-forward" size={18} color={theme.colors.textMuted} />
        </Pressable>
      ))}
      {/* Signing out changes the state -> RootNavigator removes the app screens */}
      <Pressable onPress={signOut} style={[styles.item, { backgroundColor: theme.colors.surface
}]}>
        <Ionicons name="log-out-outline" size={22} color={theme.colors.danger} />
        <AppText color={theme.colors.danger}>Sign out</AppText>
      </Pressable>
    </ScrollView>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { padding: spacing.lg, gap: spacing.sm },
  header: { alignItems: "center", gap: 4, marginBottom: spacing.lg },
  avatar: {
    width: 88,
    height: 88,
    borderRadius: 44,
    alignItems: "center",
    justifyContent: "center",
    marginBottom: spacing.sm,
  },
  item: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.lg,
    borderRadius: radius.md,
  },
});
```

9. Bottom tabs

Icons depend on `focused`, the Visits tab shows a badge, `backBehavior="history"` returns to the previously visited tab and the Vets tab returns to its list on blur (`popToTopOnBlur`). For a guest the Visits tab is blocked: `listeners.tabPress` calls `e.preventDefault()` and shows a message.

src/navigation/HomeTabs.tsx

New file – bottom tabs

```
// src/navigation/HomeTabs.tsx – bottom tabs
import { Alert } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import { DrawerToggleButton } from "@react-navigation/drawer";
import { useAppointments } from "@/state/AppointmentsContext";
```

```
import { useAppTheme } from "@/theme/theme";
import { VetsStack } from "./VetsStack";
import { ScheduleTopTabs } from "./ScheduleTopTabs";
import type { TabParamList } from "./types";
import { AppointmentsScreen } from "@/screens/appointments/AppointmentsScreen";
import { ProfileScreen } from "@/screens/profile/ProfileScreen";

const Tab = createBottomTabNavigator<TabParamList>();

type IconName = keyof typeof Ionicons.glyphMap;
const ICONS: Record<keyof TabParamList, [IconName, IconName]> = {
  VetsTab: ["medkit", "medkit-outline"],
  ScheduleTab: ["calendar", "calendar-outline"],
  AppointmentsTab: ["paw", "paw-outline"],
  ProfileTab: ["person-circle", "person-circle-outline"],
};

export function HomeTabs() {
  const theme = useAppTheme();
  const { appointments } = useAppointments();

  return (
    <Tab.Navigator
      backBehavior="history" // "back" returns to the previously visited tab
      screenOptions={({ route }) => ({
        tabBarIcon: ({ focused, color, size }) => {
          const [active, inactive] = ICONS[route.name];
          return <Ionicons name={focused ? active : inactive} size={size} color={color} />;
        },
        tabBarActiveTintColor: theme.colors.primary,
        tabBarInactiveTintColor: theme.colors.textMuted,
        tabBarHideOnKeyboard: true, // Android: the bar hides when the keyboard appears
        animation: "shift", // animation when switching tabs
        headerLeft: () => <DrawerToggleButton tintColor={theme.colors.primary} />,
      })}
      // Listening to events of all tabs
      screenListeners={{
        tabPress: (e) => console.log("[tab]", e.target),
      }}
    >
      <Tab.Screen
        name="VetsTab"
        component={VetsStack}
        options={{
          title: "Vets",
          headerShown: false, // the header comes from the nested Stack
          popToTopOnBlur: true, // leaving the tab returns it to the vet list
        }}
      />
      <Tab.Screen name="ScheduleTab" component={ScheduleTopTabs} options={{ title: "Schedule" }}
/>
      <Tab.Screen
        name="AppointmentsTab"
        component={AppointmentsScreen}
        options={{
          title: "Visits",
          tabBarBadge: appointments.length > 0 ? appointments.length : undefined,
        }}
        // An event for this tab only – it can be cancelled
        listeners={{
          tabPress: (e) => {
            if (appointments.length === 0) {
              e.preventDefault();
              Alert.alert("No visits yet", "Book your first visit in the Schedule tab.");
            }
          },
```

