MTF Prescriber Portal

Technical & Operation Views

V1.1

This document provides an insight into the Technical & Operational aspects of the Application.

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Created Date | Created By | Reviewed By | Comments |
| 1.1 | 29-June-2016 | Sushmita |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

[1. Perspective](#_Toc405214216)

[2. Technical Architecture](#_Toc405214217)

[2.1 Design Pattern – React component](#_Toc405214218)

[2.2 Physical Tiers](#_Toc405214219)

[2.3 Logical Modules](#_Toc405214220)

[2.4 Data Flow](#_Toc405214221)

[2.5 Main components of the Application](#_Toc405214222)

[2.6 Listing of libraries used in the application](#_Toc405214223)

[2.10 Development Environment of the Application](#_Toc405214227)

[3. Operational Aspects of the Application](#_Toc405214228)

[3.1 The Login](#_Toc405214229)

[3.2 The Registration](#_Toc405214230)

[4. Miscellaneous Aspects of the Application](#_Toc405214250)

[4.1 Environment Settings](#_Toc405214251)

[4.1.1 Regional Settings](#_Toc405214252)

# Perspective

This document provides 2 perspectives to the ‘Application’ solution.

1. Technical
2. Operational

We will look at each in turn & how these perspectives converge to provide the whole picture.

Before we move into the heart of the Solution, let’s look at why this Solution was envisaged.

Need for the Solution -

1. The Current Scenario

[fill about the current existing application scenerio]

1. Advantages of the Developed Solution –
2. React is, in my opinion, the premier way to build big, fast Web apps with JavaScript.
3. React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.
4. Build encapsulated components that manage their own state, then compose them to make complex UIs.
5. We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.
6. JSX performs optimization while compiling the source code to JavaScript. The generated code runs faster than an equivalent code written directly in JavaScript.
7. In contrast to JavaScript, JSX is statically-typed and mostly type-safe. The quality of applications becomes higher when being developed using JSX, since many errors will be caught during the compilation process. It also offers debugging features at the compiler level as well.
8. JSX offers a solid class system much like Java, freeing the developers from working with the too-primitive prototype-based inheritance system provided by JavaScript.
9. The Operating environment is constrained since both the Reference Data & User Data are stored on a Central Server & changes to either, are visible to all.
10. Business logic is embedded in the application & cannot be changed without creating a new version of the application, that will be deployed to all users.
11. The application can be used only by Authenticated users & can only use Modules that they have been authorised to use.

# Technical Architecture

## Design Pattern – Component & JSX

The design pattern used in application is component structure and JSX.

However, they are not hard and fast rules, but more of an opinionated set of guidelines I choose to adhere to that make my code more readable, more robust and easily maintainable.

* 1. Component organisation

React.createClass({

propTypes: {},

mixins : [],

getInitialState: function() {},

getDefaultProps: function() {},

componentWillMount : function() {},

componentWillReceiveProps: function() {},

componentWillUnmount : function() {},

\_parseData : function() {},

\_onSelect : function() {},

render : function() {}

})

* 1. JSX
* Multi-line JSX

return (

<div>

<ComponentOne />

<ComponentTwo />

</div>

);

* Conditional JSX

var optionalElement;

if (this.props.condition) {

optionalElement = (<div> … </div>);

}

return (

<div>

…

{optionalElement}

…

</div>

);

Component

attribute={...}

anotherAttribute={...}

attributeThree={...}

…

/>

## Physical Tiers



## 3. Logical Modules

1. User lands on login page

* Provider clicks on Register Now Hyperlink and will be redirected to Registration page where he/she can select whether he/she is an NPI Owner or not.
* If Yes,
* The Registration Page should display the following fields on the screen.

1. NPI Number: Required and editable.

2. Email ID: Required and editable

3. Next Button.

When Provider clicks on “Next” button, system validates the NPI Number and Email ID and does the following:

a) Sends an Email to the given ID which includes a link and a pin number for activating the account.

b) Navigates to the next page where the provider should enter the Pin Number.

Upon navigating to the next page, provider should be able to:

1. Enter the Pin number

2. Select the checkbox to accept MTF Prescriber Portal as the preferred method of communication and click on “continue” button. Else, provider cannot move forward and enrol.

When Provider clicks on “continue” button, systems verifies the pin number and redirects to the next page, where

a) The provider’s information is automatically filled on to the editable Form from the Provider’s file by making an AJAX call.

b) Provider should create a Username and a Password to complete registration.

Provider’s file should be updated each time the provider makes changes in the portal.

Upon creating the username and password provider should agree the privacy policy (privacy statement of 1974, HIPPA).

• If No,

Display a disclaimer which states the benefits/reasons of joining the portal.

2.2 Automatically fill out the form

• System takes NPI Number as input and returns the corresponding provider information in the editable form which contains:

1. Provider Name: Captured. Editable.

2. NPI Information: Captured. Editable.

3. State License: Captured. Editable.

4. Dea: Optional. Editable.

5. Work Address: Captured. Editable.

6. E-mail: Captured. Editable.

7. Work Phone: Captured. Editable.

8. Fax: Optional. Editable.

9. Clinic: Optional. Editable.

10. MTF Location: Captured. Editable.

11. UI Validations

System performs following validations when the Provider clicks Submit button on the registration Page.

• If Name is empty, system displays the error message “Name is required”.

• If NPI Number is empty, system displays the error message “NPI Number is required”.

• If State License is empty, system displays the error message “License is required”.

• If Work Address is empty, system displays the error message “Work address is required”.

• If E-mail is empty, system displays the error message “Email is required and should be valid ”.

Regex for a valid Email ID: “[a-zA-Z]+@.\*\.mil$”

This regular expression checks for a valid email with “.mil” extension.

• If Work Phone is empty, system displays the error message “Work Phone is required”.

• If MTF Location is empty, system displays the error message “MTF Location is required”.

• If Username is empty, system displays the error message “Username is required”.

• If Password is empty, system displays the error message “Password is required and should match the following criteria.”

 Minimum of 15 characters.

 One upper case character.

 One lower case character.

 Special character “!@#$%^&\*()\_-+”.

## 4. Data Flow

* Node service layer created for common access of database.
* Service hosted on server and accessible to all the clients
* Service can be accessible through GET and POST protocols.
* Sample Request

Get NPI number or state License.

NPI Number: 1

* Sample Response

{"info":[

{

"id":"1",

"name": "Pete Hunt",

"npi": "1",

"license":"dl123",

"dea":"dea123",

"address":"addr1",

"email":"email1@abc.mil",

"phone":"1234",

"fax":"",

"clinic":"",

"location":"loc1"

}

}

* Response Codes

List of all HTTP Status Codes

Status Code Status Message

200 Success

400 Invalid Request

401 Unauthorized

403 Forbidden

401 Not found

* + 1. Internal Server Error

## 5. Main components of the Application

* Register.js
* RegisterFields.js

## Listing of libraries used in the application

• React

• 'joi';

• 'react-validation-mixin';

• 'joi-validation-strategy';

• fetch

• node js / node server

• jasmine and karma

• ESLint

# Operational Aspects of the Application

[screenshot of the web pages]

* The Login
* The Registration

# Development Environment of the Application

[put the tools using for development]