12-2017

Developing Node.Js Portal For IBM BPM

Sandeep Vemulapalli
St. Cloud State University, vsandeep1818@gmail.com

Creating Node JS Portal for IBM BPM

by

Sandeep Vemulapalli

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

in Information Assurance

December, 2017

Abstract

Business process management (BPM) solutions enable an enterprise to choreograph processes and the process steps across disparate applications, people, and systems. In addition to reduced cost through continued process improvement and automation, BPM provides the foundation for converged and agile business and IT responsiveness.

IBM Business Process Management is one of such product which comes with great features for front end and back end solutions. Though BPM provides out of the box portal for the end users to work on the tasks, we can create more responsive portal and highly customizable portal using Node js frameworks according to client specifications. This portal can be used on multiple devices such as desktop browsers, smart phones and tablets.

Acknowledgements

I am greatly indebted to my advisor, Dr. Dennis Guster, Ph.D., who has supported me throughout my project. I have taken efforts in this project. However, it would not have been possible without individual node JS modules that were developed by technology enthusiasts across the world. I would like to extend my sincere thanks to all of them.

I would like to express my gratitude towards my friends and parents for their kind cooperation and encouragement which help me in completion of this project.

Table of Contents

Appendices Page

List of Figures

1. Introduction

The Node JS Portal is a new express js portal that can be used as a starting point for creating and extending your own portals. It is responsive, meaning that it works on various devices (mobile, tablet, laptop, and traditional work station). The Node JS Portal can be used in a single IBM BPM environment or in a Process Federation Server environment BPM suite consists of process center (Development Environment), process server (Run time environment). These both components have REST capabilities to communicate with any external enterprise systems.

Following are the some of the advantages with the plugin that is going to be developed.

1. Responsive, and rich in Look & Feel portal for the end users and their team managers.
2. High availability and load balancing web servers lightweight and can handle a great number of simultaneous requests.
3. Since we are using MEAN framework we will be using JavaScript in all places (Client, Server, and Database). This means the developers needs to know JavaScript and HTML to use and extend this plugin.

For the development of responsive BPM portal, MEAN JS frame work is used .MEAN.JS framework is a full-stack JavaScript solution that helps you build fast, robust, and maintainable production web applications using MongoDB, Express, AngularJS, and Node.js

1.1 Current Design

The Out of the Box architecture of the IBM BPM is l like following when you install it.

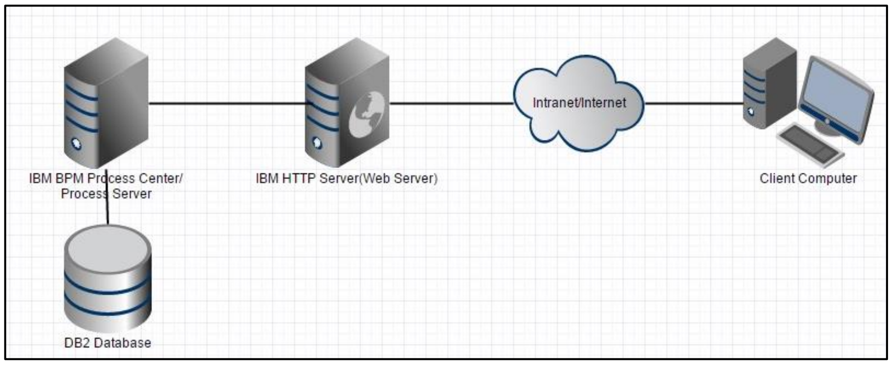


Figure 1: IBM BPM current architecture [1].

The above picture (Figure 1) shows that the end user has to use Process Portal shipped Out of Box (OOB) by IBM. This portal can be improved a lot by using MEAN framework.

1.2 Proposed Design

We will introduce MEAN frame work between App Server and client so that we will have more control over what we show to the client and UI design which can be customized according to client needs.

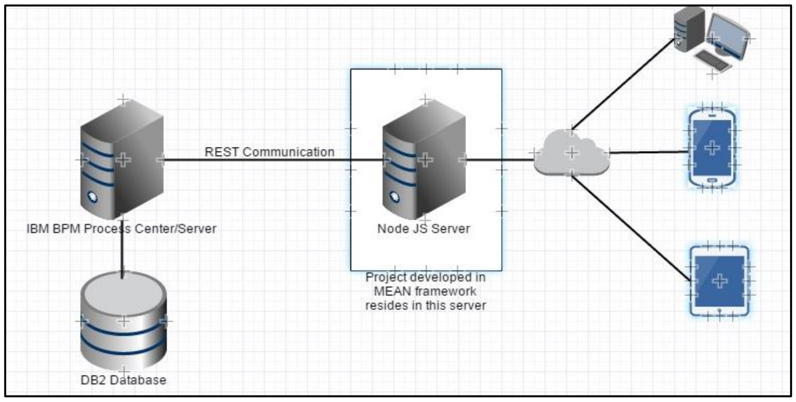


Figure 2: Integration of BPM with Node JS portal [2].

As shown in the Figure 2, the Node JS server can get the data using REST protocol from IBM BPM server. Using this architecture there is another advantage, since REST protocol works over HTTP/HTTPS there is no need for Node JS server to be on the same network where BPM server is installed. Node JS server also works as front end interface for BPM server masking application logic & sensitive intra-network parameters such as ip addresses. Now any device can access the responsive portal from the Node JS server.

1.3 Current Design vs Proposed Design

In traditional architecture, there is only webserver component whose sole purpose is to server the client requests. Usually organizations use IBM HTTP server (built on Apache) installed on Windows or Linux operation system. In the proposed design, we will be replacing this webserver component with powerful MEAN stack i.e. we will be having database along with the webserver. Following is the detailed analysis on introducing MEAN.

(a) Operating System: The first choice in any tech stack is the operating system. Both the HTTP Server and MEAN stack doesn’t have any restrictions on operating systems. We can use any of the Windows, Linux, Solaris operating systems.

(b) Web Server: In the MEAN stack, the web server is provided by Node.js while IBM HTTP Server is based on the Apache Software Foundation’s Apache HTTP Server. This can improve the performance of the application, as Node.js is entirely non-blocking and event-based, allowing for true concurrency among requests. Node.js is lightweight and relatively new, however, which ultimately means that organization using MEAN will be largely on its own when it comes to non-standard extensions. While there is active plug-in development for Node.js, the technology is not as matured as Apache. This usually means that organization need to write its own plug-ins to cover the areas where Node.js is missing functionality. Additionally, choosing Node.js locks all code on web server into JavaScript. For new development, this isn’t a major concern, but converting a back-end of significant complexity can be time-consuming.

(c) Data Store: The MEAN stack comes with MongoDB (or an equivalent non-relational database). This Data Base is completely different from the traditional SQL based databases. This is very significant change one needs to consider if the application is being migrated to the MEAN frame work. Translating the data in an existing SQL database requires a lot of forethought to eliminate redundant/ unnecessary object attributes, and will likely require a custom software suite to accomplish. However, once this is done the database will be much faster for data retrieval. We will be using this database for our client portal development.

2. Process Application

To realize this project, we will be using an imaginary health institution called CHARAKA HOSPITALS. CHARAKA hospitals IT department developed a process application using IBM BPM software. But the user experience in process portal that was shipped OOTB is not according to their specifications. They are looking for rich looking and highly customizable User Interfaces with little or no impact when they migrate to higher version of IBM BPM.

Four departments (Front Desk, Nurse, Physician, and Medical Lab) will participate in this process flow. In each department, there will be N number of employees. When patient arrives at the front desk, front desk receptionist will log in the process portal (either default process portal or the one developed using Node JS) register the patient information such as address and insurance information. Once check In task is finished a new task is created automatically for Nurse department. The task routing can be of round-robin or load-balance criteria. Now nurse will take the vital parameters of the patient such as temperature, heart-beat, height and weight. Based on the patient condition nurse will forward the patient to the physician's department. A new task will be created to the doctor, now doctor can start diagnosing the patient. If doctor feels that a medical test is needed to treat the patient he/she will create a task for the Medical Lab department. Once the medical tests are finished on the patient, Medical lab department sends the test results back to the doctor. If no further medical tests are needed, a new task will be created for the front desk department for billing purposes.

Please find the complete process details in the Process Documentation document.

In this process, there human tasks indicated with the activity with the human symbol on the top left corner, system activity is indicated with double gear symbol and once decision gateway. Each human task is associated with one user interface (UI). These interfaces can be implemented using Node JS technology.

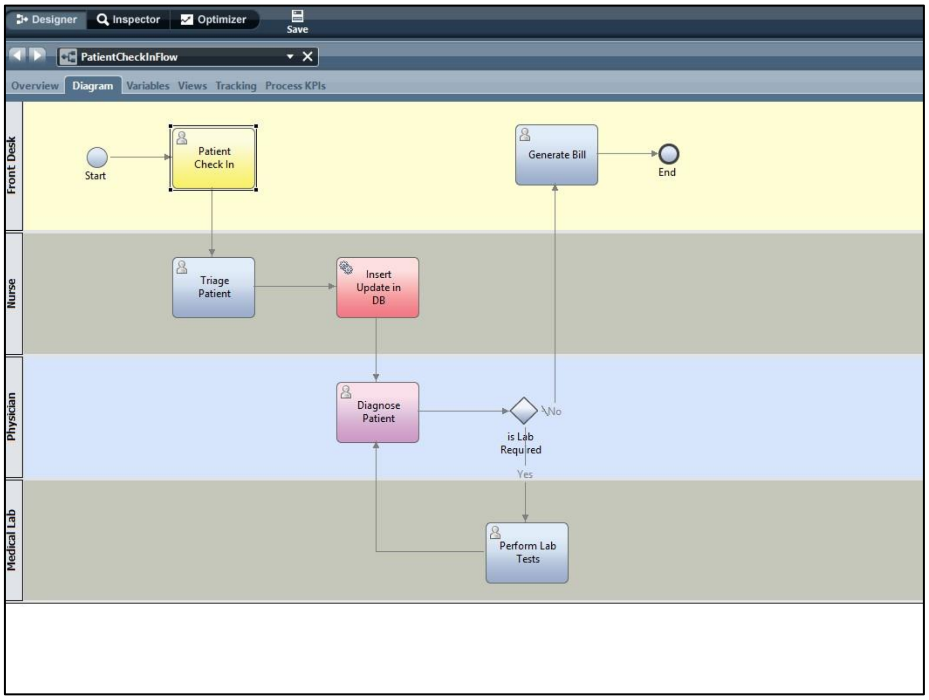


Figure 3: Patient Check In process overview [3].

3. Application Development

There are several phases involved to develop responsive portal. Following are the high-level steps.

1. Installation of pre-required software;
2. Preparing REST URLs;
3. Installing node js modules;
4. Developing User Interfaces;
5. Implementing back end code.