```
        }}
      />
      <Tab.Screen name="ProfileTab" component={ProfileScreen} options={{ title: "Profile" }} />
    </Tab.Navigator>
  );
}
```

10. Side menu (Drawer)

Custom menu content: a header with the user, the standard screen list (`DrawerItemList`) and an extra "Sign out" item (`DrawerItem`). Menu screens: clinics, settings, about.

src/navigation/MainDrawer.tsx

New file – side menu with custom content

```
// src/navigation/MainDrawer.tsx – side menu with custom content
import { StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import {
  DrawerContentScrollView,
  DrawerItem,
  DrawerItemList,
  createDrawerNavigator,
  type DrawerContentComponentProps,
} from "@react-navigation/drawer";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import { spacing, useAppTheme } from "@/theme/theme";
import { HomeTabs } from "./HomeTabs";
import type { DrawerParamList } from "./types";
import { ClinicsScreen } from "@/screens/drawer/ClinicsScreen";
import { SettingsScreen } from "@/screens/drawer/SettingsScreen";
import { AboutScreen } from "@/screens/drawer/AboutScreen";

const Drawer = createDrawerNavigator<DrawerParamList>();

function DrawerContent(props: DrawerContentComponentProps) {
  const theme = useAppTheme();
  const { user, signOut } = useAuth();
  return (
    <DrawerContentScrollView {...props}>
      <View style={styles.header}>
        <View style={[styles.avatar, { backgroundColor: theme.colors.primary }]}>
          <AppText variant="title" color={theme.colors.onPrimary}>
            {user?.name.charAt(0).toUpperCase() ?? "?"}
          </AppText>
        </View>
        <AppText variant="subtitle">{user?.name}</AppText>
        <AppText variant="caption" muted>
          {user?.isGuest ? "Guest" : user?.email}
        </AppText>
      </View>
      {/* Standard items generated from the navigator's screens */}
      <DrawerItemList {...props} />
      <DrawerItem
        label="Sign out"
        icon={({ size }) => (
          <Ionicons name="log-out-outline" size={size} color={theme.colors.danger} />
        )}
        labelStyle={{ color: theme.colors.danger }}
        onPress={signOut}
      />
    </DrawerContentScrollView>
  );
}
```

```
export function MainDrawer() {
  const theme = useAppTheme();
  return (
    <Drawer.Navigator
      drawerContent={(props) => <DrawerContent {...props} />}
      screenOptions={{
        drawerType: "front", // "slide" also moves the screen, "permanent" – fixed menu (tablet)
        drawerActiveTintColor: theme.colors.primary,
        drawerActiveBackgroundColor: theme.dark ? "#123430" : "#E2F5F2",
        swipeEdgeWidth: 40, // area at the edge from which the menu can be swiped open
        headerTintColor: theme.colors.primary,
      }}
    >
      <Drawer.Screen
        name="HomeTabs"
        component={HomeTabs}
        options={{
          title: "Home",
          headerShown: false, // headers come from the tabs and their stacks
          drawerIcon: ({ color, size }) => (
            <Ionicons name="home-outline" size={size} color={color} />
          ),
        }}
      />
      <Drawer.Screen
        name="Clinics"
        component={ClinicsScreen}
        options={{
          title: "Our clinics",
          drawerIcon: ({ color, size }) => (
            <Ionicons name="business-outline" size={size} color={color} />
          ),
        }}
      />
      <Drawer.Screen
        name="Settings"
        component={SettingsScreen}
        options={{
          title: "Settings",
          drawerIcon: ({ color, size }) => (
            <Ionicons name="settings-outline" size={size} color={color} />
          ),
        }}
      />
      <Drawer.Screen
        name="About"
        component={AboutScreen}
        options={{
          title: "About",
          drawerIcon: ({ color, size }) => (
            <Ionicons name="information-circle-outline" size={size} color={color} />
          ),
        }}
      />
    </Drawer.Navigator>
  );
}

const styles = StyleSheet.create({
  header: { padding: spacing.lg, gap: 4 },
  avatar: {
    width: 56,
    height: 56,
    borderRadius: 28,
    alignItems: "center",
    justifyContent: "center",
```

```
    marginBottom: spacing.sm,
  },
});
```

src/screens/drawer/ClinicsScreen.tsx

New file – the list of clinics (a side-menu screen)

