Contents

[1.0 Introduction to PWA 2.](file:///C:\Users\2119181\Downloads\Case%20Study%20-%20SB%20Java%20-%20Legato%20-%20New.docx#_Toc112772601)

[2.0 Architecture Diagram for the PWA 4.](file:///C:\Users\2119181\Downloads\Case%20Study%20-%20SB%20Java%20-%20Legato%20-%20New.docx#_Toc112772602)

[3.0 Most important feature of PWA 5.](file:///C:\Users\2119181\Downloads\Case%20Study%20-%20SB%20Java%20-%20Legato%20-%20New.docx#_Toc112772603)

[4.0 Skills to develop the project 6.](file:///C:\Users\2119181\Downloads\Case%20Study%20-%20SB%20Java%20-%20Legato%20-%20New.docx#_Toc112772607)

**What is a progressive web app (PWA) ?**

A progressive web app (PWA) is the set of mobile web application development techniques that entails building apps that feel and look like native ones. Using a web stack (JS, HTML, and CSS), progressive web apps combine a rich functionality and smooth user experience associated with native apps. Simply put, PWA is the web app with the native-app flavor: After the installation, a user clicks on its icon on a device home screen and gets straight to the website.

Every time you access a progressive web application, you set a range of processes into motion to load the front end interface depending on the server setup, and how the web-app is optimised. One process is Server-side and the other client-side.

