**Front-End Development Plan for Saint**

Table of Contents

**Type chapter title (level 1)1**

Type chapter title (level 2)2

Type chapter title (level 3)3

**Type chapter title (level 1)4**

Type chapter title (level 2)5

Type chapter title (level 3)6

# 1. Technologies to Use

* **HTML5**: For structuring the web pages.
* **CSS3**: For styling and layout (consider using a framework like **Tailwind CSS** or **Bootstrap** for faster development).
* **JavaScript**: For interactivity and dynamic content.
* **React.js**: A powerful JavaScript library for building reusable components and managing state.
* **Chart.js** or **D3.js**: For displaying tracking data (e.g., maps, graphs, or live footage).
* **Google Maps API**: For integrating maps and location tracking.
* **Axios**: For making API requests to the backend.

# 2. Key Pages and Components

Here’s a breakdown of the front-end structure:

## a. Landing Page

* A clean, professional landing page with:
  + Project name: **Saint**
  + Brief description of the project.
  + Login/Signup buttons.
  + Navigation menu (e.g., Home, Features, About, Contact).

## b. Dashboard

* The main interface where users can:
  + Track devices (IMEI or IP-based).
  + View live footage from cameras.
  + Access satellite imagery.
  + Perform WiFi penetration tasks.

### Include:

* + A sidebar for navigation.
  + Cards or widgets for each feature.
  + Interactive maps for tracking.

## c. Device Tracking Page

* A page to input IMEI or IP addresses and display:
  + Location on a map (using Google Maps API).
  + Device details (e.g., manufacturer, model, last seen).

## d. Live Footage Page

* A page to display live footage from connected cameras.
* Include:
  + A video player.
  + Controls for pausing, playing, and recording.

## e. Satellite Imagery Page

* A page to display real-time satellite footage.
* Include:
  + A map with satellite overlay.
  + Controls for zooming and panning.

## f. WiFi Tools Page

* A page for WiFi penetration and fake WiFi creation.
* Include:
  + Input fields for target WiFi details.
  + Buttons to initiate penetration or create a fake WiFi network.
  + Display area for captured data.

# 3. Code Implementation

## a. Setting Up the Project

1. **Install Node.js and npm**: If you haven’t already, download and install Node.js from [nodejs.org](https://nodejs.org/).
2. **Create a React App**:

bash

Copy

npx create-react-app saint-frontend

cd saint-frontend

npm start

1. **Install Dependencies**:

bash

Copy

npm install axios react-router-dom chart.js react-google-maps-api tailwindcss

## b. Folder Structure

Copy

saint-frontend/

├── public/

├── src/

│ ├── components/ # Reusable components

│ ├── pages/ # Page components

│ ├── App.js # Main app component

│ ├── index.js # Entry point

│ ├── styles/ # CSS or Tailwind styles

│ └── utils/ # Utility functions (e.g., API calls)

**c. Example Code Snippets**

**1. Landing Page (pages/LandingPage.js)**

javascript

Copy

import React from 'react';

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

const LandingPage = () => {

return (

<div className="min-h-screen bg-gray-100 flex flex-col items-center justify-center">

<h1 className="text-4xl font-bold text-blue-600 mb-4">Saint</h1>

<p className="text-lg text-gray-700 mb-8">Your ultimate tracking and security tool.</p>

<div className="space-x-4">

<Link to="/login" className="bg-blue-600 text-white px-6 py-2 rounded-lg">Login</Link>

<Link to="/signup" className="bg-green-600 text-white px-6 py-2 rounded-lg">Sign Up</Link>

</div>

</div>

);

};

export default LandingPage;

## 2. Dashboard (pages/Dashboard.js)

javascript

Copy

import React from 'react';

import Sidebar from '../components/Sidebar';

import DeviceTracker from '../components/DeviceTracker';

import LiveFootage from '../components/LiveFootage';

const Dashboard = () => {

return (

<div className="flex h-screen bg-gray-100">

<Sidebar />

<div className="flex-1 p-6">

<h1 className="text-2xl font-bold mb-6">Dashboard</h1>

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

<DeviceTracker />

<LiveFootage />

</div>

</div>

</div>

);

};

export default Dashboard;

**3. Device Tracker Component (components/DeviceTracker.js)**

javascript

Copy

import React, { useState } from 'react';

import axios from 'axios';

const DeviceTracker = () => {

const [imei, setImei] = useState('');

const [location, setLocation] = useState(null);

const handleTrack = async () => {

try {

const response = await axios.post('/api/track', { imei });

setLocation(response.data.location);

} catch (error) {

console.error('Error tracking device:', error);

}

};

return (

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

<h2 className="text-xl font-semibold mb-4">Device Tracker</h2>

<input

type="text"

placeholder="Enter IMEI or IP"

value={imei}

onChange={(e) => setImei(e.target.value)}

className="w-full p-2 border rounded-lg mb-4"

/>

<button

onClick={handleTrack}

className="bg-blue-600 text-white px-4 py-2 rounded-lg"

>

Track

</button>

{location && (

<div className="mt-4">

<p>Location: {location}</p>

{/\* Integrate Google Maps here to show the location \*/}

</div>

)}

</div>

);

};

export default DeviceTracker;

**4. Live Footage Component (components/LiveFootage.js)**

javascript

Copy

import React from 'react';

const LiveFootage = () => {

return (

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

<h2 className="text-xl font-semibold mb-4">Live Footage</h2>

<div className="w-full h-64 bg-gray-200 flex items-center justify-center">

<p className="text-gray-500">Live video stream will appear here.</p>

</div>

</div>

);

};

export default LiveFootage;

**5. Google Maps Integration**

Install the @react-google-maps/api package:

bash

Copy

npm install @react-google-maps/api

Example usage:

javascript

Copy

import { GoogleMap, LoadScript, Marker } from '@react-google-maps/api';

const MapComponent = ({ location }) => {

const mapStyles = {

height: '400px',

width: '100%',

};

return (

<LoadScript googleMapsApiKey="YOUR\_GOOGLE\_MAPS\_API\_KEY">

<GoogleMap

mapContainerStyle={mapStyles}

zoom={10}

center={location}

>

<Marker position={location} />

</GoogleMap>

</LoadScript>

);

};

export default MapComponent;

**4. Styling with Tailwind CSS**

Tailwind CSS is a utility-first CSS framework that makes styling easy. To set it up:

1. Install Tailwind:

bash

Copy

npm install tailwindcss

npx tailwindcss init

1. Add Tailwind to your src/index.css:

css

Copy

@tailwind base;

@tailwind components;

@tailwind utilities;

1. Use Tailwind classes in your components (as shown in the examples above).

**5. Routing with React Router**

Install react-router-dom:

bash

Copy

npm install react-router-dom

Set up routes in App.js:

javascript

Copy

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import LandingPage from './pages/LandingPage';

import Dashboard from './pages/Dashboard';

function App() {

return (

<Router>

<Routes>

<Route path="/" element={<LandingPage />} />

<Route path="/dashboard" element={<Dashboard />} />

</Routes>

</Router>

);

}

export default App;

**6. Testing and Debugging**

* Use **React Developer Tools** for debugging.
* Test responsiveness using browser developer tools.
* Validate functionality with unit tests (e.g., using **Jest** and **React Testing Library**).