```
// src/screens/drawer/ClinicsScreen.tsx – the list of clinics (a side-menu screen)
import { FlatList, StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { CLINICS } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function ClinicsScreen() {
  const theme = useAppTheme();
  return (
    <FlatList
      data={CLINICS}
      keyExtractor={(c) => c.id}
      contentContainerStyle={styles.content}
      ItemSeparatorComponent={() => <View style={styles.separator} />}
      renderItem={({ item }) => (
        <View style={[styles.card, { backgroundColor: theme.colors.surface }]}>
          <Ionicons
            name={item.isEmergency ? "medical" : "business"}
            size={32}
            color={item.isEmergency ? theme.colors.danger : theme.colors.primary}
          />
          <View style={styles.flex}>
            <AppText variant="subtitle">{item.name}</AppText>
            <AppText variant="caption" muted>
              {item.address} · {item.rooms} rooms{item.isEmergency ? " · open 24/7" : ""}
            </AppText>
          </View>
        </View>
      )}
    />
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { padding: spacing.lg },
  separator: { height: spacing.sm },
  card: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.lg,
    borderRadius: radius.md,
  },
});
```

src/screens/drawer/SettingsScreen.tsx

New file – settings (working options come in L04)

```
// src/screens/drawer/SettingsScreen.tsx – settings (working options come in L04)
import { useState } from "react";
import { ScrollView, StyleSheet, Switch, View } from "react-native";
import { AppText } from "@/components/AppText";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function SettingsScreen() {
```

```
  const theme = useAppTheme();
  const [reminders, setReminders] = useState(true);
  const [tips, setTips] = useState(false);

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <View style={[styles.card, { backgroundColor: theme.colors.surface }]}>
        <Row label="Visit reminders" value={reminders} onChange={setReminders} />
        <Row label="Pet care tips by email" value={tips} onChange={setTips} />
      </View>
      <AppText variant="caption" muted>
        Theme, favorites and persisted settings come in L04.
      </AppText>
    </ScrollView>
  );
}

function Row({
  label,
  value,
  onChange,
}: {
  label: string;
  value: boolean;
  onChange: (v: boolean) => void;
}) {
  return (
    <View style={styles.row}>
      <AppText style={styles.flex}>{label}</AppText>
      <Switch value={value} onValueChange={onChange} />
    </View>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { padding: spacing.lg, gap: spacing.md },
  card: { borderRadius: radius.md, paddingHorizontal: spacing.lg },
  row: { flexDirection: "row", alignItems: "center", minHeight: 56, gap: spacing.md },
});
```

src/screens/drawer/AboutScreen.tsx

New file – info + outgoing links (Linking)

```
// src/screens/drawer/AboutScreen.tsx – info + outgoing links (Linking)
import { Alert, Linking, Pressable, ScrollView, StyleSheet } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import * as ExpoLinking from "expo-linking";
import { AppText } from "@/components/AppText";
import { radius, spacing, useAppTheme } from "@/theme/theme";

type IconName = keyof typeof Ionicons.glyphMap;

export function AboutScreen() {
  const theme = useAppTheme();

  // Linking.openURL – open another app (browser, phone, email)
  const open = async (url: string) => {
    const supported = await Linking.canOpenURL(url);
    if (supported) await Linking.openURL(url);
    else Alert.alert("Cannot open", url);
  };

  const links: { icon: IconName; label: string; url: string }[] = [
```

```
    { icon: "globe-outline", label: "Website", url: "https://example.com" },
    { icon: "call-outline", label: "Call reception", url: "tel:+15551234567" },
    { icon: "mail-outline", label: "Email us", url: "mailto:hello@example.com" },
  ];

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <AppText variant="title">PawCare Clinic</AppText>
      <AppText muted>A sample app for the “Mobile Applications” lecture.</AppText>
      {links.map((l) => (
        <Pressable
          key={l.url}
          onPress={() => open(l.url)}
          style={[styles.item, { backgroundColor: theme.colors.surface }]}
        >
          <Ionicons name={l.icon} size={22} color={theme.colors.primary} />
          <AppText>{l.label}</AppText>
        </Pressable>
      ))}
      <AppText variant="caption" muted selectable>
        Deep link to this screen: {ExpoLinking.createURL("about")}
      </AppText>
    </ScrollView>
  );
}

const styles = StyleSheet.create({
  content: { padding: spacing.lg, gap: spacing.md },
  item: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.lg,
    borderRadius: radius.md,
  },
});
```

11. Sign-in, booking and the 404 screen

Screens for signed-out users (welcome, simplified sign-in – the full form comes in lecture 5). `BookAppointmentScreen` is a modal with a leave-guard: when a slot is selected, `usePreventRemove` intercepts closing (gesture, button, Back) and asks for confirmation. Booking calls `navigation.replace` – the visit replaces the booking modal, so Back does not return to it.

src/screens/auth/WelcomeScreen.tsx

New file – welcome screen (signed-out users only)