This portal is developed using The Model/View/Controller (MVC) pattern. The main objective of the MVC design pattern is separation of concerns. It provides an isolation of the application's presentation layer that displays the data in the user interface, from the way the data is actually processed.

3.1 Installation of Pre-required Software

To develop this portal, we need following software modules to be installed on the system. It does not matter which operating systems they are installed on (Windows/Linux).

1. IBM BPM Process Center/Server;
2. Node JS;
3. NPM.

BPM software is necessary as it acts as back end for Node JS portal. Node JS serves as cross-platform runtime environment for development of the portal. NPM (the node package manager) is the default package manager for the JavaScript runtime environment Node.js. It is an online repository for the publishing of open-source Node.js projects; second, it is a command-line utility for interacting with said repository that aids in package installation, version management, and dependency management.

Please visit IBM BPM knowledge site for the procedure to install IBM BPM process center & Process Server. Please see Appendix A for the installation procedure for Node JS and NPM software

3.2 Preparing REST URLs

IBM Business Process Manager provides a set of APIs that are implemented using Representational State Transfer (REST) services. A set of business process definition (BPD) related REST resources are provided for accessing business process, human task and business category data. These REST APIs enable developers to build a user interface or customize an existing portal application.

The APIs are simple enough to be called from mobile devices or from Rich Internet Applications (RIAs) such as our node js portal.

Following schematic shows the REST URL communication between the BPM and Node JS Portal. Please note that we can implement this communication in non-encrypted way (HTTP) and encrypted way (HTTPS) depending upon the security requirements.

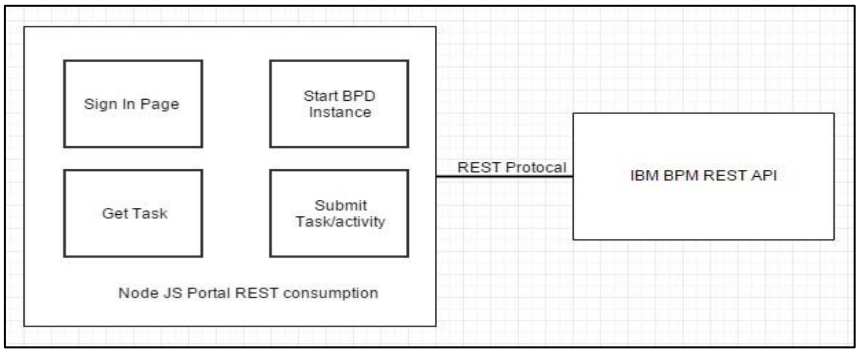


Figure 4: REST communication between Node JS & BPM [4].

Sign in Page: Sign In page is the point of entry to Node JS portal. All users who wish to use portal must present username and password to log in. Since we are using BPM as back end for all the application logic there is no need to configure separate authentication and authorization system such as LDAP with Node JS portal. When user enters credentials, we make REST call to the BPM to check user authenticity, if BPM returns a status code 200, it allows to log in if BPM return 400 code, user cant log in.

To implement this authentication mechanism passport module is used. Please see appendix B authentication section for the implementation of this REST call.

Start BPD Instance: A BPD is a reusable model of a process that defines the common aspects of all runtime instances of that process model. Start URL is exposed by IBM BPM via REST API to start a BPD or trigger an instance for BPD from an external system. In order to Use the start URL one should know the parameters like bodied, snapshot Id and Process App Id, which may change every time a new snapshot version is deployed. This is one of the disadvantage of using above method. Please see the Appendix B start BPD Instance source code section for complete implementation of this REST call.

GET Tasks: Once the BPD instance is started, the first task will be created for the front desk department. When a user that belongs to Front Desk Department logs in, the UI should display all the tasks that are assigned to that user. To populate these tasks in table a REST call has to be made to BPM server. Once these tasks are populated in the table, the user may wish to complete task. When a user clicks on specific task in the table, all the details about that task should be displayed in the task details section. This means another REST call has to be made to get the task details.

Please see the Get Tasks Source Code section for the implementation of these two REST calls. Please note that the code snippet shows only for Front Desk Department get task details. Similar calls has to be made for all the departments. Please see the complete project code for all the REST calls implementation.

Submit Task/Activity: Once the task opens for the user, the user may have to fill in the fields in that task. For example the nurse has to fill in the patients vital parameters such as temperature, hear-beat, height, weight etc. This data should be captured these details from UI, parse into JSON format which can be understood by IBM BPM server before making Post REST call. Please see the Appendix B Submit Task/Activity Source Code for the implementation details for Nurse Submit Task.

3.3 Creating Node JS Project

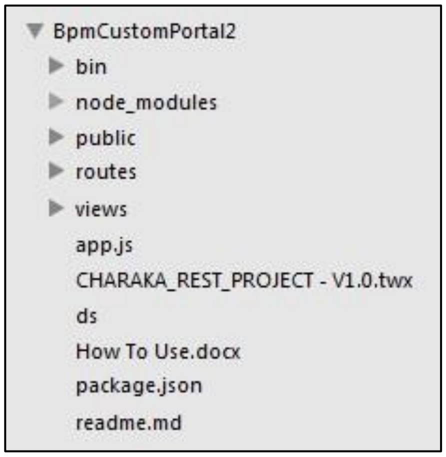
Node Js portal project structure developed based on the Model-View-Controller (MVC) design pattern. This pattern is great for separating the responsibility of the different parts of app and makes code easier to maintain.

In Figure 4, Node JS project structure is shown. Following is the break down the intent of each item:

bin: Anything that does not fit nicely inside of an npm script.

node modules: This folder holds all the dependent node modules. *public:* contains all static files like images, styles and JavaScript.

routes: This folder holds all the back end code logic such as making REST calls to the servers, parsing data, filtering date, etc.



```
▼ BpmCustomPortal2
  ▶ bin
  ▶ node_modules
  ▶ public
  ▶ routes
  ▶ views
    app.js
    CHARAKA_REST_PROJECT - V1.0.twx
    ds
    How To Use.docx
    package.json
    readme.md
```

Figure 5: Node JS Portal project structure [5].

views: Provides templates which are rendered and served by your routes. *app.js:* Initializes the app and glues everything together.

CHARAKA _REST PROJECT V1.0.twx: This is the project developed using IBM BPM software.

How to Use: This document describes how to use this Node JS portal. *package.json:* This file remembers all packages that your app depends on and their versions.

Readme.md: This file describes about the version level of the project and changes from the previous versions.

To download the complete project please visit following website. https://github.com/harishfysx/BpmCustomPortal2

3.4 Developing User Interfaces

In this project to develop user interfaces (UIs) EJS template engine is used rather than traditional HTML technology. EJS combines data and a template to produce HTML. The beauty of EJS is that, you can create partial views using EJS. For example, you can have a common header, footer, navigation for all pages and just change the internal content using EJS. Also, you are able to pass data to views. For instance, consider the username, which in case of each user is different. Using EJS you can do something like this.

```
App .get('/', function(req, res){ res.render('index',{user: "John Smith"})

});
```

The above code sets the dynamic username each time. In this project, we have several UIs implemented. These UIs reside in views folder in the project with. ejs extension. All UIs share the same layout. These layouts are in /partials folder. For complete code and how they render in browser please see User Interfaces section in Appendix C.

3.5 Implementing Back End Code

We use the REST URLs that we prepared in previous section. Node-xhr module is used to consume the REST calls. We use GET, POST, PUT methods in order to retrieve, send and update the data in BPM server from Node JS portal. Each API call takes a set of data in JSON format and returns data also in JSON format. The exception to this is a GET request, which instead takes its data as a query string. Because this is NodeJS, all of the functionality is handled asynchronously; we need to set up handlers to accept incoming data, as well as to deal with the response once it's complete. In this case, incoming data is simple. We just have to accept the data and concatenate it onto a string. Once the transfer is done, we can take that string and parse the JSON back into JavaScript objects we can work with. Finally, we call the success callback with this response info. Please see Back End Code in Appendix D.

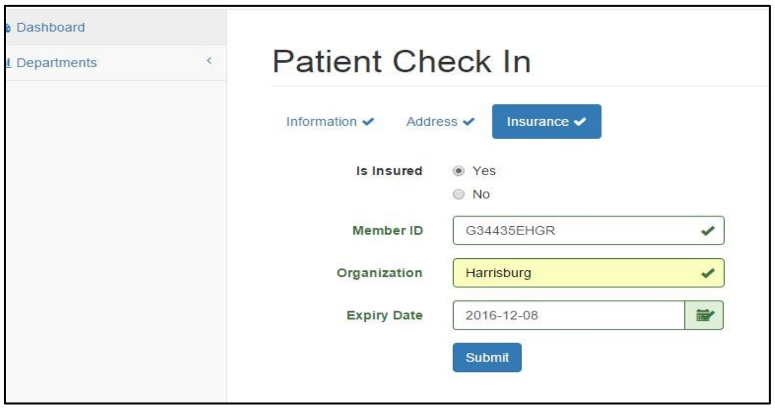
4. Testing Application

Once the application is installed and configured as specified in Appendix A, it can be tested as described following. Please see the” How to Use” document for additional information.

4.1 Creating Process Instance

As specified in the CHARAKA REST PROJECT, the Front Desk Team member will be able to start the process instance. So log in as front desk team member. In my case I am logging in as pcp test frontdesk1.

To kick start the process you need to fill the patient details. Please notice the validations implemented with Bootstrap validator frame work. Once you fill all the details click on submit button. This will start the process and assign the task to the front desk team. You can check this in the process inspector.



Patient Check In

Is Insured: [✓ Yes] No

Member ID: G34435EHGR

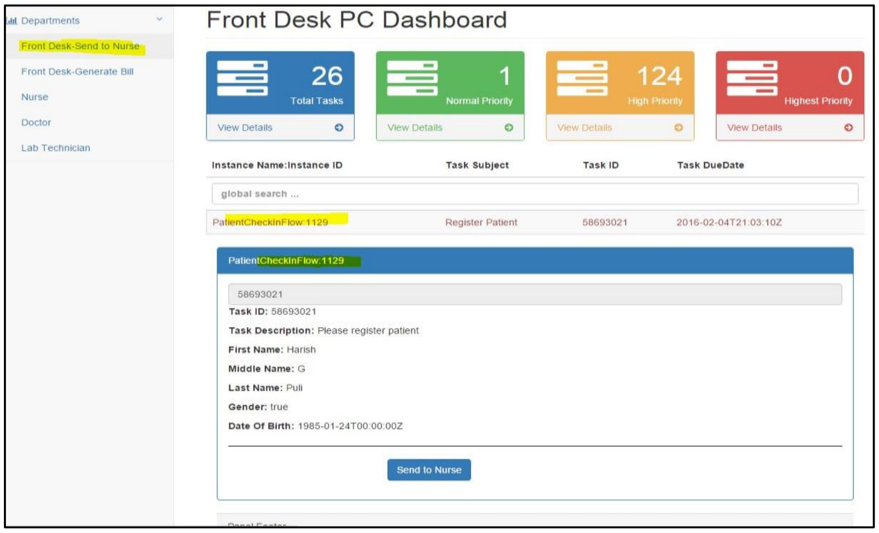
Organization: Harrisburg

Expiry Date: 2016-12-08

Figure 6: Kick starting Process Instance [6].

