# **Frontend Technical Specification [FRONT-SPEC-001]**

## **1. Project Architecture Overview**

### **1.1 Project Structure**

Copy

hiplando-frontend/

├── public/

├── src/

│ ├── components/

│ │ ├── common/ # Reusable UI components

│ │ ├── layout/ # Layout components

│ │ └── features/ # Feature-specific components

│ ├── hooks/ # Custom React hooks

│ ├── services/ # API and external services

│ │ ├── api/

│ │ └── auth/

│ ├── utils/ # Helper functions

│ ├── pages/ # Page components

│ ├── types/ # TypeScript definitions

│ ├── contexts/ # React contexts

│ ├── constants/ # Constants and enums

│ └── assets/ # Static assets

├── package.json

└── tsconfig.json

### **1.2 Core Technical Stack**

json

Copy

{

"dependencies": {

"react": "^18.2.0",

"react-dom": "^18.2.0",

"react-router-dom": "^6.x",

"@tanstack/react-query": "^5.x",

"axios": "^1.x",

"tailwindcss": "^3.x",

"date-fns": "^2.x",

"formik": "^2.x",

"yup": "^1.x",

"zustand": "^4.x"

},

"devDependencies": {

"typescript": "^5.x",

"@types/react": "^18.x",

"@types/react-dom": "^18.x",

"vite": "^5.x",

"@vitejs/plugin-react": "^4.x"

}

}

### **1.3 Environment Configuration**

typescript

Copy

*// src/config/environment.ts*

export const environment = {

apiUrl: process.env.REACT\_APP\_API\_URL || 'https://api.hiplando.com',

imageUrl: process.env.REACT\_APP\_IMAGE\_URL,

stripePublicKey: process.env.REACT\_APP\_STRIPE\_PUBLIC\_KEY,

environment: process.env.NODE\_ENV,

isProduction: process.env.NODE\_ENV === 'production'

};

### **1.4 API Client Setup**

typescript

Copy

*// src/services/api/client.ts*

import axios from 'axios';

import { environment } from '@/config/environment';

export const apiClient = axios.create({

baseURL: environment.apiUrl,

timeout: 10000,

headers: {

'Content-Type': 'application/json'

}

});

apiClient.interceptors.request.use((config) => {

const token = localStorage.getItem('auth\_token');

if (token) {

config.headers.Authorization = `Bearer ${token}`;

}

return config;

});

apiClient.interceptors.response.use(

(response) => response,

async (error) => {

if (error.response?.status === 401) {

*// Handle token refresh or logout*

}

return Promise.reject(error);

}

);

### **1.5 Core Types**

typescript

Copy

*// src/types/common.ts*

export interface PaginationParams {

page: number;

pageSize: number;

}

export interface PaginatedResponse<T> {

data: T[];

total: number;

page: number;

pageSize: number;

totalPages: number;

}

export interface ApiError {

message: string;

code: string;

details?: Record<string, any>;

}

*// src/types/auth.ts*

export interface User {

id: string;

email: string;

firstName?: string;

lastName?: string;

role: UserRole;

profileImageUrl?: string;

}

export enum UserRole {

ADMIN = 'ADMIN',

USER = 'USER',

STABLE\_OWNER = 'STABLE\_OWNER'

}

export interface AuthState {

user: User | null;

token: string | null;

isAuthenticated: boolean;

}

### **1.6 Route Configuration**

typescript

Copy

*// src/routes/index.tsx*

import { createBrowserRouter } from 'react-router-dom';

export const router = createBrowserRouter([

{

path: '/',

element: <MainLayout />,

children: [

{ path: '/', element: <HomePage /> },

{ path: '/horses', element: <HorseListPage /> },

{ path: '/horses/:id', element: <HorseDetailPage /> },

{

path: '/profile',

element: <PrivateRoute><ProfilePage /></PrivateRoute>

},

{

path: '/horses/new',

element: <PrivateRoute><HorseCreatePage /></PrivateRoute>

}

]

},

{

path: '/auth',

element: <AuthLayout />,

children: [

{ path: 'login', element: <LoginPage /> },

{ path: 'register', element: <RegisterPage /> },

{ path: 'forgot-password', element: <ForgotPasswordPage /> }

]