```
// src/screens/auth/WelcomeScreen.tsx – welcome screen (signed-out users only)
import { Pressable, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import { LinearGradient } from "expo-linear-gradient";
import { SafeAreaView } from "react-native-safe-area-context";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import type { RootStackScreenProps } from "@/navigation/types";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function WelcomeScreen({ navigation }: RootStackScreenProps<"Welcome">) {
  const theme = useAppTheme();
  const { signIn } = useAuth();

  return (
    <View style={styles.flex}>
```

```
      <Image
        source="https://picsum.photos/seed/pawcare-welcome/1200/1800"
        style={StyleSheet.absoluteFill}
        contentFit="cover"
      />
      <LinearGradient colors={["rgba(0,0,0,0.1)", "#000000"]} style={StyleSheet.absoluteFill} />
      <SafeAreaView style={[styles.flex, styles.content]}>
        <AppText variant="label" color={theme.colors.primary}>
          PawCare Clinic
        </AppText>
        <AppText variant="hero" color="#FFFFFF">
          Healthy pets, happy homes
        </AppText>
        <AppText color="#D1D5DB">
          Find a vet, book a visit and keep your pet’s records in one place.
        </AppText>
        <Pressable
          style={[styles.button, { backgroundColor: theme.colors.primary }]}
          onPress={() => navigation.navigate("SignIn")}
        >
          <AppText variant="subtitle" color={theme.colors.onPrimary}>
            Sign in
          </AppText>
        </Pressable>
        {/* Changing the auth state switches the screen set in RootNavigator by itself */}
        <Pressable
          style={[styles.button, styles.ghost]}
          onPress={() => signIn({ name: "Guest", isGuest: true })}
        >
          <AppText variant="subtitle" color="#FFFFFF">
            Continue as guest
          </AppText>
        </Pressable>
      </SafeAreaView>
    </View>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { justifyContent: "flex-end", padding: spacing.xl, gap: spacing.md },
  button: { height: 52, borderRadius: radius.md, alignItems: "center", justifyContent: "center"
},
  ghost: { borderWidth: 1, borderColor: "rgba(255,255,255,0.4)" },
});
```

src/screens/auth/SignInScreen.tsx

New file – simplified sign-in (full form in L05)

```
// src/screens/auth/SignInScreen.tsx – simplified sign-in (full form in L05)
import { useState } from "react";
import { Pressable, StyleSheet, TextInput, View } from "react-native";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import type { RootStackScreenProps } from "@/navigation/types";
import { fonts, radius, spacing, useAppTheme } from "@/theme/theme";

export function SignInScreen({ route }: RootStackScreenProps<"SignIn">) {
  const theme = useAppTheme();
  const { signIn } = useAuth();
  // Optional parameter (e.g. from the deep link pawcare://login?email=...)
  const [email, setEmail] = useState(route.params?.email ?? "");
  const canSubmit = email.includes("@");

  return (
```

```
    <View style={styles.container}>
      <AppText variant="title">Welcome back</AppText>
      <TextInput
        value={email}
        onChangeText={setEmail}
        placeholder="Email"
        placeholderTextColor={theme.colors.textMuted}
        keyboardType="email-address"
        autoCapitalize="none"
        style={[
          styles.input,
          {
            color: theme.colors.text,
            borderColor: theme.colors.border,
            backgroundColor: theme.colors.surface,
          },
        ]}
      />
      <Pressable
        disabled={!canSubmit}
        onPress={() => signIn({ name: email.split("@")[0], email, isGuest: false })}
        style={[styles.button, { backgroundColor: theme.colors.primary }, !canSubmit &&
styles.dim]}
      >
        <AppText variant="subtitle" color={theme.colors.onPrimary}>
          Sign in
        </AppText>
      </Pressable>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: spacing.xl, gap: spacing.lg },
  input: {
    height: 50,
    borderWidth: 1,
    borderRadius: radius.md,
    paddingHorizontal: spacing.md,
    fontFamily: fonts.regular,
    fontSize: 16,
  },
  button: { height: 50, borderRadius: radius.md, alignItems: "center", justifyContent: "center"
},
  dim: { opacity: 0.5 },
});
```

src/screens/booking/BookAppointmentScreen.tsx

New file – booking (modal) that guards against leaving