Now go to Front Desk-send to Nurse Page in the navigator. You can search for the tasks assigned to front desk team as well the user logged in. You can see the Normal Priority Tasks and Highest Priority tasks were only implemented here.

To assign the patient to the Nurse click on Send to Nurse button (make sure you are still logged in as front desk team member)This closes the task and creates new task to the Nurse. You can view this instance and task in the IBM.



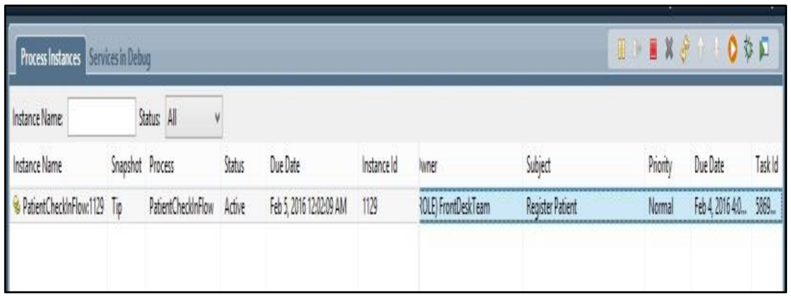
Instance Name:Instance ID	Task Subject	Task ID	Task DueDate
PatientCheckInFlow:1129	Register Patient	58693021	2016-02-04T21:03:10Z

PatientCheckInFlow:1129

58693021
Task ID: 58693021
Task Description: Please register patient
First Name: Harish
Middle Name: G
Last Name: Puli
Gender: true
Date Of Birth: 1985-01-24T00:00:00Z

Figure 7: Completing Front Desk task [7].

BPM process inspector view.



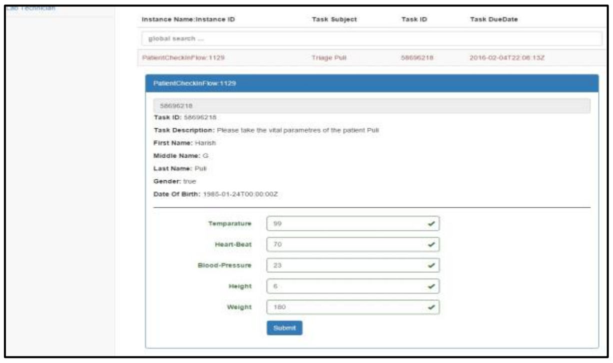
Instance Name: Status: All

Instance Name	Snapshot	Process	Status	Due Date	Instance Id	wner	Subject	Priority	Due Date	Task Id
PatientCheckInFlow:1129	Tip	PatientCheckInFlow	Active	Feb 5, 2016 12:02:09 AM	1129	OLE) FrontDeskTeam	Register Patient	Normal	Feb 4, 2016 4:0...	5869...

Figure 8: Process Inspector View 1 [7].

Once this task is completed the task is assigned to nurse department.

4.2 Nurse to Doctor



Instance Name:Instance ID	Task Subject	Task ID	Task DueDate
global search ...			
PatientCheckInFlow:1129	Triage Puli	58696218	2016-02-04T22:06:13Z

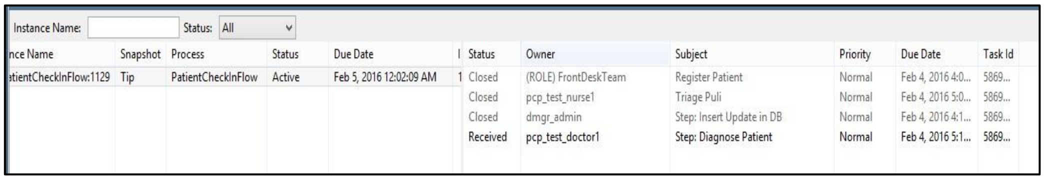
PatientCheckInFlow:1129

58696218
Task ID: 58696218
Task Description: Please take the vital parametres of the patient Puli
First Name: Harish
Middle Name: G
Last Name: Puli
Gender: true
Date Of Birth: 1985-01-24T00:00:00Z

Temparature	99
Heart-Beat	70
Blood-Pressure	23
Height	6
Weight	180

Figure 9: Nurse UI View [8].

Now log in as nurse and see the tasks assigned in the smart-table. You can perform global search on this table. Fill the vital parameters and click on submit. This will close the current task and creates new task for the doctor.



Instance Name: Status: All

nce Name	Snapshot	Process	Status	Due Date	I
atientCheckInFlow:1129	Tip	PatientCheckInFlow	Active	Feb 5, 2016 12:02:09 AM	1

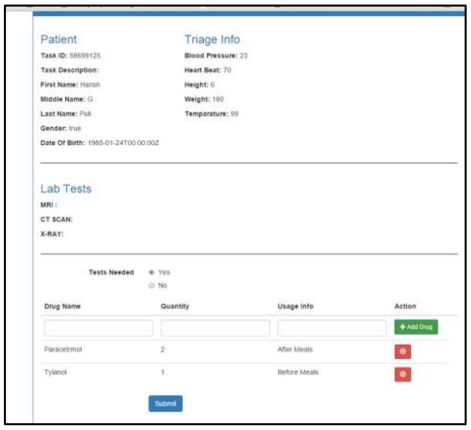
Status	Owner	Subject	Priority	Due Date	Task Id
Closed	(ROLE) FrontDeskTeam	Register Patient	Normal	Feb 4, 2016 4:0...	5869...
Closed	pcp_test_nurse1	Triage Puli	Normal	Feb 4, 2016 5:0...	5869...
Closed	dmgr_admin	Step: Insert Update in DB	Normal	Feb 4, 2016 4:1...	5869...
Received	pcp_test_doctor1	Step: Diagnose Patient	Normal	Feb 4, 2016 5:1...	5869...

Figure 10: Process Inspector View 2 [8].

In the process inspector view of IBM BPM you can see the task is finished.

4.3 Doctor to Front Desk

As you can see in the process-inspector the task is created for pcp test doctor1. Log in as that user and perform the task. In this screen you can see the table with add and remove button is implemented and all the data from Front Desk & Nurse tasks were carried forward to the doctor’s task. In the background we implemented BPM REST API.



Patient

Task ID: 58699125
Task Description:
First Name: Harish
Middle Name: G
Last Name: Puli
Gender: true
Date Of Birth: 1985-01-24T00:00:00Z

Triage Info

Blood Pressure: 23
Heart Beat: 70
Height: 6
Weight: 180
Temparature: 99

Lab Tests

MRI :
CT SCAN:
X-RAY:

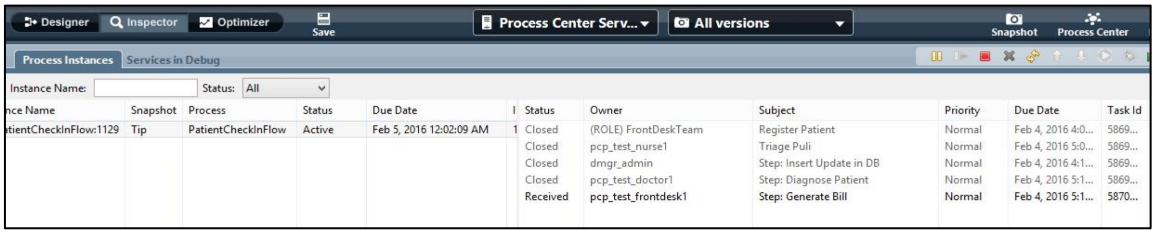
Tests Needed [✓ Yes] No

Drug Name	Quantity	Usage Info	Action
			+ Add Drug
Paracetrmol	2	After Meals	
Tylanol	1	Before Meals	

Submit

Figure 11: Doctor UI View [9].

Once you submit task, the task can go either to Lab Technician team or to Front Desk team depending on Tests Needed Flg. Let us assume no tests are needed in this case. Then the task is routed to front desk team.



Process Instances | Services in Debug

Instance Name: Status: All

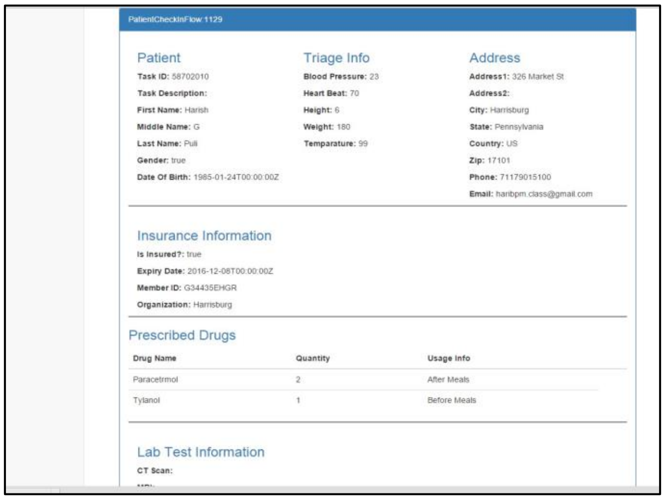
nce Name	Snapshot	Process	Status	Due Date
tientCheckInFlow:1129	Tip	PatientCheckInFlow	Active	Feb 5, 2016 12:02:09 AM

Status	Owner	Subject	Priority	Due Date	Task Id
Closed	(ROLE) FrontDeskTeam	Register Patient	Normal	Feb 4, 2016 4:0...	5869...
Closed	pcp_test_nurse1	Triage Puli	Normal	Feb 4, 2016 5:0...	5869...
Closed	dmgr_admin	Step: Insert Update in DB	Normal	Feb 4, 2016 4:1...	5869...
Closed	pcp_test_doctor1	Step: Diagnose Patient	Normal	Feb 4, 2016 5:1...	5869...
Received	pcp_test_frontdesk1	Step: Generate Bill	Normal	Feb 4, 2016 5:1...	5870...

Figure 12: Process Inspector View 3.

4.4 Front Desk to Patient

Log in as front desk team member In this screen we will show all the details filled by all departments. When the front desk team member clicks on the Dispatch button, the flow ends.