}

]);

## **2. Common Components Library**

### **2.1 Button Component**

typescript

Copy

*// src/components/common/Button/index.tsx*

import React from 'react';

type ButtonVariant = 'primary' | 'secondary' | 'outline' | 'ghost';

type ButtonSize = 'sm' | 'md' | 'lg';

interface ButtonProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {

variant?: ButtonVariant;

size?: ButtonSize;

isLoading?: boolean;

leftIcon?: React.ReactNode;

rightIcon?: React.ReactNode;

}

const variantStyles: Record<ButtonVariant, string> = {

primary: 'bg-primary-dark hover:bg-primary text-white',

secondary: 'bg-neutral-200 hover:bg-neutral-300 text-neutral-800',

outline: 'border-2 border-primary-dark text-primary-dark hover:bg-primary-dark hover:text-white',

ghost: 'text-primary-dark hover:bg-neutral-100'

};

const sizeStyles: Record<ButtonSize, string> = {

sm: 'px-3 py-1.5 text-sm',

md: 'px-4 py-2',

lg: 'px-6 py-3 text-lg'

};

export const Button: React.FC<ButtonProps> = ({

variant = 'primary',

size = 'md',

isLoading,

leftIcon,

rightIcon,

children,

className,

disabled,

...props

}) => {

return (

<button

className={`

inline-flex items-center justify-center

rounded-lg font-medium

transition-colors duration-200

${variantStyles[variant]}

${sizeStyles[size]}

${disabled || isLoading ? 'opacity-50 cursor-not-allowed' : ''}

${className || ''}

`}

disabled={disabled || isLoading}

{...props}

>

{isLoading && <Spinner className="mr-2" />}

{!isLoading && leftIcon && <span className="mr-2">{leftIcon}</span>}

{children}

{!isLoading && rightIcon && <span className="ml-2">{rightIcon}</span>}

</button>

);

};

### **2.2 Input Component**

typescript

Copy

*// src/components/common/Input/index.tsx*

import React from 'react';

interface InputProps extends React.InputHTMLAttributes<HTMLInputElement> {

label?: string;

error?: string;

helperText?: string;

}

export const Input: React.FC<InputProps> = ({

label,

error,

helperText,

className,

...props

}) => {

return (

<div className="flex flex-col gap-1">

{label && (

<label className="text-sm font-medium text-neutral-700">

{label}

</label>

)}

<input

className={`

px-4 py-2 rounded-lg border

focus:outline-none focus:ring-2 focus:ring-primary

${error ? 'border-error' : 'border-neutral-300'}

${className || ''}

`}

{...props}

/>

{(error || helperText) && (

<span className={`text-sm ${error ? 'text-error' : 'text-neutral-500'}`}>

{error || helperText}

</span>

)}

</div>

);

};

## **3. Feature Components**

### **3.1 Horse Profile Card**

typescript

Copy

*// src/components/features/horses/HorseProfileCard/index.tsx*

import React from 'react';

import { Link } from 'react-router-dom';

import { Horse } from '@/types/horse';

interface HorseProfileCardProps {

horse: Horse;

onFavorite?: (id: string) => void;

}

export const HorseProfileCard: React.FC<HorseProfileCardProps> = ({

horse,

onFavorite

}) => {

return (

<div className="bg-white rounded-lg shadow-md overflow-hidden">

<div className="aspect-w-4 aspect-h-3">

<img

src={horse.mainImage || '/placeholder-horse.jpg'}

alt={horse.name}

className="object-cover w-full h-full"

/>

</div>

<div className="p-4">

<div className="flex justify-between items-start">

<h3 className="text-lg font-semibold">{horse.name}</h3>

<button

onClick={() => onFavorite?.(horse.id)}

className="text-neutral-500 hover:text-primary"

>

<HeartIcon className={horse.isFavorited ? 'text-primary' : ''} />

</button>

</div>

<div className="mt-2 text-sm text-neutral-600">

<p>{horse.breed}</p>

<p>{horse.age} years old</p>

</div>

<div className="mt-3 flex justify-between items-center">

{horse.price && (

<span className="font-medium text-primary-dark">

${horse.price.toLocaleString()}

</span>

)}

<Link

to={`/horses/${horse.id}`}

className="text-sm text-primary-dark hover:underline"

