|  | | |
| --- | --- | --- |
|  | |  |
|  |  | **System Architecture Document** |
|  |  | Project: HCSC Small Group Renewals |
|  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 19-Mar-2018 | 0.1 | Draft Version | Infosys |

Disclaimer

The document would be the System Architecture (SA) reference and guidance document to serve the application developers get started with the high level design and architecture.

Table of Contents

[1. Executive Summary 7](#_Toc509294445)

[1.1 Objective 7](#_Toc509294446)

[1.2 Scope 7](#_Toc509294447)

[1.3 Key Considerations 7](#_Toc509294448)

[2. Application Tech Stack Overview 8](#_Toc509294449)

[2.1 Technology Blueprint 8](#_Toc509294450)

[2.2 SG Renewals Preference of the Technology Stack 8](#_Toc509294451)

[3. Architecture Details 10](#_Toc509294452)

[3.1 Architecture Principles and Design Guidelines 10](#_Toc509294453)

[3.2 Logical Architecture 11](#_Toc509294454)

[3.3 Physical Deployment Architecture 11](#_Toc509294455)

[3.4 Build Test and Release Architecture 12](#_Toc509294456)

[3.5 External Services 12](#_Toc509294457)

[3.5.1 Enterprise Services 12](#_Toc509294458)

[3.5.2 Other external services to be used by App 12](#_Toc509294459)

[3.6 Integration with ERP 12](#_Toc509294460)

[3.7 Security 12](#_Toc509294461)

[3.7.1 Authentication and Authorization 12](#_Toc509294462)

[3.7.2 Web Security 12](#_Toc509294463)

[3.7.3 [TBD] 12](#_Toc509294464)

[3.7.4 Other 12](#_Toc509294465)

[3.8 Performance and Scalability 12](#_Toc509294466)

[3.8.1 Scalability 12](#_Toc509294467)

[3.8.2 Performance 13](#_Toc509294468)

[3.8.3 Web Server 13](#_Toc509294469)

[3.9 Maintainability 13](#_Toc509294470)

[3.10 Availability and Reliability 13](#_Toc509294471)

[3.10.1 Availability 13](#_Toc509294472)

[3.10.2 Reliability 13](#_Toc509294473)

[3.11 Usability 13](#_Toc509294474)

[3.12 Integration and Deployment processes 13](#_Toc509294475)

[3.13 Database 13](#_Toc509294476)

[3.13.1 Database Schema 13](#_Toc509294477)

[3.14 Archival and Purging 13](#_Toc509294478)

[4. Target System 14](#_Toc509294479)

[4.1 Technical Feasibility 14](#_Toc509294480)

[4.2 Technology Stack and Technology versions 14](#_Toc509294481)

[4.3 Technical Recommendations 14](#_Toc509294482)

[4.3.1 Prioritization 14](#_Toc509294483)

[4.3.2 Application Physical Deployment Architecture 14](#_Toc509294484)

[4.3.3 UI Layer Architecture - Angular 14](#_Toc509294485)

[4.3.4 UI Exception Handling Design 14](#_Toc509294486)

[4.3.5 UI User Data Caching (transactional/non-transactional) 14](#_Toc509294487)

[4.3.6 Server Side and Micro services Architecture 14](#_Toc509294488)

[4.3.7 Application Logging Architecture & Design 14](#_Toc509294489)

[4.3.8 Application Exception Handling Architecture & Design 15](#_Toc509294490)

[4.3.9 Data caching options 16](#_Toc509294491)

[4.3.10 Code Quality/Testing Framework – UI 16](#_Toc509294492)

[4.3.11 Code Quality/Testing Framework – JAVA 16](#_Toc509294493)

[4.3.12 Application Monitoring 16](#_Toc509294494)

[4.3.13 Concurrency Management 17](#_Toc509294495)

[5. Technology Assumptions 18](#_Toc509294496)

[6. Best Practices and Guidelines 19](#_Toc509294497)

[7. Technology Risks and Mitigations 20](#_Toc509294498)

Index of Figures

[Figure 1. Technology Blueprint 8](#_Toc509294499)

[Figure 2. Logical Technology Architecture 11](#_Toc509294500)

[Figure 3. Indicative Deployment Architecture 12](#_Toc509294501)

[Figure 4. Build, Test and Release 12](#_Toc509294502)

[Figure 5. Token based authentication 12](#_Toc509294503)

[Figure 6. Application Physical Deployment 14](#_Toc509294504)

[Figure 7. UI Layer Architecture 14](#_Toc509294505)