PatientCheckInFlow:1129

Patient

Task ID: 58702010

Task Description:

First Name: Harish

Middle Name: G

Last Name: Puli

Gender: true

Date Of Birth: 1985-01-24T00:00:00Z

Triage Info

Blood Pressure: 23

Heart Beat: 70

Height: 6

Weight: 180

Temparature: 99

Address

Address1: 326 Market St

Address2:

City: Harrisburg

State: Pennsylvania

Country: US

Zip: 17101

Phone: 71179015100

Email: haribpm.class@gmail.com

Insurance Information

Is Insured?: true

Expiry Date: 2016-12-08T00:00:00Z

Member ID: G34435EHGR

Organization: Harrisburg

Prescribed Drugs

Drug Name	Quantity	Usage Info
Paracetrmol	2	After Meals
Tylanol	1	Before Meals

Lab Test Information

CT Scan:

Figure 13: Front Desk Generate Bill UI View [10].

5. Conclusion

A stand alone, responsive and rich looking portal is developed and published (https://github.com/harishfysx/BpmCustomPortal2) which can be installed on any platform. Developers across the world can download this application and extend more features to it. Users who use this software can provide feedback in the form which they can find on the how to use document.

References

[1] L. Dyer, F. Henry, I. Lehmann, G. Lip, F. Osmani, D. Parrott, W. Peeters, and J. Zahn, "Scaling BPM adoption: From project to program with IBM business process manager," IBM Redbooks, Digital File, 2012.

[2] A. Q. Haviv, *MEAN Web Development.* Packt Publishing, Ltd., 2014.

[3] G. Hohpe and B. Woolf, *Enterprise Integration Patterns.* AddisonWesley, 2012.

[4] Dr. A. Arsanjani, N. Bharade, M. Borgenstrand, P. Schume, J. K. Wood, and V. Zheltonogov, *Business Process Management Design Guide.* IBM Redbooks, Digital File, 2015.

[5] R. Petersen, *Red Hat Enterprise Linux 6: Desktop and Administration.* Surfing Turtle Press, 2011.

[6] K. Kanoun and L. Spainhower, *Dependability Benchmarking for Computer Systems.* John Wiley & Sons, 2008.

[7] R. Bowen and K. Coar, *Apache Cookbook: Solutions and Examples for Apache Administration.* O'Reilly Media, Inc., 2007.

[8] D. Galin, *Software Quality Assurance: From Theory to Implementation.* Pearson/ Addison Wesley, 2004.

[9] E. Eloff and D. Torstensson, "An Investigation into the Applicability of Node.js as a Platform for Web Services," M.S. thesis, Department of Computer and Information Science, Department of Computer and Information Science, 2012.

[10] @article teddziuba website, Author = Dziuba, Title = Node.js is cancer, Publisher = http://teddziuba.com/2011/10/node-js-is-cancer. html

[11] @article joelonsoftware website, Author = Sparsky, J, Title = Human task switches considered harmful, Publisher = http://www.joelonsoftware. com/articles/fog0000000022.html

[12] @article nelhage blog, Author = Elhage, N, Title = why-node-js-is-cool, Publisher = http://blog.nelhage.com/2012/03/ why-node-js-is-cool

[13] @article IBM Technical Topics, Author = Nikhil, Thaker, Title = Using the REST APIs in IBM Business Process Manager V7.5, Publisher = http://www.ibm.com/ developerworks/websphere/ library/techarticles/1108_thaker/1108_thaker.html

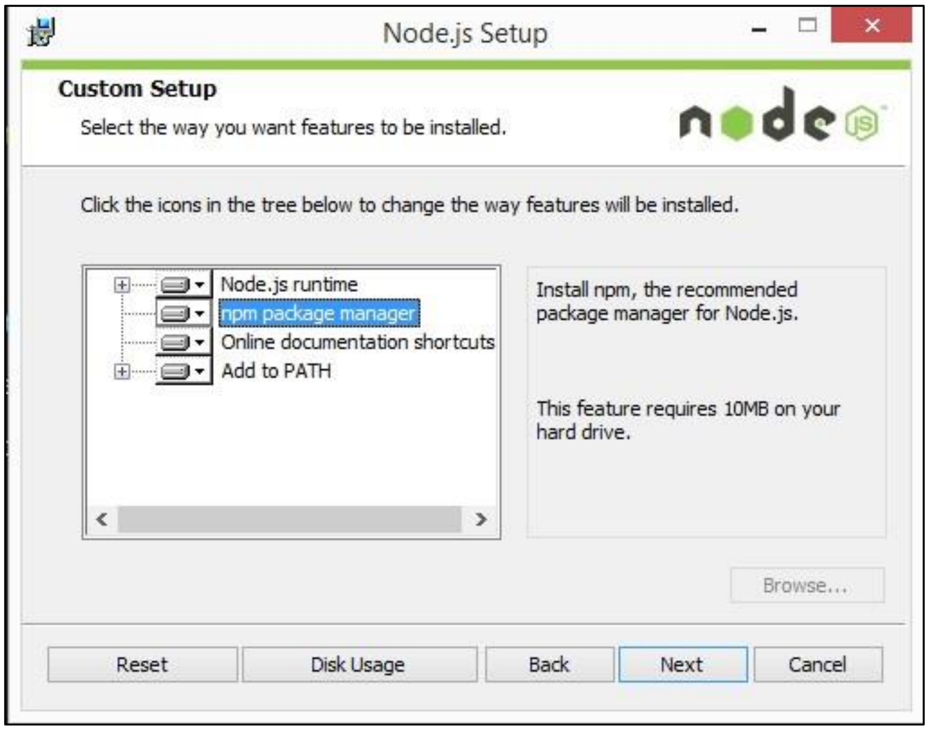
[14] @article IBM Technical Topics, Author = Stefan, Title = Best practices for Express app structure, Publisher = https://www.terlici.com/2014/08/25/best-practices-express-structure.html

[15] @article IBM Technical Topics, Author = Naeem, Title = Node.js With Express And EJS, Publisher = https://www.codementor.io/nodejs/ tutorial/node-with-express-and-ejs

Appendix A: Node JS Installation Procedure

Node JS Installation procedure

1. Download the Windows installer from the Nodes.js web site.
2. Run the installer (the .msi file you downloaded in the previous step.)
3. Follow the prompts in the installer (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings).


Node.js Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

- Node.js runtime
- npm package manager
- Online documentation shortcuts
- Add to PATH

Install npm, the recommended package manager for Node.js.

This feature requires 10MB on your hard drive.

Figure 14: Node JS Installation Wizard [10].

4. Restart your computer. You will not be able to run Node.js until you restart your computer.

Make sure you have Node and NPM installed by running simple commands to see what version of each is installed and to run a simple test program:

Test Node: To see if Node is installed, open the Windows Command Prompt, PowerShell or a similar command line tool, and type node -v. This should print a version number, so you'll see something like this v0.10.35.

Test NPM: To see if NPM is installed, type npm -v in Terminal. This should print NPMs version number so you'll see something like this 1.4.28.

Test Sample Code: Create a test file and run it. A simple way to test that node.js works is to create a JavaScript file: name it hello.js, and just add the code console.log ('Node is installed!');. To run the code simply open your command line program, navigate to the folder where you save the file and type node hello.js. This will start Node and run the code in the hello.js file. You should see the output Node is installed!

Appendix B: Authentication Source Code

Authentication Source Code

```
passport.serializeUser(function(user, done) { done(null, user); });

passport.deserializeUser(function(user, done) { done(null, user);

});

//passport to use local strategy passport.use(new LocalStrategy({

passReqToCallback: true

}, function(req, username, password, done) { var auth = "Basic " + new

   Buffer(username + ":" +

          password).toString("base64"); app.locals.userName = username;

   app.locals.auth = auth;

   // check in mongo if a user with username exists or not

   xhr.get({ url: config.baseUrl + '/bpm/wle/v1/user/current', headers: {

         'Content-Type': 'application/json',

         'Authorization': auth

      },

   }, function(err, res) { if (err) {

         //console.log(err.message); return;

      } if (JSON.stringify(res.status.code) == 401) {

         //console.log(res); return done(null, null, req.flash('message', 'Invalid Username Or

                Password'))

      } else { var user = res.body.data; req.user =

         res.body.data.userName;

         //console.log("harish log"+res.body.data.userName);

         //console.log(req.user) return done(null, user);
```

```
        }

    }); }));
```

---

start BPD Instance source code

```
var express = require('express'); var passport = require('passport'); var
ensureAuth = require('connect-ensure-login'); var xhr = require('node-xhr');
var favicon = require('serve-favicon');

var config = require('./config'); var router = express.Router(); var a = require("array-tools");
router.get('/', ensureAuth.ensureLoggedIn('/login'), function(req, res, next) {
    //console.log(res.locals.userName)
    res.render('pages/home', { pageHeader: 'Patient Check In'
    });
}); //login page router.get('/login', function(req, res, next) {
res.render('pages/login', { message: req.flash('message')
    }); }); router.post('/login', passport.authenticate('local', { successRedirect: '/',
failureRedirect: '/login', // see text failureFlash: true // optional, see text as well
})); router.get('/logout', function(req, res) { req.logout();
res.redirect('/login');
});
router.post('/startInst', function(req, res, next) { console.log(req.body) var
    patientData = {
        "person": {
            "firstName": req.body.firstName,
            "lastName": req.body.lastName,
            "middleName": req.body.midName,
            "gender": req.body.gender,
            "dateOfBirth": req.body.dob
        },
        "address": {
            "address1": req.body.address1,
            "address2": req.body.address2,
            "city": req.body.city,
            "state": req.body.state,
            "country": req.body.country,
            "zip": req.body.zip,
            "phone": req.body.phone,
            "email": req.body.email
        },
        "insuranceInfo": {
            "isInsured": req.body.isInsured,
            "memberId": req.body.memberId,
            "organization": req.body.organization,
            "expiration": req.body.insExpDate
        } } console.log(patientData) xhr.post({ url: config.baseUrl +
    '/bpm/wle/v1/process?', headers: {
            'Content-Type': 'application/json',
            'Authorization': res.locals.auth
        }, params: { action: 'start', bpdId: config.bpdId,
        branchId: config.branchId, params: patientData,
        parts: 'all'
        },
    }, function(err, resp) { if (err) {
            console.log(err.message); return;
        }
        //console.log(resp) //console.log(resp.body.data.totalCount); if
        (resp.status.code == 200) { res.redirect('/');

        } else if (resp.status.code == 401) { res.render('pages/401');
        } else {
            res.json('something went wrong');
        }
        }
    });
    //console.log(resp); }); module.exports =
router;
```