>

View Details

</Link>

</div>

</div>

</div>

);

};

### **3.2 Horse List with Filters**

typescript

Copy

*// src/components/features/horses/HorseList/index.tsx*

import React from 'react';

import { useInfiniteQuery } from '@tanstack/react-query';

import { useInView } from 'react-intersection-observer';

import { HorseProfileCard } from '../HorseProfileCard';

import { fetchHorses } from '@/services/api/horses';

interface HorseListProps {

filters: HorseFilters;

}

export const HorseList: React.FC<HorseListProps> = ({ filters }) => {

const { ref, inView } = useInView();

const {

data,

fetchNextPage,

hasNextPage,

isLoading,

isFetchingNextPage

} = useInfiniteQuery({

queryKey: ['horses', filters],

queryFn: ({ pageParam = 1 }) => fetchHorses({ ...filters, page: pageParam }),

getNextPageParam: (lastPage) => lastPage.hasNextPage ? lastPage.nextPage : undefined,

});

React.useEffect(() => {

if (inView && hasNextPage) {

fetchNextPage();

}

}, [inView, fetchNextPage, hasNextPage]);

if (isLoading) return <LoadingSpinner />;

return (

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

{data?.pages.map((page) =>

page.horses.map((horse) => (

<HorseProfileCard key={horse.id} horse={horse} />

))

)}

<div ref={ref}>

{isFetchingNextPage && <LoadingSpinner />}

</div>

</div>

);

};

## **4. Authentication Implementation**

### **4.1 Auth Store**

typescript

Copy

*// src/stores/authStore.ts*

import create from 'zustand';

import { AuthState, User } from '@/types/auth';

interface AuthStore extends AuthState {

login: (user: User, token: string) => void;

logout: () => void;

updateUser: (user: Partial<User>) => void;

}

export const useAuthStore = create<AuthStore>((set) => ({

user: null,

token: null,

isAuthenticated: false,

login: (user, token) => set({

user,

token,

isAuthenticated: true

}),

logout: () => {

localStorage.removeItem('auth\_token');

set({

user: null,

token: null,

isAuthenticated: false

});

},

updateUser: (userData) => set((state) => ({

user: state.user ? { ...state.user, ...userData } : null

}))

}));

### **4.2 Protected Route Component**

typescript

Copy

*// src/components/auth/ProtectedRoute.tsx*

import React from 'react';

import { Navigate, useLocation } from 'react-router-dom';

import { useAuthStore } from '@/stores/authStore';

export const ProtectedRoute: React.FC<{ children: React.ReactNode }> = ({ children }) => {

const { isAuthenticated } = useAuthStore();

const location = useLocation();

if (!isAuthenticated) {

return <Navigate to="/auth/login" state={{ from: location }} replace />;

}

return <>{children}</>;

};

## **5. API Integration Services**

### **5.1 Horse Service**

typescript

Copy

*// src/services/api/horses.ts*

import { apiClient } from './client';

import { PaginatedResponse, PaginationParams } from '@/types/common';

import { Horse, HorseFilters } from '@/types/horse';

export const horseService = {

fetchHorses: async (params: PaginationParams & HorseFilters): Promise<PaginatedResponse<Horse>> => {

const { data } = await apiClient.get('/horses', { params });

return data;

},

getHorse: async (id: string): Promise<Horse> => {

const { data } = await apiClient.get(`/horses/${id}`);

return data;

},

createHorse: async (horseData: Partial<Horse>): Promise<Horse> => {

const { data } = await apiClient.post('/horses', horseData);

return data;

},

updateHorse: async (id: string, horseData: Partial<Horse>): Promise<Horse> => {

const { data } = await apiClient.put(`/horses/${id}`, horseData);

return data;

},

deleteHorse: async (id: string): Promise<void> => {

await apiClient.delete(`/horses/${id}`);

}

};