```
// src/screens/booking/BookAppointmentScreen.tsx – booking (modal) that guards against leaving
import { useState } from "react";
import { Alert, Pressable, ScrollView, StyleSheet, View } from "react-native";
import { usePreventRemove } from "@react-navigation/native";
import { useSafeAreaInsets } from "react-native-safe-area-context";
import { SCHEDULES, SERVICES, findClinic, findService, findVet } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { SlotGrid } from "@/components/SlotGrid";
import type { RootStackScreenProps } from "@/navigation/types";
import { useAppointments } from "@/state/AppointmentsContext";
import { formatDayLabel, formatPrice, formatTime } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function BookAppointmentScreen({
  navigation,
```

```
  route,
}: RootStackScreenProps<"BookAppointment">) {
  const theme = useAppTheme();
  const insets = useSafeAreaInsets();
  const { addAppointment } = useAppointments();
  const [serviceId, setServiceId] = useState(SERVICES[0].id);
  const [slot, setSlot] = useState<string | null>(null);

  // Prevent the screen from being closed by accident with an unsaved choice
  usePreventRemove(slot !== null, ({ data }) => {
    // Our own "replace" after booking is let through without asking (state would not update in
time)
    if (data.action.type === "REPLACE") {
      navigation.dispatch(data.action);
      return;
    }
    Alert.alert("Discard booking?", "The selected time will not be reserved.", [
      { text: "Stay", style: "cancel" },
      // Perform the navigation action that we blocked earlier
      { text: "Discard", style: "destructive", onPress: () => navigation.dispatch(data.action)
},
    ]);
  });

  const schedule = SCHEDULES.find((s) => s.id === route.params.scheduleId);
  const vet = schedule && findVet(schedule.vetId);
  const clinic = schedule && findClinic(schedule.clinicId);
  const service = findService(serviceId);
  if (!schedule || !vet || !clinic || !service) {
    return (
      <EmptyState title="Schedule not found" message="Check the link or choose another day." />
    );
  }

  const confirm = () => {
    if (!slot) return;
    const appointment = addAppointment({
      scheduleId: schedule.id,
      slot,
      serviceId: service.id,
      total: service.price,
    });
    // replace: the booking modal is replaced by the visit – "back" does not return to booking
    navigation.replace("AppointmentDetails", { appointmentId: appointment.id });
  };

  return (
    <View style={styles.flex}>
      <ScrollView contentContainerStyle={styles.content}>
        <AppText variant="title">{vet.name}</AppText>
        <AppText muted>
          {formatDayLabel(schedule.date)} · {clinic.name}
        </AppText>
        <View style={styles.services}>
          {SERVICES.map((s) => {
            const active = s.id === serviceId;
            return (
              <Pressable
                key={s.id}
                onPress={() => setServiceId(s.id)}
                style={[
                  styles.service,
                  {
                    backgroundColor: active ? theme.colors.primary : theme.colors.surface,
                    borderColor: active ? theme.colors.primary : theme.colors.border,
                  },
```