Get Tasks Source Code

```
var express = require('express'); var passport = require('passport'); var favicon = require('serve-
favicon'); var ensureAuth = require('connect-ensure-login'); var bodyParser = require('body-parser'); var
xhr = require('node-xhr'); var config = require('./config'); var a = require("array-tools"); var router =
express.Router(); router.get('/', ensureAuth.ensureLoggedIn('/login'), function(req, res, next) {
res.render('pages/frontDesk', { pageHeader: 'Front Desk PC Dashboard'
    });
}); router.get('/logout', function(req, res) { req.logout();
res.redirect('/login');
}) router.get('/success200', function(req, res) {
res.render('pages/success200');
}); router.post('/frontDeskTasks', function(req, res, next) {
    //console.log(res.locals.userName) xhr.put({ url: config.baseUrl +
    '/bpm/wle/v1/search/query/?', headers: {
            'Content-Type': 'application/json',

            'Authorization': res.locals.auth

        }, params: { condition: ['taskActivityName|Equals|Patient Check In',

                    'taskStatus|Equals|Received'], organization: 'byInstance'

        },

    }, function(err, resp) { if (err) {

            console.log(err.message); return;

        } var refinedData = {}; if (resp.body.data) { if (resp.body.data.data) { refinedData.totalData

        = resp.body.data.data; refinedData.normalTcount = a.where(resp.body.data.data, {

        taskPriority: "Normal" }).length; refinedData.highTcount = a.where(resp.body.data.data, {

        taskPriority: "High" }).length; refinedData.lowTcount = a.where(resp.body.data.data, {

        taskPriority: "Low"

                }).length;

            }

            //console.log("tasks were fetched") //console.log(resp.body.data.totalCount);

            res.json(refinedData);

        } else {

            res.json('soemthing went wrong');

        }

    });

    //console.log(resp);

    //res.render('pages/home',{tagline:req.user.userName||'test'});

});

//get specific task details router.get('/frontDeskTask/:id', function(req, res, next) {

console.log(res.locals.userName) xhr.get({ url: config.baseUrl + '/bpm/wle/v1/task/' +

req.params.id, headers: {

            'Content-Type': 'application/json',
            'Authorization': res.locals.auth

        }, params: { parts: 'all'

        },

    }, function(err, resp) { if (err) {

            console.log(err.message); return;

        } //console.log(resp.body.data);

        res.json(resp.body.data);

    });

});

//finish Front Desk Task router.post('/postFrntDeskTask', function(req, res, next) { xhr.put({ url:

config.baseUrl + '/bpm/wle/v1/task/' + req.body.tkiid + '?', headers: {


            'Content-Type': 'application/json',

            'Authorization': res.locals.auth

        }, params: { action: 'finish', parts:

        'all'

        },

    }, function(err, resp) { if (err) {

            console.log(err.message); return;

        } res.json(resp)

    }); }); module.exports = router;
```

Reference 11:@article joelonsoftware website, Author = Sparsky, J, Title = Human task switches considered harmful, Publisher = http://www.joelonsoftware.com/articles/fog0000000022.html

Submit Task/Activity Source Code

```
var express = require('express'); var passport = require('passport'); var
ensureAuth = require('connect-ensure-login'); var xhr = require('node-xhr');
var bodyParser = require("body-parser"); var favicon = require('serve-
favicon');

var config = require('./config'); var path = require('path'); var a = require("array-tools"); var router =
express.Router(); router.get('/', ensureAuth.ensureLoggedIn('/login'), function(req, res, next) {
res.render('pages/nurse', { pageHeader: 'Nurse Dashboard'
   }); }); router.get('/logout', function(req, res) { req.logout();
res.redirect('/login');
   })
   //get Nurse Tasks router.post('/nurseTasks', function(req, res, next) {
console.log(res.locals.userName) xhr.put({ url: config.baseUrl +
'/bpm/wle/v1/search/query/?', headers: {
         'Content-Type': 'application/json',
         'Authorization': res.locals.auth
      }, params: { condition:
      ['taskActivityName|Equ
      als|Triage Patient',

               'taskStatus|Equals|Received'], organization: 'byInstance'
      },
   }, function(err, resp) { if (err) {
         console.log(err.message); return;
      } var refinedData = {}; if (resp.body.data) { if (resp.body.data.data) { refinedData.totalData
      = resp.body.data.data; refinedData.normalTcount = a.where(resp.body.data.data, {
      taskPriority: "Normal" }).length; refinedData.highTcount = a.where(resp.body.data.data, {
        taskPriority: "High" }).length; refinedData.lowTcount = a.where(resp.body.data.data, {
        taskPriority: "Low"
            }).length; } res.json(refinedData);
        } else {
            res.json('soemthing went wrong');
        }
    });
});
//route to triage patient when clicked on work button
/* router.get('/workTask/:id',function(req,res){
        //console.log( req.params.id);
        //res.send("tagId is set to " + req.params.id); res.render('pages/triage',{pageHeader:'Triage
        Patient'})
});
*/
//get specific task details router.get('/nurseTask/:id', function(req, res, next) {
console.log(res.locals.userName) xhr.get({ url: config.baseUrl + '/bpm/wle/v1/task/' +
req.params.id, headers: {
            'Content-Type': 'application/json',
            'Authorization': res.locals.auth
        }, params: { parts: 'all'
        },
    }, function(err, resp) {
        if (err) {
            console.log(err.message); return;
        } //console.log(resp.body.data);
        res.json(resp.body.data);
```

```
  });

});

//post Triage Form(Patient Vital parameteres) router.post('/postTriage', function(req, res, next) {

  //console.log(res.locals.userName) xhr.put({ url: config.baseUrl + '/bpm/wle/v1/task/' +

  req.body.tkiid + '?', headers: {

      'Content-Type': 'application/json',

      'Authorization': res.locals.auth

    }, params: { action: 'finish', params:

    req.body.taskParam, parts: 'all'

    },

  }, function(err, resp) { if (err) {

      console.log(err.message);

      return;

    } res.json(resp)

  }); }); module.exports = router;
```

Appendix C: User Interfaces

User Interfaces

This section shows the complete code of the pages and how they render in the browser.

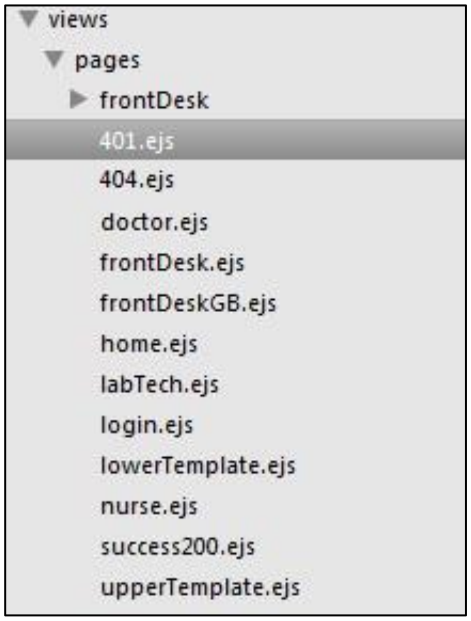
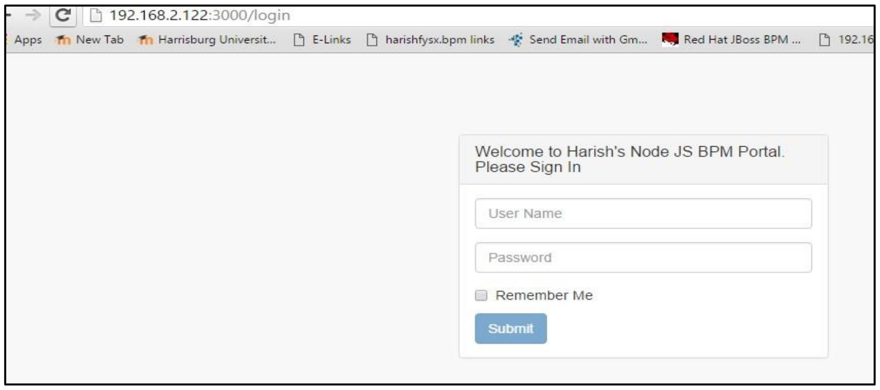

```
▼ views
  ▼ pages
    ▶ frontDesk
      401.ejs
      404.ejs
      doctor.ejs
      frontDesk.ejs
      frontDeskGB.ejs
      home.ejs
      labTech.ejs
      login.ejs
      lowerTemplate.ejs
      nurse.ejs
      success200.ejs
      upperTemplate.ejs
```

Figure 15: Pages in Node Js Portal [11].

Sign In Page

```
<!DOCTYPE html>

<html lang="en">


<head>
```



Welcome to Harish's Node JS BPM Portal. Please Sign In

Figure 16: Sign In Page [12].

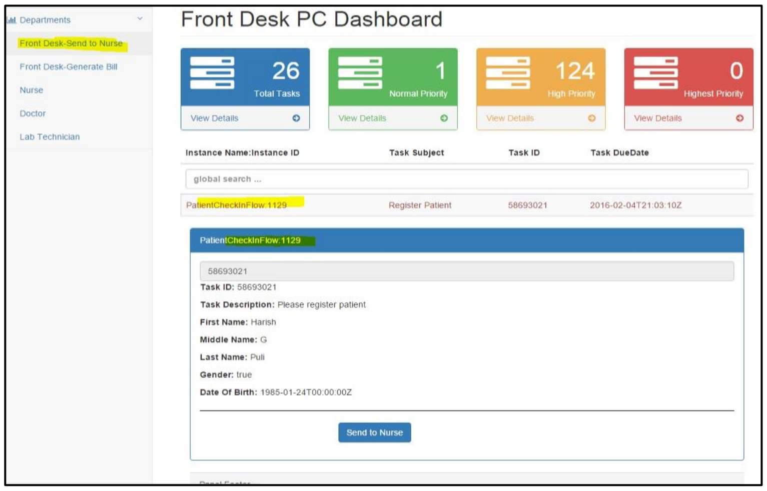
```
<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="description" content="">

<meta name="author" content="">


<title>SB Admin 2 - Bootstrap Admin Theme</title>


<!-- Bootstrap Core CSS -->

<link href="../bower_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- MetisMenu CSS -->

<link href="../bower_components/metisMenu/dist/metisMenu.min.css" rel="stylesheet">


<!-- Custom CSS -->

<link href="../dist/css/sb-admin-2.css" rel="stylesheet">


<!-- Custom Fonts -->

<link href="../bower_components/font-awesome/css/font-awesome.min.css" rel="stylesheet"

    type="text/css">
    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->

    <!-- WARNING: Respond.js doesn’t work if you view the page via file:// -->

    <!--[if lt IE 9]>

        <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>

        <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>

    <![endif]-->


</head>

<body>


    <div class="container">

        <div class="row">

            <div class="col-md-4 col-md-offset-4">

                <div class="login-panel panel panel-default">

                    <div class="panel-heading">

                                                <h3 class="panel-title">Welcome to Harish’s Node

                                JS BPM Portal. Please Sign In</h3>

                    </div>

                    <div class="panel-body">

                        <form action="/login" method="post" data-toggle="validator" role="form">

                            <fieldset>

                                        <div class="form-group">

                                    <input class="form-control" class="control-label"

                                         placeholder="User Name" name="username"

                                         type="username" data-error="Bruh, that email address is

                                         invalid" required>

                                      </div>
```

```
                                          <div class="form-group">
                                      <input class="form-control"
                                           placeholder="Password" name="password"
                                           type="password" value="" required>
                                  </div>
                                          <div class="checkbox">
                                             <label>
                                          <input name="remember" type="checkbox"
                                                value="Remember
                                                     Me">Remember Me
                                             </label>
                                  </div>
                                  <!-- Change this to a button or input when using this as a form -->
                                  <button type="submit" class="btn btn-primary">Submit</button>
                              </fieldset>
                          </form>
                                   <% if (message.length > 0) { %>
  <div class="alert alert-danger"><%= message %></div>
<% } %>
                      </div>
                  </div>
              </div>
      </div>
</div>
<!-- jQuery -->
<script src="../bower_components/jquery/dist/jquery.min.js"></script>
```

```
    <!-- Bootstrap Core JavaScript -->

    <script src="../bower_components/bootstrap/dist/js/bootstrap.min.js"></script>


    <!-- Metis Menu Plugin JavaScript -->

    <script src="../bower_components/metisMenu/dist/metisMenu.min.js"></script>


   <script src="../js/validator.js"></script>


    <!-- Custom Theme JavaScript -->

    <script src="../dist/js/sb-admin-2.js"></script>


</body>


</html> [13]
```