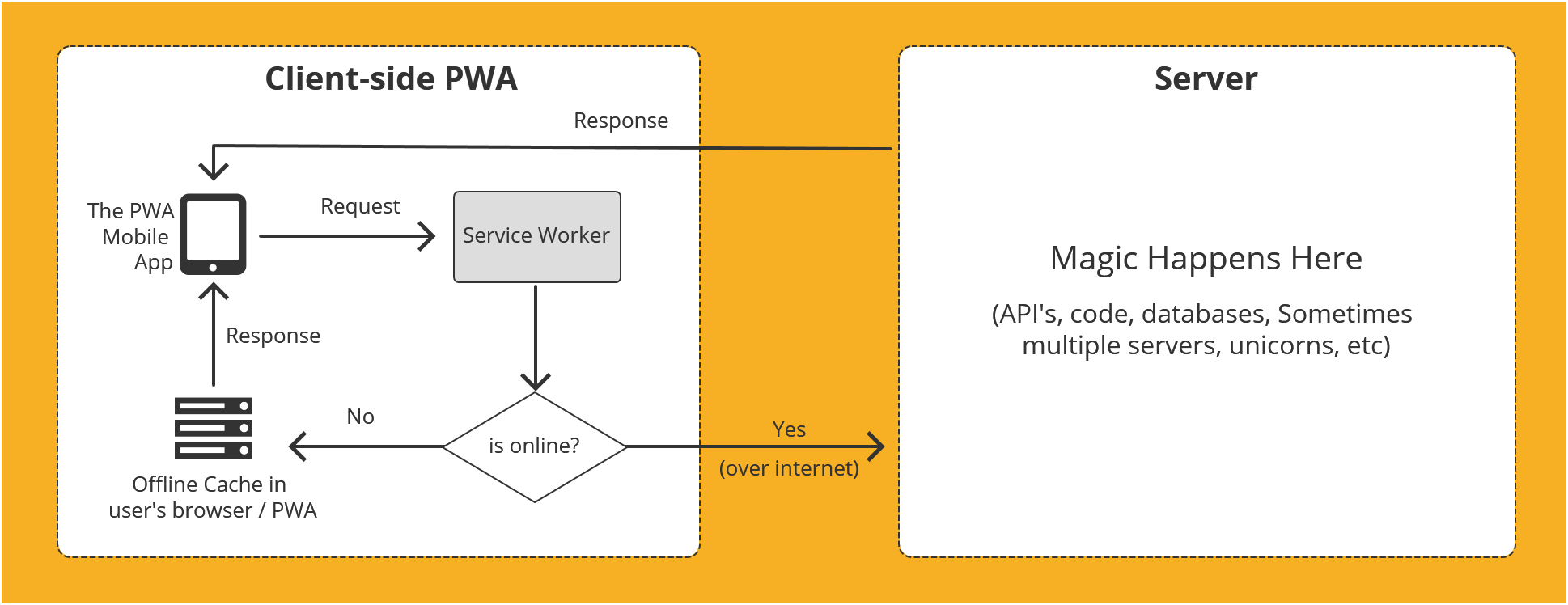
### **Server-side rendering:**

The first is server-side rendering (SSR). When you click on a PWA, the server receives a request to find the suitable HTML files and page content and renders it. This request is made for every separate page visited, so it's excellent for quick-page loads but can be a pain when navigating many different pages as the page needs to be refreshed each time.

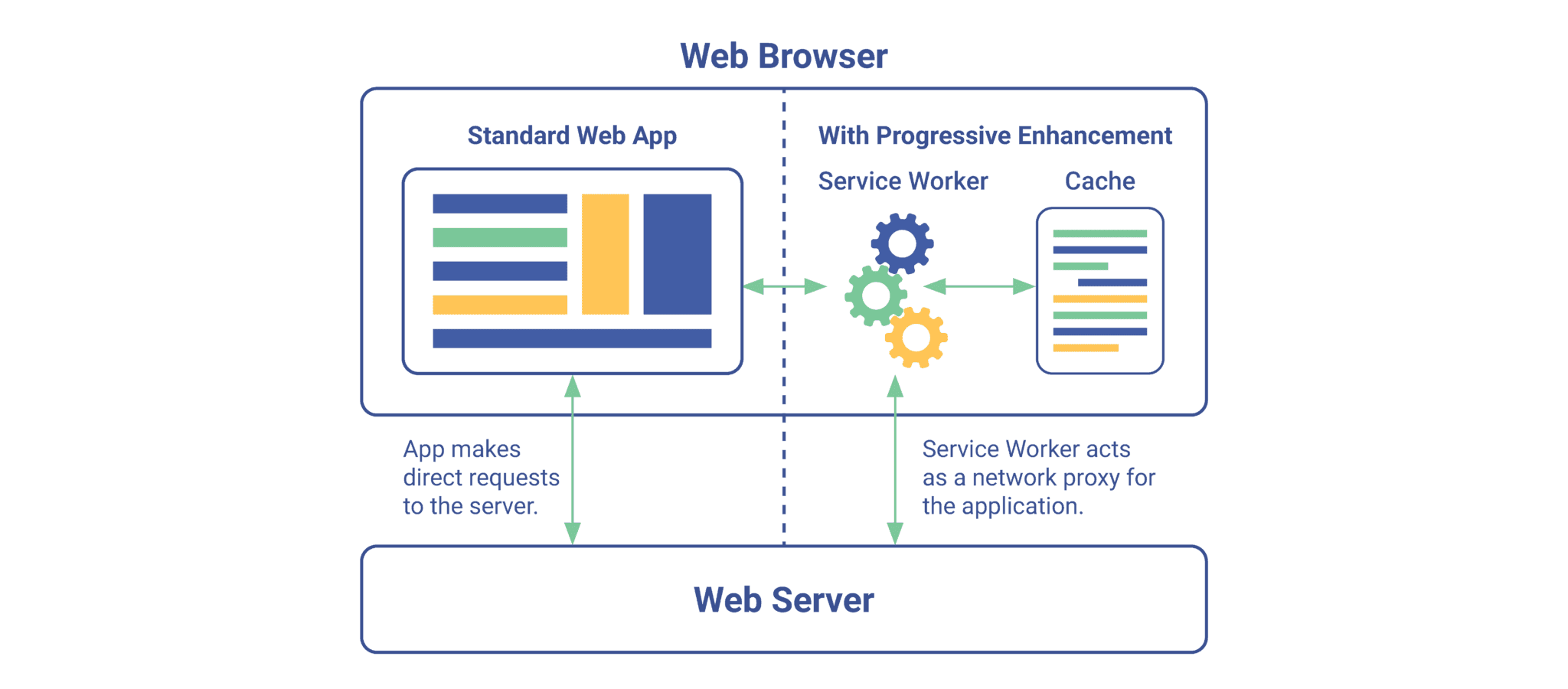
### **Client-side rendering :**

The second option is client-side rendering (CSR), which only sends one request to the server. From this, the entire skeleton of the PWA will be loaded, making for seamless navigation but a longer initial loading time.