## **6. Authentication Implementation**

### **6.1 Login Page Component**

typescript

Copy

*// src/pages/auth/LoginPage.tsx*

import React from 'react';

import { useNavigate, useLocation } from 'react-router-dom';

import { Formik, Form } from 'formik';

import \* as Yup from 'yup';

import { Input, Button } from '@/components/common';

import { useAuthStore } from '@/stores/authStore';

import { authService } from '@/services/api/auth';

const loginSchema = Yup.object({

email: Yup.string()

.email('Invalid email address')

.required('Required'),

password: Yup.string()

.min(8, 'Password must be at least 8 characters')

.required('Required'),

});

export const LoginPage: React.FC = () => {

const navigate = useNavigate();

const location = useLocation();

const { login } = useAuthStore();

const handleLogin = async (values: { email: string; password: string }) => {

try {

const { user, token } = await authService.login(values);

login(user, token);

const from = location.state?.from?.pathname || '/';

navigate(from, { replace: true });

} catch (error) {

*// Handle error*

}

};

return (

<div className="min-h-screen flex items-center justify-center bg-neutral-50">

<div className="max-w-md w-full p-8 bg-white rounded-lg shadow-md">

<h1 className="text-2xl font-semibold text-center mb-6">Welcome Back</h1>

<Formik

initialValues={{ email: '', password: '' }}

validationSchema={loginSchema}

onSubmit={handleLogin}

>

{({ isSubmitting }) => (

<Form className="space-y-4">

<Input

name="email"

type="email"

label="Email Address"

placeholder="Enter your email"

/>

<Input

name="password"

type="password"

label="Password"

placeholder="Enter your password"

/>

<Button

type="submit"

className="w-full"

isLoading={isSubmitting}

>

Log In

</Button>

</Form>

)}

</Formik>

</div>

</div>

);

};

### **6.2 User Profile Management**

typescript

Copy

*// src/pages/profile/ProfilePage.tsx*

import React from 'react';

import { Formik, Form } from 'formik';

import \* as Yup from 'yup';

import { Input, Button } from '@/components/common';

import { useAuthStore } from '@/stores/authStore';

import { userService } from '@/services/api/user';

const profileSchema = Yup.object({

firstName: Yup.string().required('Required'),

lastName: Yup.string().required('Required'),

phone: Yup.string(),

bio: Yup.string().max(500, 'Bio must be less than 500 characters'),

});

export const ProfilePage: React.FC = () => {

const { user, updateUser } = useAuthStore();

const handleUpdateProfile = async (values: typeof user) => {

try {

const updatedUser = await userService.updateProfile(values);

updateUser(updatedUser);

} catch (error) {

*// Handle error*

}

};

return (

<div className="max-w-2xl mx-auto p-6">

<h1 className="text-2xl font-semibold mb-6">Profile Settings</h1>

<Formik

initialValues={user || {}}

validationSchema={profileSchema}

onSubmit={handleUpdateProfile}

>

{({ isSubmitting }) => (

<Form className="space-y-6">

<div className="grid grid-cols-2 gap-4">

<Input name="firstName" label="First Name" />

<Input name="firstName" label="Last Name" />

</div>

<Input name="phone" label="Phone Number" />

<Input

name="bio"

label="Bio"

component="textarea"

rows={4}

/>

<Button type="submit" isLoading={isSubmitting}>

Save Changes

</Button>

</Form>

)}

</Formik>

</div>

);

};

## **7. Horse Management Implementation**

### **7.1 Horse Creation Form**

typescript

Copy

*// src/pages/horses/CreateHorsePage.tsx*

import React from 'react';

import { useNavigate } from 'react-router-dom';

import { Formik, Form } from 'formik';

import \* as Yup from 'yup';

import { Input, Select, Button } from '@/components/common';

import { MediaUpload } from '@/components/features/horses/MediaUpload';

import { horseService } from '@/services/api/horses';

const horseSchema = Yup.object({

name: Yup.string().required('Required'),

breed: Yup.string().required('Required'),

age: Yup.number().required('Required').positive('Must be a positive number'),

gender: Yup.string().required('Required'),

height: Yup.number().positive('Must be a positive number'),

discipline: Yup.array().of(Yup.string()),

price: Yup.number().positive('Must be a positive number'),

description: Yup.string().max(2000, 'Description too long'),

media: Yup.array().min(1, 'At least one image required'),

});

export const CreateHorsePage: React.FC = () => {

const navigate = useNavigate();

const handleCreateHorse = async (values: any) => {

try {

const horse = await horseService.createHorse(values);

navigate(`/horses/${horse.id}`);