[Figure 8. Custom Exception 16](#_Toc509294506)

Index of Tables

[Table 1. HCSC Preferred Technology Stack 9](#_Toc509294507)

[Table 2. Additional Third Party Library and Plug-in 9](#_Toc509294508)

[Table 3. SGR Database Objects 13](#_Toc509294509)

[Table 4. SGR Audit Database Objects 13](#_Toc509294510)

[Table 5. Log Levels to be used 15](#_Toc509294511)

Appendices

[Appendix 1. Reference Documents 21](#_Toc509294512)

# Executive Summary

## Objective

The objective of this document is to explain the System Architecture and recommend the technology choices for various layers of the HCSC SG Renewals application to help make architecturally correct design decisions.

This document will serve as guideline for development of the SG Renewals application.

## Scope

This System Architecture Document (SAD) will include the following in its scope –

* Application Architecture
* Technology, tools and libraries recommended for various application architecture swim lanes.
* Best practices for developers
* Application architecture and design ‘considerations’ with the underlying infrastructure in mind.

The below items are out of scope of this document –

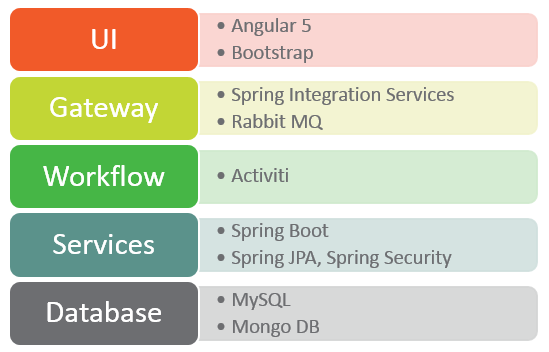
* Cloud Infrastructure configurations and related capacity planning
* Operational run book and Standard operating procedures
* Component detailed and low level design (Separate set of service design documents will be provided)
* Testing tools, strategy and methodology

## Key Considerations

* Architecture Guidelines as listed in Sec 3.1
* Build for **sustainability**. Use the latest viable technology stack.
* Architect for **scalability** with PCF features in mind.
* Use micro services architecture for **maintainability** and flexibility.
* Design for **security**. Adhere to secure coding practices handling the OWASP top 10 vulnerabilities.
* Design for **extensibility** into different languages and multi-tenancy of data (different HCSC departments, different organizations, etc)
* Build smooth **operability** into the design with good level of logging, exception handling, monitoring, etc.

# Application Tech Stack Overview

## Technology Blueprint



1. Technology Blueprint

## SG Renewals Preference of the Technology Stack

|  |  |  |
| --- | --- | --- |
| Layer | Technology | Additional Information |
| Presentation Layer | Angular 5 (Final)  HTML 5  Bootstrap | * Typescript is the recommended language for use. * Use Angular CLI to scaffold the application. * Use Angular forms, ng2-data-table. |
| Application Layer | Spring Boot, Spring Cloud (spring-cloud-config), Spring Security |  |
| Service Interface | RESTful |  |
| Database ORM Layer | Spring JPA / Hibernate |  |
| Database | MS SQL Server , MongoDB |  |
| Integration | Spring Integration Services |  |
| Security | Spring Security |  |
| Logging | SLF4j (ready or Log4j) | * SLF4J can run in no-op as well as Log4J mode. Configure two separate services so that when needed, a no-op can be deployed for production (scenarios like unexplained growth of lo files, etc) * Configure for daily log rolling as well as log rolling post 10 MB log file size. Helps in debugging. |
| Dev & CI Tools | SoapUI, Maven, Jenkins, SonarQube, Visual Source Code |  |
| Infrastructure | PCF |  |
| Operation Monitoring | JMX Metrics |  |
| JDK | Oracle JDK 1.8 & Java EE7 |  |
| Application Server | JBoss? |  |

1. HCSC Preferred Technology Stack

**Following Third Party Library or Plug-in may be used**

|  |  |  |
| --- | --- | --- |
| Technology Name | Recommended version | Purpose |
| Git | Latest | Source code repository & version control |
| Jasmine | 2.4.1 | UI unit testing framework |
| Mockito | Latest | Mock data framework |
| Nexus | Latest | Application artifacts repository |
| NodeJS | (LTS) | Prefer the LTS over the latest for compatibility. |
| NPM | 3.10.6 | Node package manager. |
| JUnit / TestNG |  | Unit Testing Framework JUnit or TestNG. |

