Alternate Solutions

The scope of the MediBooki project identifies MediBooki-approved service locations, services, providers, service areas, and target populations.

# MediBooki is defined as a location where providers are located:

- The patient can book any doctor through the website

- Conducting face-to-face visits with patients and documenting those encounters in the patient’s medical records.

- Patients can be diagnosed and treated themselves.

- X-rays and analyzes can be uploaded to the site, and the patient's condition can be diagnosed through x-rays

- The provision of services through or on behalf of the MediBooki website, which means that the health center's board of directors must have control and authority over the services provided at the site.

- Providing services on a regular schedule.

**Web Application**

Several different approaches are now open to the developer. As we are developing Web applications, there are the following approaches from which best can be selected by the developing team:

Native:

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

Native:

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

Native:

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

**Native (Front-end developing):**

The default way of developing on the web (Front-end) is to write native code for each device – usually Html, CSS, Bootstrap, and JavaScript for Web Application.

React Native:

The release of React Native gave us the ability to write JavaScript code (with React syntax) that

used entirely native widgets. This takes away a major limitation of older hybrid apps but can still

suffer performance issues due to reliance on the JavaScript 'bridge'

**Angular:**

The release of Angular gave us the ability to write JavaScript code (with Typescript syntax) that use completely native UI elements. This improves design architecture, promotes code reuse, faster application development, and handles dependencies.

**Rest API in Angular:**

API (Application Programming Interface) in Angular is a set of global JavaScript functions used to carry out common tasks such as comparing objects, iterating objects, and converting data.

**React:**

React is a front-end JavaScript framework that runs on top of JavaScript. It was created by Facebook to develop apps faster. React helps developers by offering features such as client-side routing and declarative bindings, state management, and data flow.

**Rest API in ReactJS:**

ReactJS component is a top-level API. It makes the code completely individual and reusable in the application and You can consume REST APIs in a React application in a variety of ways, but in this guide, we will look at two of the most popular approaches: Axios (a promise-based HTTP client) and Fetch API (a browser in-built web API).

**Native (Back-end developing):**

The default way of developing on the web (Front-end) is to write native code for each device – usually PHP or C#, Database (MySQL), JSON for Web Application.

**Laravel:**

Laravel is a web application framework with expressive, elegant syntax. this offers superior start-up times and app performance. we’ve already laid the foundation.

**ASP.NET:**

ASP.NET is an open-source, server-side web-application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic websites, applications, and services.

**MySQL:**

MySQL is a relational database management system that uses SQL and allows you to handle, store, modify and delete data and store data in an organized way and offers support for multiple storage engines along with plug-in storage, making it more flexible.

**Mobile Application**

Several different approaches are now open to the developer. As we are developing mobile applications, there are the following approaches from which best can be selected by the developing team:

**Native:**

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each

device – usually Java for Android and Swift for iOS.

**React Native:**

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The release of React Native gave us the ability to write JavaScript code (with React syntax) that

used entirely native widgets. This takes away a major limitation of older hybrid apps but can still

suffer performance issues due to reliance on the JavaScript 'bridge'.

The release of React Native gave us the ability to write JavaScript code (with React syntax) that used entirely native widgets. This takes away a major limitation of older hybrid apps but can still suffer performance issues due to reliance on the JavaScript 'bridge'.

There are seve different approaches that are now open for the developer.

As we are developing and le applications, there are follthe owing approaches from which bestthe

can be selected by devethe loping team:

There are a nuseveralferent approaches that are now open for the developer.

As we are developing and le applications, there are follthe owing approaches from which best

can be selected by developing team:

**Hybrid App:**

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The release of React Native gave us the ability to write JavaScript code (with React syntax) that

used entirely native widgets. This takes away a major limitation of older hybrid apps but can still

suffer performance issues due to reliance on the JavaScript 'bridge'.

Create hybrid apps that behave like native but use common APIs that run consistently across iOS and Android devices. The biggest advantage of developing the hybrid app is a consistent, cross-platform UI that is compatible with most devices.

**Flutter:**

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The default way of developing on mobile is to write native code for each device – usually Java

for Android and Swift for iOS.

The release of React Native gave us the ability to write JavaScript code (with React syntax) that

used entirely native widgets. This takes away a major limitation of older hybrid apps but can still

suffer performance issues due to reliance on the JavaScript 'bridge'.

Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase and transforms the app development process. Build, test, and deploy beautiful mobile, web, desktop, and embedded apps from a single codebase.