Front Desk Page

```
<% include upperTemplate %>
```



Front Desk PC Dashboard

Instance Name:Instance ID	Task Subject	Task ID	Task DueDate
global search ...			
PatientCheckInFlow:1129	Register Patient	58693021	2016-02-04T21:03:10Z

PatientCheckInFlow:1129

58693021

Task ID: 58693021

Task Description: Please register patient

First Name: Harish

Middle Name: G

Last Name: Puli

Gender: true

Date Of Birth: 1985-01-24T00:00:00Z

Figure 17: Sign In Page [14].

```
<div ng-app="frontDeskApp" ng-controller="safeCtrl">

    <!-- /.row -->

    <div class="row">

        <div class="col-lg-3 col-md-6">

            <div class="panel panel-primary">

                        <div class="panel-heading">

                            <div class="row">

                                <div class="col-xs-3">

                                                <i class="fa fa-tasks fa-5x"></i>

                        </div>

                                            <div class="col-xs-9 text-right">

                            <div class="huge">26</div>

                                    <div>Total Tasks</div>

                        </div>

                    </div>

                </div>

                <a href="#">

                    <div class="panel-footer">

                        <span class="pull-left">View Details</span>

                        <span class="pull-right"><i class="fa fa-arrow-circle-

                            right"></i></span>

                                <div class="clearfix"></div>

                    </div>

                </a>

            </div>

        </div>

        <div class="col-lg-3 col-md-6">
```

```
<div class="panel panel-green">



            <div class="panel-heading">
                <div class="row">
                            <div class="col-xs-3">
                        <i class="fa fa-tasks fa-5x"></i>
                    </div>
                    <div class="col-xs-9 text-right">
                                          <div class="huge">{{normalTCount}}</div>
                                <div>Normal Priority</div>
                    </div>
                </div>
            </div>
            <a href="#">
                <div class="panel-footer">
                    <span class="pull-left">View Details</span>
                    <span class="pull-right"><i class="fa fa-arrow-circle-
                         right"></i></span>
                    <div class="clearfix"></div>
                </div>
            </a>
        </div>
    </div>
    <div class="col-lg-3 col-md-6">
        <div class="panel panel-yellow">
            <div class="panel-heading">
```

```
                <div class="row">
                    <div class="col-xs-3">
                        <i class="fa fa-tasks fa-5x"></i>
                    </div>
                    <div class="col-xs-9 text-right">
                            <div class="huge">124</div>
                            <div>High Priority</div>
                    </div>
                </div>
            </div>
            <a href="#">
                <div class="panel-footer">
                    <span class="pull-left">View Details</span>
                    <span class="pull-right"><i class="fa fa-arrow-circle-
                        right"></i></span>
                            <div class="clearfix"></div>
                </div>
            </a>
        </div>
    </div>
    <div class="col-lg-3 col-md-6">
        <div class="panel panel-red">
            <div class="panel-heading">
                <div class="row">
                    <div class="col-xs-3">
                        <i class="fa fa-tasks fa-5x"></i>
                    </div>
```

```
                                        <div class="col-xs-9 text-right">

                                           <div class="huge">{{highTcount}}</div>

                                                  <div>Highest Priority</div>

                                        </div>

                                     </div>

                                  </div>

                                  <a href="#">

                                            <div class="panel-footer">

                                                           <span class="pull-left">View Details</span>

                                        <span class="pull-right"><i class="fa fa-arrow-circle-

                                            right"></i></span>

                                               <div class="clearfix"></div>

                                     </div>

                                  </a>

                               </div>

                            </div>

                         </div>

                         <!-- /.row -->


                    <table st-table="displayedCollection" st-safe-src="rowCollection" class="table table-striped">

                      <thead>

                      <tr>

                         <th st-sort="instanceName">Instance Name:Instance ID</th>

                         <th st-sort="taskSubjecte">Task Subject</th>

                         <th st-sort="taskId">Task ID</th>

                         <th st-sort="taskDueDate">Task DueDate</th>
```

```
    </tr>
    <tr>
        <th colspan="5"><input st-search="" class="form-control" placeholder="global search ..."
              type="text"/></th>
    </tr>
    </thead>
    <tbody>
    <tr ng-repeat="row in displayedCollection" ng-click="workTask(row,$index)" ng-class="{'alert
          alert-danger':$index== selectedRow}">

        <td>{{row.instanceName}}</td>
        <td>{{row.taskSubject}}</td>
        <td>{{row.taskId}}</td>
        <td>{{row.taskDueDate}}</td>

    </tr>
    </tbody>
</table>
    <div ng-controller="postFrontDeskCtrl" >
<div class="col-lg-12">
                    <div class="panel panel-primary">
                         <div class="panel-heading">
                                        {{processInstanceName}}
                         </div>
                         <div class="panel-body">
                  <input class="form-control" id="disabledInput" type="text" placeholder={{tkiid}} ng-
                        model="tkiid" disabled>
                    <p><strong>Task ID: </strong>{{tkiid}}</p>

                    <p><strong>Task Description: </strong>{{taskDesc}}</p>

                    <p><strong>First Name: </strong>{{patientFName}}</p>

                    <p><strong>Middle Name: </strong>{{patientMName}}</p>

                    <p><strong>Last Name: </strong>{{patientLName}}</p>

                    <p><strong>Gender: </strong>{{patientGender}}</p>

                    <p><strong>Date Of Birth: </strong>{{patientDob}}</p>

            <hr style="width: 100%; color: black; height: 1px; background-color:black;" />


            <form id="frontTaskForm" ng-submit="finishFrontDesk()" class="form-horizontal" >




                <div class="form-group">

                    <div class="col-lg-9 col-lg-offset-3">

                                        <button class="btn btn-primary">Send to Nurse

                                </button>

                    </div>

                </div>

            </form>

        </div>




                        </div>

                                <div class="panel-footer">

                                    Panel Footer

                        </div>

                    </div>

                </div>
```

```
  </div>
<% include lowerTemplate %>
    <script src="../controllers/frontDeskController.js"></script>
```

Appendix D: Configuration Back End Code

Configuration Back End Code [15]

```
module.exports = {
    //'url' :
                          'mongodb://<dbuser>:<dbpassword>@novus.modulusmongo.net:27017/<dbName>'
    'baseUrl' : 'http://192.168.2.140:9080/rest',
    'bpdId':'25.01dd33b6-4c2a-49ec-a186-15b89a761d8b',
    'branchId':'2063.8a73dc16-43e1-444f-a119-8c3170b8ddf1'



}
```

Doctor Page Back End Code

```
var express = require('express'); var passport = require('passport'); var
ensureAuth =require('connect-ensure-login'); var xhr = require('node-
xhr'); var bodyParser=require("body-parser"); var favicon = require('serve-
favicon'); var config = require('./config'); var path = require('path'); var a =
require("array-tools");

var router = express.Router();


router.get('/',ensureAuth.ensureLoggedIn('/login'), function(req, res, next) {
    res.render('pages/doctor',{pageHeader:'Doctor Dashboard'});
}); router.get('/logout', function(req, res) { req.logout();

res.redirect('/login');
```

```
    })



//get Doctor Tasks router.post('/doctorTasks',function(req, res, next) {

console.log(res.locals.userName)


    xhr.put({ url: config.baseUrl+'/bpm/wle/v1/search/query/?', headers: {

                'Content-Type': 'application/json',

                'Authorization': res.locals.auth

          }, params: { condition:

          ['taskActivityName|Equ

          als|Diagnose Patient',

                      'taskStatus|Equals|Received'], organization: 'byInstance'

          },

      }, function(err, resp) { if (err) {

                console.log(err.message); return;

          }


          var refinedData={};


          if(resp.body.data){ if(resp.body.data.data){


          refinedData.totalData=resp.body.data.data;

          refinedData.normalTcount=a.where(resp.body.data.data, { taskPriority: "Normal" }).length;

          refinedData.highTcount=a.where(resp.body.data.data, { taskPriority: "High" }).length;

          refinedData.lowTcount=a.where(resp.body.data.data, { taskPriority: "Low" }).length;

          }
```

```
            res.json(refinedData);

      } else{ res.json('soemthing went wrong');

            }

      } );


     });


//get specifi doctor task router.get('/doctorTask/:id',function(req, res, next) {

console.log(res.locals.userName)


     xhr.get({ url: config.baseUrl+'/bpm/wle/v1/task/'+req.params.id, headers: {

                  'Content-Type': 'application/json',

                  'Authorization': res.locals.auth

            }, params: {


                  parts: 'all'

            },

      }, function(err, resp) {

            if (err) {

                  console.log(err.message); return;

            }




            //console.log(resp.body.data);


            res.json(resp.body.data);
```

```
   } );

  });

//post (fnish) doctor task router.post('/postDoctor',function(req, res, next) {
  //console.log(res.locals.userName)

  xhr.put({ url: config.baseUrl+'/bpm/wle/v1/task/'+req.body.tkiid+'?', headers: {
          'Content-Type': 'application/json',
                            'Authorization': res.locals.auth
       }, params: { action: 'finish', params:
       req.body.taskParam, parts:'all'
       },
   }, function(err, resp) { if (err) {
          console.log(err.message); return;
       }

       res.json(resp)

   });

  });

module.exports = router;
```