1. Additional Third Party Library and Plug-in

# Architecture Details

## Architecture Principles and Design Guidelines

**Key Points**

1. Application is based upon micro services architecture
2. The application should be cloud native – i.e. able to leverage PCF cloud infrastructure for high availability, scalability, and performance.
3. Use only mature open source components. HCSC will review open source usage for suitability.
4. The application’s core back-end architecture should be micro-services oriented. The core domain objects will define the micro-service boundary. The services should be exposed using standard REST patterns. Use spring/spring-boot/spring-cloud projects to accomplish this. Use JPA/hibernate for ORM.
5. When integration with external services, leverage messaging and ESB patterns.
6. The application’s user interface should be modern single page application oriented with no full page refreshes.
7. Follow best practices for error and exception handling – no generic exception handling, logging details, use custom exception hierarchy when possible.
8. Follow best practices for application logging – use slf4j, no sys.out, and no sensitive info in logs.
9. All system resources (like db, message-queue, remote services) used by application should be externalized in appropriate configuration repositories (files/db) – no hardcoding of properties that vary by environments. Additionally there should be provision to re-load the configuration changes.
10. Transactions involving multiple resources or multiple services should be coordinated using distributed transactions
11. Key application health parameters (like # pending requests in memory, etc.) should be exposed as JMX metrics.
12. Application should be able to support continuous integration philosophy using right set of tools – build automation (maven), unit testing, functional testing, continuous integration (Jenkins), db changes (liquibase).
13. Follow these standards for Coding –

Java Coding

Secure Coding

Note – Java Coding standards will have be modified to include spring.

## Logical Architecture

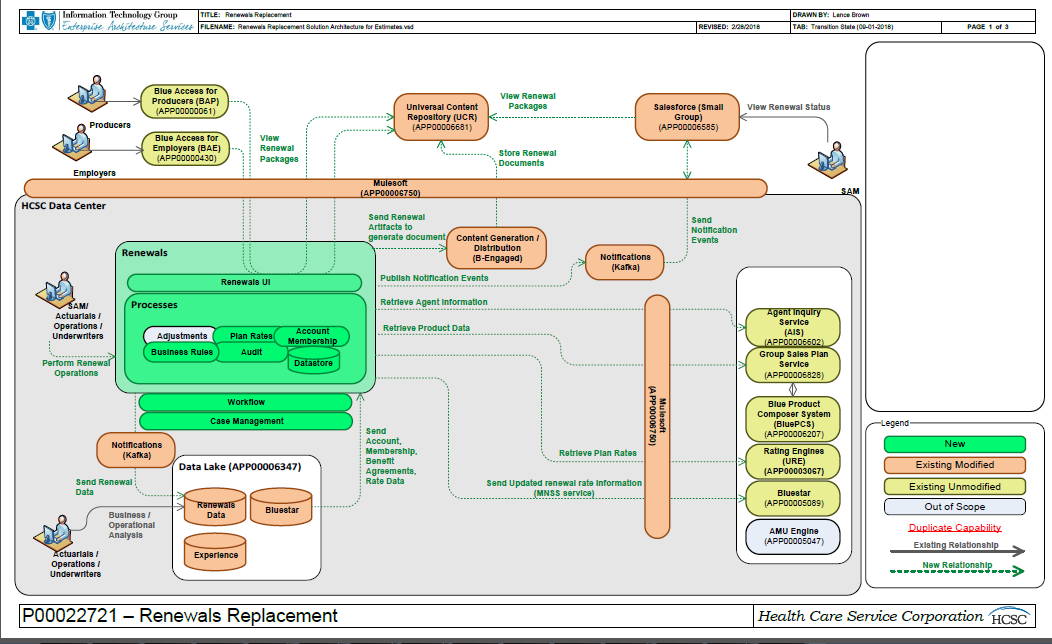


1. Logical Technology Architecture

The logical architecture for Application Container, Integration Platform and Platform is available in [Appendix- 1](#Appendix_1)

## Physical Deployment Architecture

The following figure depicts the deployment of Production and non-Production network environment for HCSC “SG Renewals Platform” in PCF.



1. Indicative Deployment Architecture

## Build Test and Release Architecture



1. Build, Test and Release

## External Services

### Enterprise Services

[TBD]

### Other external services to be used by App

[TBD]

## Integration with ERP

[TBD].

## Security

[TBD].

### Authentication and Authorization

[TBD].

1. Token based authentication

### Web Security

### 

### Other

* [TBD].

## Performance and Scalability

### Scalability

* [TBD]

### Performance

* [TBD].