You can design PWAs blend both client and server-side rendering techniques in-one. By doing this, the site can be rendered by the server and the contents cached, with new client-side rendering only occurring when needed. The PWA will achieve superfast loading speeds whilst keeping smooth navigation, creating an experience focused on the user. The way server or client-side rendering is optimised may also impact your search engine performance, making it easier for engines to navigate your app content and index it for users to find.



**Architecture Diagram for the PWA**



The three main components of the **PWA** architecture are:

* **The App Shell** is a set of local resources that together form the user interface and handle interactions.
* **Service Workers** are aimed at reducing data traffic. These event-driven scripts are run in the background to liaise between the app and network and to manage cache information.
* **Web App Manifest** – a JSON file containing basic information about the application, so as to describe it to the client’s browser.

# Problem statement

The purpose of the requirements document is to systematically capture requirements for the project and the system “**Patient Management Application**” to be developed. The application should follow cloud native architecture with microservices. Both functional and non-functional requirements are captured in this document. It also serves as the input for the scope of the project.

**About the System**

The client would like to develop an application which can help Patient in various ways. The portal provides facilities like login, get patient details, add patient, view patient disease.

**Scope of the System**

The scope of the system is explained through its modules as follows,

* Login – system should allow registered members to login into the portal. Use spring Basic Authentication.
* Get a patient – one should be able to search patient and details.
* The application is Progressive Web Application.
* Supports all CRUD operations on patient’s details.

|  |  |
| --- | --- |
| Compute & Integration | As an application developer, develop the application as a microservice architecture. And implement as follows:   1. Follow the Single Data Store per microservice practice 2. Document REST endpoints with OpenAPI or Swagger |
| Security & Identity | As an application developer:   1. Restrict the access over all operation (secured operations) by adding basic authentication |
| Database & Storage | As an application developer:   1. Ensure Application is implementing JPA 2. Restructure the same as per microservice architecture 3. Use can use Index DB database |
| Governance & tooling | As an application developer:   1. Containerize the complete application 2. Perform unit testing of your application and do proper CI/CD |
| Code Quality/Optimizations | 1. Optimize the DB implementation using index search 2. Use SonarQube to scan the backend application for security vulnerabilities 3. Should have written clean code that is readable 4. Should have written testable code |

# Technical Scope

Use case details

|  |  |  |
| --- | --- | --- |
| **User Story #** | **User Story Name** | **User Story** |
| US\_01 | Login | As a user, I should be able to login the system with valid credentials.  Acceptance criteria:   * User should be able login with their registered User Id or Email Id and Password * When a user tries to login with incorrect credentials, valid error message should be shown |
| US\_02 | Get Patients  (Both in online and offline mode) | As a user, I should be able to get patient in the system.  Acceptance criteria:   * User should be able to get patient * Patient can be get based on below criteria -   Patient ID (Generated once member register in the system),First Name, Last Name, Phone\_no, Address, Disease   * If the required fields are empty/invalid then appropriate validation error message must be shown * Malicious text should be validated to prevent from CSRF and XSS attacks |
| US\_03 | Add and Delete Patients  (Both in online and offline mode) | As a user, I should be able to add and delete a patient in the system.  Acceptance criteria:   * User should be able to add and delete patient * Patient can be add and delete based on below criteria -   Patient ID (Generated once member register in the system),First Name, Last Name, Phone\_no, Address, Disease   * If the required fields are empty/invalid then appropriate validation error message must be shown   Malicious text should be validated to prevent from CSRF and XSS attacks |



**Skills to develop the project**

List the Technology required to develop the project.

|  |  |
| --- | --- |
| Skill Stack | Java 8 / 11 |
| Front end | React,Vanilla JavaScript,HTML5 |
| Service End | Java |
| Database | Postgresql , IndexedDB |
| Source Control | GIT, GitHub |
| Cloud | AWS |
| Unit testing | Light house |