```
                ]}
                accessibilityRole="radio"
                accessibilityLabel={`${s.name}, ${formatPrice(s.price)}`}
                accessibilityState={{ selected: active }}
              >
                <AppText
                  variant="caption"
                  color={active ? theme.colors.onPrimary : theme.colors.text}
                >
                  {s.name} · {formatPrice(s.price)}
                </AppText>
              </Pressable>
            );
          })}
        </View>
        <SlotGrid
          slots={schedule.slots}
          taken={schedule.takenSlots}
          selected={slot}
          onSelect={setSlot}
        />
      </ScrollView>
      <View
        style={[
          styles.summary,
          {
            backgroundColor: theme.colors.surface,
            borderTopColor: theme.colors.border,
            paddingBottom: insets.bottom + spacing.md,
          },
        ]}
      >
        <View style={styles.flex}>
          <AppText variant="caption" muted>
            {slot ? `${service.name} at ${formatTime(slot)}` : "Choose a time"}
          </AppText>
          <AppText variant="subtitle">{formatPrice(service.price)}</AppText>
        </View>
        <Pressable
          disabled={slot === null}
          onPress={confirm}
          style={[
            styles.button,
            { backgroundColor: theme.colors.primary },
            slot === null && styles.dim,
          ]}
        >
          <AppText variant="subtitle" color={theme.colors.onPrimary}>
            Book
          </AppText>
        </Pressable>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { padding: spacing.lg, gap: spacing.md },
  services: { flexDirection: "row", flexWrap: "wrap", gap: spacing.sm },
  service: {
    paddingHorizontal: spacing.md,
    paddingVertical: spacing.sm,
    borderRadius: radius.pill,
    borderWidth: 1,
  },
  summary: {
```

```
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    paddingHorizontal: spacing.lg,
    paddingTop: spacing.md,
    borderTopWidth: StyleSheet.hairlineWidth,
  },
  button: { paddingHorizontal: spacing.xl, paddingVertical: spacing.md, borderRadius: radius.md
},
  dim: { opacity: 0.5 },
});
```

src/screens/NotFoundScreen.tsx

New file – unknown deep link

```
// src/screens/NotFoundScreen.tsx – unknown deep link
import { Pressable, StyleSheet, View } from "react-native";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import type { RootStackScreenProps } from "@/navigation/types";
import { spacing, useAppTheme } from "@/theme/theme";

export function NotFoundScreen({ navigation }: RootStackScreenProps<"NotFound">) {
  const theme = useAppTheme();
  const { user } = useAuth();
  // The available screens depend on the auth state (guard in RootNavigator)
  const goHome = () => (user ? navigation.navigate("Main") : navigation.navigate("Welcome"));
  return (
    <View style={styles.container}>
      <EmptyState
        icon="help-circle-outline"
        title="Page not found"
        message="The link is invalid."
      />
      <Pressable onPress={() => (navigation.canGoBack() ? navigation.goBack() : goHome())}>
        <AppText color={theme.colors.primary}>Go back</AppText>
      </Pressable>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, alignItems: "center", justifyContent: "center", gap: spacing.lg },
});
```

12. Deep links

The `linking` config mirrors how navigators are nested. URL params are always strings – `parse` converts them and `stringify` converts back. The `*` path catches unknown addresses.

src/navigation/linking.ts

New file – deep links: URL -> screen (with parameters)

```
// src/navigation/linking.ts – deep links: URL -> screen (with parameters)
import * as Linking from "expo-linking";
import type { LinkingOptions } from "@react-navigation/native";
import type { RootStackParamList } from "./types";

export const linking: LinkingOptions<RootStackParamList> = {
  // createURL("/") returns exp://IP:8081/--/ in Expo Go and pawcare:// in a build
  prefixes: [Linking.createURL("/"), "pawcare://", "https://pawcare.example.com"],
  config: {
```

```
    // The structure mirrors how the navigators are nested
    screens: {
      Welcome: "welcome",
      SignIn: "login",
      Main: {
        screens: {
          HomeTabs: {
            screens: {
              VetsTab: {
                screens: {
                  Vets: "vets",
                  VetDetails: "vet/:vetId",
                  Filters: "filters",
                },
              },
              ScheduleTab: {
                screens: {
                  Today: "schedule/today",
                  Tomorrow: "schedule/tomorrow",
                  DayAfter: "schedule/day-after",
                },
              },
              AppointmentsTab: "appointments",
              ProfileTab: "profile",
            },
          },
          Clinics: "clinics",
          Settings: "settings",
          About: "about",
        },
      },
      BookAppointment: "book/:scheduleId",
      AppointmentDetails: "appointment/:appointmentId",
      VetVideo: "video/:vetId",
      NotFound: "*", // any unknown address
    },
  },
};
```

13. Root Stack with route guards

The key file of this stage. When `user === null` only the welcome and sign-in screens exist – the app cannot be entered even with a deep link. After signing in, React Navigation switches the screen set by itself. `Stack.Group` with `presentation: "modal"` gives several screens shared options. `onStateChange` logs screen changes (replaced by analytics in lecture 9).

src/navigation/RootNavigator.tsx

New file – navigation container + root Stack with route guards