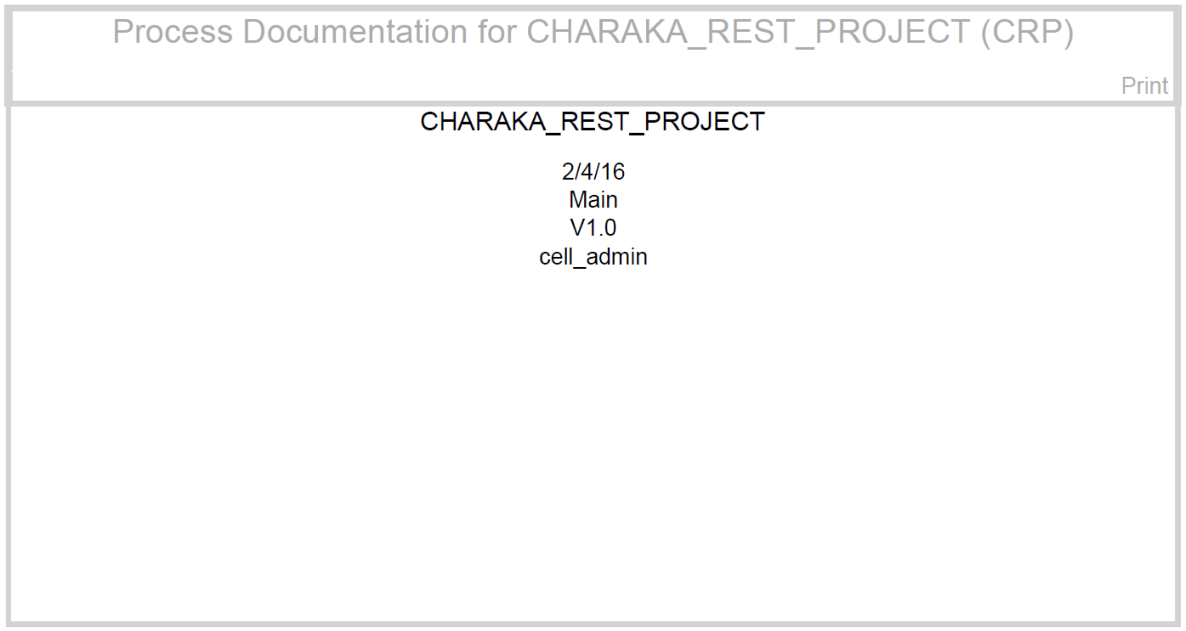
Lab Technician Page Back End Code

```
var express = require('express'); var passport = require('passport'); var
ensureAuth =require('connect-ensure-login'); var xhr = require('node-
xhr'); var bodyParser=require("body-parser"); var favicon = require('serve-
favicon'); var config = require('./config'); var path = require('path'); var a =
require("array-tools"); var router = express.Router();

router.get('/',ensureAuth.ensureLoggedIn('/login'), function(req, res, next) {
    res.render('pages/labTech',{pageHeader:'Lab Tech Dashboard'});
}); router.get('/logout', function(req, res) { req.logout();

res.redirect('/login');

    })



//get Nurse Tasks
router.post('/labTechTasks',function(req, res, next) { console.log(res.locals.userName)

    xhr.put({ url: config.baseUrl+'/bpm/wle/v1/search/query/?', headers: {
                    'Content-Type': 'application/json',
                    'Authorization': res.locals.auth
            }, params: { condition: ['taskActivityName|Equals|Perform Lab Tests',
                        'taskStatus|Equals|Received'], organization: 'byInstance'
            },
     }, function(err, resp) { if (err) {
                    console.log(err.message); return;
            }
            var refinedData={};


            if(resp.body.data){ if(resp.body.data.data){

            refinedData.totalData=resp.body.data.data;

            refinedData.normalTcount=a.where(resp.body.data.data, {

            taskPriority: "Normal" }).length;

            refinedData.highTcount=a.where(resp.body.data.data, {

            taskPriority: "High" }).length;

            refinedData.lowTcount=a.where(resp.body.data.data, { taskPriority:

            "Low" }).length;

            }


            res.json(refinedData);


        } else{ res.json('soemthing went wrong');

            }

        } );


      });

//route to triage patient when clicked on work button


/* router.get('/workTask/:id',function(req,res){

     //console.log( req.params.id);

     //res.send("tagId is set to " + req.params.id); res.render('pages/triage',{pageHeader:'Triage
     Patient'})

});

*/
```

```
//get specific task details router.get('/labTechTask/:id',function(req, res, next) {
console.log(res.locals.userName)


   xhr.get({ url: config.baseUrl+'/bpm/wle/v1/task/'+req.params.id, headers: {
                 'Content-Type': 'application/json',
                 'Authorization': res.locals.auth
            }, params: {


                 parts: 'all'
            },
     }, function(err, resp) { if (err) {
                 console.log(err.message); return;
            }
            //console.log(resp.body.data);


            res.json(resp.body.data);




     } );


   });


//post Triage Form(Patient Vital parameteres)


router.post('/postlabTech',function(req, res, next) {
   //console.log(res.locals.userName)


   xhr.put({ url: config.baseUrl+'/bpm/wle/v1/task/'+req.body.tkiid+'?', headers: {
```

```
                ’Content-Type’: ’application/json’,

                ’Authorization’: res.locals.auth

            }, params: { action: ’finish’, params:

            req.body.taskParam, parts:’all’


            },

        }, function(err, resp) { if (err) {

                console.log(err.message); return;

            }


            res.json(resp)




        } );


      });


module.exports = router;
```

Appendix E: Process Documentation



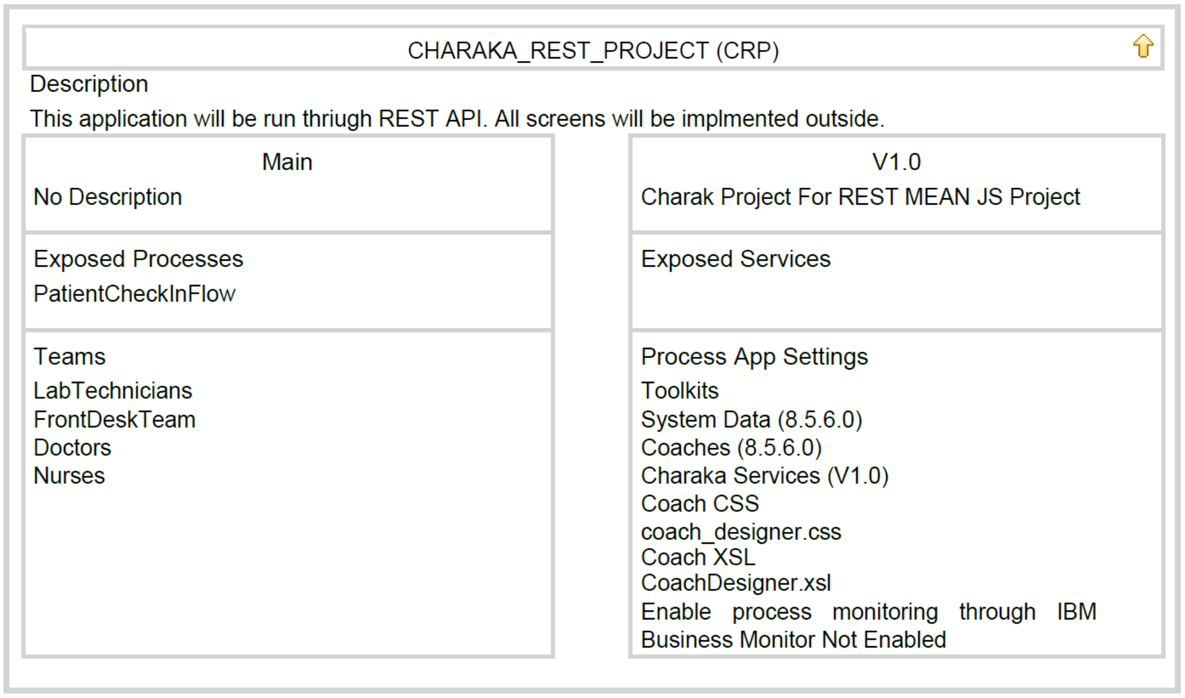
Process Documentation for CHARAKA_REST_PROJECT (CRP)

Print

CHARAKA_REST_PROJECT

2/4/16
Main
V1.0
cell_admin

Table of Contents



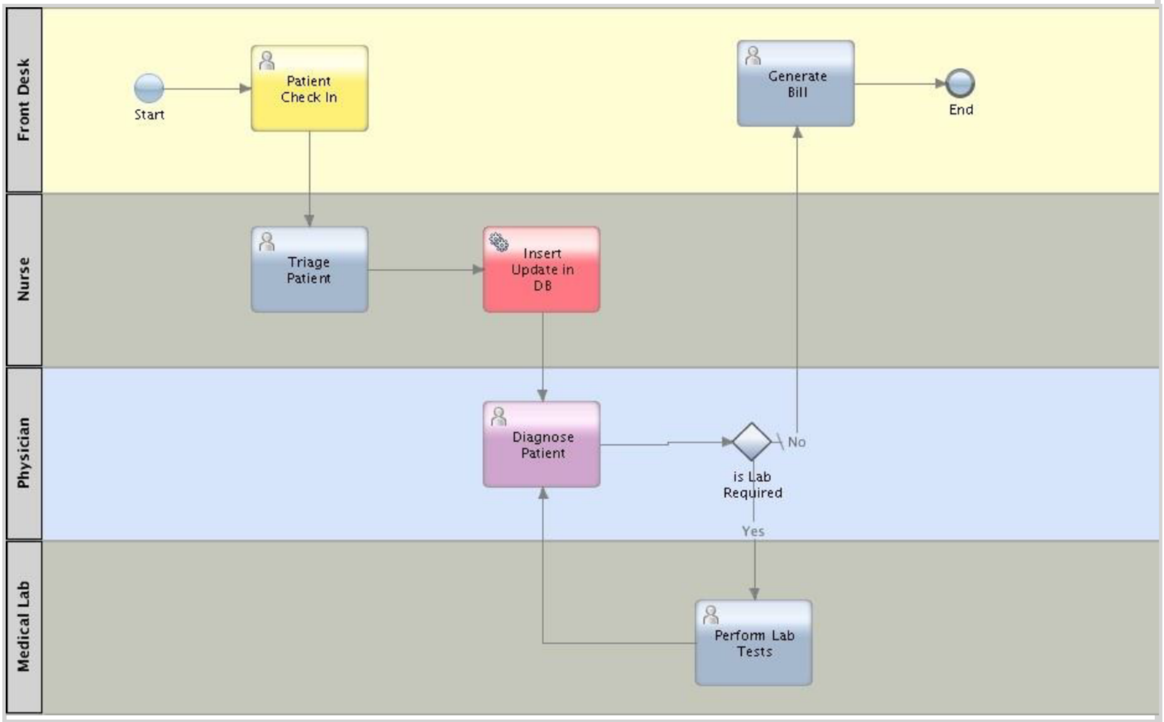
CHARAKA_REST_PROJECT (CRP)

Description

This application will be run thriugh REST API. All screens will be implmented outside.