} catch (error) {

*// Handle error*

}

};

return (

<div className="max-w-3xl mx-auto p-6">

<h1 className="text-2xl font-semibold mb-6">Add New Horse</h1>

<Formik

initialValues={{

name: '',

breed: '',

age: '',

gender: '',

height: '',

discipline: [],

price: '',

description: '',

media: [],

}}

validationSchema={horseSchema}

onSubmit={handleCreateHorse}

>

{({ isSubmitting }) => (

<Form className="space-y-6">

<div className="grid grid-cols-2 gap-4">

<Input name="name" label="Horse Name" />

<Input name="breed" label="Breed" />

<Input name="age" type="number" label="Age" />

<Select

name="gender"

label="Gender"

options={[

{ value: 'mare', label: 'Mare' },

{ value: 'stallion', label: 'Stallion' },

{ value: 'gelding', label: 'Gelding' },

]}

/>

</div>

<Input name="height" type="number" label="Height (hands)" />

<Input name="price" type="number" label="Price" />

<Select

name="discipline"

label="Disciplines"

multiple

options={[

{ value: 'dressage', label: 'Dressage' },

{ value: 'jumping', label: 'Jumping' },

{ value: 'eventing', label: 'Eventing' },

*// Add more disciplines*

]}

/>

<Input

name="description"

label="Description"

component="textarea"

rows={6}

/>

<MediaUpload

name="media"

label="Horse Photos"

maxFiles={20}

/>

<div className="flex justify-end gap-4">

<Button

variant="ghost"

onClick={() => navigate(-1)}

>

Cancel

</Button>

<Button type="submit" isLoading={isSubmitting}>

Create Profile

</Button>

</div>

</Form>

)}

</Formik>

</div>

);

};

### **7.2 Horse Detail Page**

typescript

Copy

*// src/pages/horses/HorseDetailPage.tsx*

import React from 'react';

import { useParams } from 'react-router-dom';

import { useQuery } from '@tanstack/react-query';

import { horseService } from '@/services/api/horses';

import { Gallery } from '@/components/features/horses/Gallery';

import { HorseSpecs } from '@/components/features/horses/HorseSpecs';

import { ContactOwner } from '@/components/features/horses/ContactOwner';

export const HorseDetailPage: React.FC = () => {

const { id } = useParams<{ id: string }>();

const { data: horse, isLoading } = useQuery({

queryKey: ['horse', id],

queryFn: () => horseService.getHorse(id!),

});

if (isLoading) return <LoadingSpinner />;

if (!horse) return <NotFound />;

return (

<div className="max-w-7xl mx-auto p-6">

<div className="grid grid-cols-1 lg:grid-cols-3 gap-8">

<div className="lg:col-span-2">

<Gallery images={horse.media} />

<div className="mt-8">

<h1 className="text-3xl font-semibold">{horse.name}</h1>

<div className="mt-4 flex items-center gap-4">

<Badge>{horse.breed}</Badge>

<Badge>{horse.age} years</Badge>

<Badge>{horse.gender}</Badge>

</div>

<div className="mt-6">

<p className="text-lg text-neutral-700">{horse.description}</p>

</div>

<HorseSpecs

className="mt-8"

height={horse.height}

disciplines={horse.discipline}

achievements={horse.achievements}

training={horse.training}

/>

</div>

</div>

<div>

<div className="sticky top-6">

<div className="bg-white rounded-lg shadow-md p-6">

<div className="text-2xl font-semibold">

${horse.price?.toLocaleString()}

</div>

<ContactOwner

className="mt-6"

horseId={horse.id}

ownerId={horse.owner.id}

/>

</div>

</div>

</div>

</div>

</div>

);

};