```
// src/navigation/RootNavigator.tsx – navigation container + root Stack with route guards
import { useRef } from "react";
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import { ActivityIndicator } from "react-native";
import { useAuth } from "@/auth/AuthContext";
import { useAppTheme } from "@/theme/theme";
import { linking } from "./linking";
import { navigationRef } from "./navigationRef";
import { toNavigationTheme } from "./navTheme";
import { MainDrawer } from "./MainDrawer";
import type { RootStackParamList } from "./types";
import { WelcomeScreen } from "@/screens/auth/WelcomeScreen";
import { SignInScreen } from "@/screens/auth/SignInScreen";
import { BookAppointmentScreen } from "@/screens/booking/BookAppointmentScreen";
import { AppointmentDetailsScreen } from "@/screens/appointments/AppointmentDetailsScreen";
```

```
import { VetVideoScreen } from "@/screens/vets/VetVideoScreen";
import { NotFoundScreen } from "@/screens/NotFoundScreen";

const Stack = createNativeStackNavigator<RootStackParamList>();

export function RootNavigator() {
  const theme = useAppTheme();
  const { user } = useAuth();
  const routeNameRef = useRef<string | undefined>(undefined);

  return (
    <NavigationContainer
      ref={navigationRef}
      theme={toNavigationTheme(theme)}
      linking={linking}
      fallback={<ActivityIndicator style={{ flex: 1 }} />} // while a deep link is resolved
      onReady={() => {
        routeNameRef.current = navigationRef.getCurrentRoute()?.name;
      }}
      // Screen tracking – the base of analytics (extended in L09)
      onStateChange={() => {
        const current = navigationRef.getCurrentRoute()?.name;
        if (current && current !== routeNameRef.current) {
          console.log(`[navigation] ${routeNameRef.current} -> ${current}`);
        }
        routeNameRef.current = current;
      }}
    >
      <Stack.Navigator
        screenOptions={{
          headerTintColor: theme.colors.primary,
          headerBackButtonDisplayMode: "minimal", // iOS: arrow only, no text
          contentStyle: { backgroundColor: theme.colors.background },
        }}
      >
        {user === null ? (
          // Signed out: only the sign-in screens exist (a guard via conditional rendering)
          <Stack.Group screenOptions={{ headerShown: false, animation: "fade" }}>
            <Stack.Screen name="Welcome" component={WelcomeScreen} />
            <Stack.Screen
              name="SignIn"
              component={SignInScreen}
              options={{ headerShown: true, title: "Sign in", animation: "slide_from_right" }}
            />
          </Stack.Group>
        ) : (
          <>
            <Stack.Screen name="Main" component={MainDrawer} options={{ headerShown: false }} />
            {/* Modal group: the screen slides up from the bottom */}
            <Stack.Group screenOptions={{ presentation: "modal" }}>
              <Stack.Screen
                name="BookAppointment"
                component={BookAppointmentScreen}
                options={{ title: "Book a visit", gestureEnabled: false }}
              />
            </Stack.Group>
            <Stack.Screen
              name="AppointmentDetails"
              component={AppointmentDetailsScreen}
              // A title that depends on a route parameter
              options={({ route }) => ({ title: `Visit ${route.params.appointmentId}` })}
            />
            {/* Transparent modal: the previous screen is visible underneath */}
            <Stack.Screen
              name="VetVideo"
              component={VetVideoScreen}
```

```
              options={{ presentation: "transparentModal", animation: "fade", headerShown: false
}}
            />
          </>
        )}
        <Stack.Screen name="NotFound" component={NotFoundScreen} options={{ title: "Not found"
}} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

14. App.tsx – providers

`GestureHandlerRootView` (Drawer gestures) → `SafeAreaProvider` → state providers → `RootNavigator`.