Main	V1.0
No Description	Charak Project For REST MEAN JS Project
Exposed Processes PatientCheckInFlow	Exposed Services
Teams LabTechnicians FrontDeskTeam Doctors Nurses	Process App Settings Toolkits System Data (8.5.6.0) Coaches (8.5.6.0) Charaka Services (V1.0) Coach CSS coach_designer.css Coach XSL CoachDesigner.xsl Enable process monitoring through IBM Business Monitor Not Enabled

PatientCheckInFlow



Description

No Description

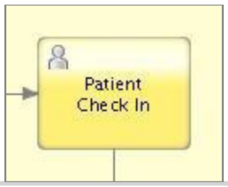
Inputs person (Person) address (Address) insuranceInfo (Insurance)	Outputs
Message Events	Ad Hoc Events
Integrations Insert Update in DB	Teams FrontDeskTeam Nurses Doctors LabTechnicians

Sub Processes and Activities

Patient Check In
Generate Bill
Triage Patient
Insert Update in DB
Diagnose Patient
Perform Lab Tests

PatientCheckInFlow

Patient Check In



User Task

In this task the front desk receptionist will enter the infromation of the patient.

Preconditions
Event :
NO_PRECONDITION

Expression :

Task Header
Subject :
Register Patient

Narrative :
Please register patient

Input Mapping tw.local.person ⇨ person tw.local.address ⇨ address tw.local.insuranceInfo ⇨ insuranceInfo

Output Mapping
person ⇨ tw.local.person
address ⇨ tw.local.address
insuranceInfo ⇨ tw.local.insuranceInfo

Attached Events

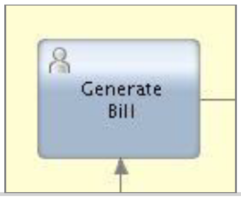
Routing
Assign To : Team
User Distribution : Last User

Associated Tags

Key Performance Indicator
Cost
Execution Time (Clock)
Labor Cost
Resource Cost
Rework
Total Time (Clock) Value Add
Wait Time (Clock)

PatientCheckInFlow

Generate Bill



User Task

No Description

Preconditions

Event :
NO_PRECONDITION

Expression :

Task Header

Subject :

Narrative :

Input Mapping

tw.local.diagInfo ⇨ diagInfo
tw.local.triageInfo ⇨ triageInfo
tw.local.medTestInfo ⇨ medTestInfo
tw.local.person ⇨ person
tw.local.address ⇨ address
tw.local.insuranceInfo ⇨ insuranceInfo

Output Mapping

diagInfo ⇨ tw.local.diagInfo
triageInfo ⇨ tw.local.triageInfo
medTestInfo ⇨ tw.local.medTestInfo
person ⇨ tw.local.person address ⇨
tw.local.address insuranceInfo ⇨
tw.local.insuranceInfo

Attached Events

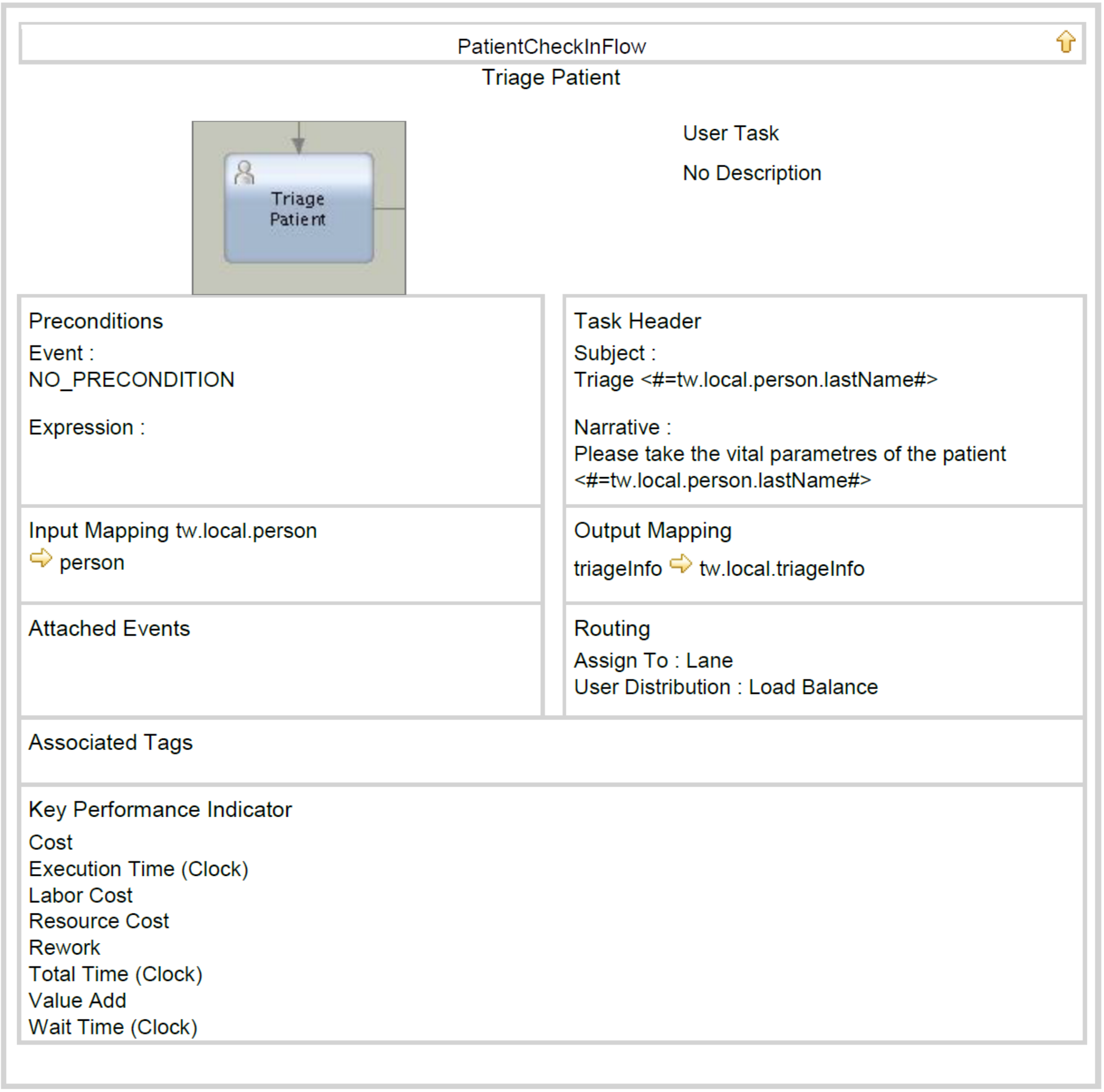
Routing

Assign To : Lane
User Distribution : Round Robin

Associated Tags

Key Performance Indicator

Cost
Execution Time (Clock)
Labor Cost
Resource Cost
Rework
Total Time (Clock)
Value Add
Wait Time (Clock)



PatientCheckInFlow

Triage Patient

User Task

No Description

Preconditions Event : NO_PRECONDITION Expression :	Task Header Subject : Triage <#=tw.local.person.lastName#> Narrative : Please take the vital parametres of the patient <#=tw.local.person.lastName#>
Input Mapping tw.local.person ⇨ person	Output Mapping triageInfo ⇨ tw.local.triageInfo
Attached Events	Routing Assign To : Lane User Distribution : Load Balance

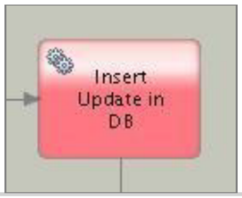
Associated Tags

Key Performance Indicator

Cost
Execution Time (Clock)
Labor Cost
Resource Cost
Rework
Total Time (Clock)
Value Add
Wait Time (Clock)

PatientCheckInFlow

Insert Update in DB



System Task

No Description

Preconditions Event : NO_PRECONDITION Expression :	Task Header Subject : Narrative :
Input Mapping tw.local.person ⇨ person tw.local.address ⇨ address	Output Mapping
Attached Events	Routing Assign To : Lane User Distribution : None

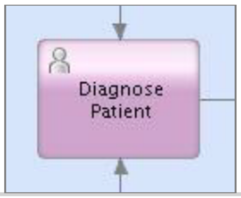
Associated Tags

Key Performance Indicator

Cost
Execution Time (Clock)
Labor Cost
Resource Cost
Rework
Total Time (Clock)
Value Add
Wait Time (Clock)

PatientCheckInFlow

Diagnose Patient



User Task

No Description

Preconditions
Event :
NO_PRECONDITION

Expression :

Task Header
Subject :

Narrative :

Input Mapping
tw.local.diagInfo ⇨ diagInfo
tw.local.triageInfo ⇨ triageInfo
tw.local.medTestInfo ⇨ medTestInfo
tw.local.person ⇨ person
tw.local.address ⇨ address
tw.local.insuranceInfo ⇨ insuranceInfo

Output Mapping
diagInfo ⇨ tw.local.diagInfo
triageInfo ⇨ tw.local.triageInfo
medTestInfo ⇨ tw.local.medTestInfo
person ⇨ tw.local.person address ⇨
tw.local.address insuranceInfo ⇨
tw.local.insuranceInfo

Attached Events

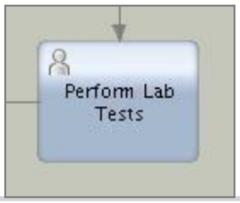
Routing
Assign To : Lane
User Distribution : Round Robin

Associated Tags

Key Performance Indicator
Cost
Execution Time (Clock)
Labor Cost
Resource Cost
Rework
Total Time (Clock)
Value Add
Wait Time (Clock)

PatientCheckInFlow

Perform Lab Tests



User Task

No Description

Preconditions

Event :
NO_PRECONDITION

Expression :

Task Header

Subject :

Narrative :

Input Mapping

tw.local.medTestInfo ⇨ medTestInfo tw.local.person ⇨ person tw.local.address ⇨ address

Output Mapping

medTestInfo ⇨ tw.local.medTestInfo
person ⇨ tw.local.person
address ⇨ tw.local.address

Attached Events

Routing

Assign To : Team
User Distribution : Round Robin

Associated Tags

Key Performance Indicator

Cost
Execution Time (Clock)
Labor Cost
Resource Cost
Rework
Total Time (Clock)
Value Add
Wait Time (Clock)

FrontDeskTeam

Description

Members - Standard Members
pcp_test_frontdesk4 (User)
pcp_test_frontdesk1 (User)
pcp_test_frontdesk3 (User)
pcp_test_frontdesk2 (User)

Nurses

Description

Members - Standard Members
PCP_Nurses (Group)

Doctors

Description

Members - Standard Members
PCP_Physicians (Group)

Lab Technicians

Description

Members - Standard Members
PCP_Lab Technicians (Group)