App.tsx

Replace the whole file – stage L03: providers + navigation

```
// App.tsx – stage L03: providers + navigation
import { useEffect } from "react";
import { StatusBar } from "expo-status-bar";
import * as SplashScreen from "expo-splash-screen";
import { GestureHandlerRootView } from "react-native-gesture-handler";
import { SafeAreaProvider } from "react-native-safe-area-context";
import {
  Inter_400Regular,
  Inter_500Medium,
  Inter_600SemiBold,
  Inter_700Bold,
  Inter_900Black,
  useFonts,
} from "@expo-google-fonts/inter";
import { AuthProvider } from "@/auth/AuthContext";
import { RootNavigator } from "@/navigation/RootNavigator";
import { AppointmentsProvider } from "@/state/AppointmentsContext";

SplashScreen.preventAutoHideAsync();

export default function App() {
  const [fontsLoaded, fontError] = useFonts({
    Inter_400Regular,
    Inter_500Medium,
    Inter_600SemiBold,
    Inter_700Bold,
    Inter_900Black,
  });

  useEffect(() => {
    if (fontsLoaded || fontError) SplashScreen.hideAsync();
  }, [fontsLoaded, fontError]);

  if (!fontsLoaded && !fontError) return null;

  return (
    // GestureHandlerRootView – required by the Drawer (swipe gestures)
    <GestureHandlerRootView style={{ flex: 1 }}>
      <SafeAreaProvider>
        <AuthProvider>
          <AppointmentsProvider>
            <StatusBar style="auto" />
            <RootNavigator />
          </AppointmentsProvider>
```

```
        </AuthProvider>
      </SafeAreaProvider>
    </GestureHandlerRootView>
  );
}
```

15. Delete unused files

The following files from the previous stage are no longer used – delete them:

- `src/components/VetDetailsModal.tsx`
- `src/screens/BookingView.tsx`
- `src/screens/ScheduleView.tsx`
- `src/screens/VetsView.tsx`

Terminal

```
rm src/components/VetDetailsModal.tsx src/screens/BookingView.tsx src/screens/ScheduleView.tsx
src/screens/VetsView.tsx
```

Run and check

Terminal

```
npx expo start -c
```

- ☐ as a guest: the Visits tab shows a message instead of opening,
- ☐ vets: search in the header, filters as a sheet (drag the grabber – two heights), the filters come back to the list,
- ☐ vet profile: transparent header, the video as a transparent modal,
- ☐ booking: select a slot and try to close the modal with a gesture – a question appears,
- ☐ swipe between days in the schedule; open the side menu with a swipe from the left edge,
- ☐ deep link: run the command below (Android) – the app opens the vet profile.

Deep link test (Android emulator)

```
npx uri-scheme open "exp://127.0.0.1:8081/--/vet/v1" --android
```

Pitfalls – watch out

preload + setOptions

A screen loaded with `navigation.preload()` must not set header buttons with `setOptions` in an effect (they would be set for an invisible screen and could crash). Header buttons are passed in the navigator `options` (`VetHeaderActions`).

usePreventRemove and replace/reset

The guard intercepts EVERY removal of the screen, including our own `navigation.replace` after booking. In the callback we let the `REPLACE` action through (`navigation.dispatch(data.action)`) without asking.

URL params are strings

Without `parse`/`stringify` an array param ends up in the URL as `[object Object]`. Pass simple, serializable values as params (no functions, no class instances).

Nested headers

When a tab contains a Stack, hide the tab's header (`headerShown: false`) – otherwise you get two headers on top of each other.

formSheet and transparentModal on the web

The sheet with detents is native on iOS and Android; in the browser it is a regular modal. Always check navigation on both mobile platforms.

Checklist

- ☐ `npx tsc --noEmit` reports no errors (every `navigate` is typed),
- ☐ app screens cannot be opened without signing in (also via a deep link),
- ☐ Android Back behaves sensibly in every navigator,
- ☐ an unknown address shows the "Not found" screen.

Extra tasks

1. Open another vet from the profile ("Similar vets") with `navigation.push` – two profiles on the stack thanks to `getId`.
2. Use `drawerType: "permanent"` on wide screens (tablet) – use `useWindowDimensions`.
3. Add a deep link to the filters screen with preselected specialties (`pawcare://filters?specialties=Surgery,Dentistry`).