### Web Server

* [TBD].

## Maintainability

* [TBD].

## Availability and Reliability

### Availability

* [TBD].

### Reliability

* [TBD].

## Usability

* [TBD].

## Integration and Deployment processes

* [TBD]

## Database

[TBD].

**DB Names:**



|  |  |  |  |
| --- | --- | --- | --- |
| Database - Servers | Environments | MSSQL DB Names | Mongo DB Names |
| **DEV-Server** | Desk1 | sprnl-rules-desk1,sprnl-Audit-desk1,sprnl-core-desk1,sprnl-wf-desk1,sprnl-Adjustments-desk1 | sprnl-data-desk1, sprnl-planrates-desk1 |
|  | Desk2 | sprnl-rules-desk2,sprnl-Audit-desk2,sprnl-core-desk2,sprnl-wf-desk2,sprnl-Adjustments-desk2 | sprnl-data-desk2, sprnl-planrates-desk2 |
|  | Dev1 | sprnl-rules-dev1,sprnl-Audit-dev1,sprnl-core-dev1,sprnl-wf-dev1,sprnl-Adjustments-dev1 | sprnl-data-dev1, sprnl-planrates-dev1 |
|  | Dev2 | sprnl-rules-dev2,sprnl-Audit-dev2,sprnl-core-dev2,sprnl-wf-dev2,sprnl-Adjustments-dev2 | sprnl-data-dev2, sprnl-planrates-dev2 |
|  | Dev3 | sprnl-rules-dev3,sprnl-Audit-dev3,sprnl-core-dev3,sprnl-wf-dev3,sprnl-Adjustments-dev3 | sprnl-data-dev3, sprnl-planrates-dev3 |
|  | Sit1 | sprnl-rules-sit1,sprnl-Audit-sit1,sprnl-core-sit1,sprnl-wf-sit1,sprnl-Adjustments-sit1 | sprnl-data-sit1, sprnl-planrates-sit1 |
|  | Sit2 | sprnl-rules-sit2,sprnl-Audit-sit2,sprnl-core-sit2,sprnl-wf-sit2,sprnl-Adjustments-sit2 | sprnl-data-sit2, sprnl-planrates-sit2 |
| **Test- Server** | UAT1 | sprnl-rules-uat1,sprnl-Audit-uat1,sprnl-core-uat1,sprnl-wf-uat1,sprnl-Adjustments-uat1 | sprnl-data-uat1, sprnl-planrates-uat1 |
|  | UAT2 | sprnl-rules-uat2,sprnl-Audit-uat2,sprnl-core-uat2,sprnl-wf-uat2,sprnl-Adjustments-uat2 | sprnl-data-uat2, sprnl-planrates-uat2 |
| **L&P- Cluster** | L&P | sprnl-rules-lp,sprnl-Audit-lp,sprnl-core-lp,sprnl-wf-lp,sprnl-Adjustments-lp | sprnl-data-lp, sprnl-planrates-lp |
| **Prod-Cluster** | PROD | sprnl-rules-prod,sprnl-Audit-prod,sprnl-core-prod,sprnl-wf-prod,sprnl-Adjustments-prod | sprnl-data-prod, sprnl-planrates-prod |

**DB Sizing:**

****

**DB Infra Structure:**

Refer to the attached document for DB related Infrastructure details.



### Database Schema

[TBD].

#### HCSC SGR Database Objects

* Please refer to attached documents in "Database Schema” section for SGR Database Objects details

1. SGR Database Objects

#### HSCS SGR Audit Database Objects

* Please refer to attached documents in “Database Schema” section for Audit Database Objects details

1. SGR Audit Database Objects

## Archival and Purging

[TBD]

# Target System

## Technical Feasibility

## Technology Stack and Technology versions

For Technology Stack please refer to [section 2.2](#_Customer_Preference_of).

## Technical Recommendations

### Prioritization

[TBD]

### Application Physical Deployment Architecture

[TBD] Indicative physical deployment architecture.

1. Application Physical Deployment

### UI Layer Architecture - Angular

[TBD]

1. UI Layer Architecture

### UI Exception Handling Design

[TBD]

### UI User Data Caching (transactional/non-transactional)

[TBD].

### Server Side and Micro services Architecture

[TBD]

High level draft version of the microservices architecture. Need to weigh in optimal solution between numbers of components/war files.

### Application Logging Architecture & Design

[TBD]

Log level will be maintained in configuration file and will have value WARN or INFO in production. All application level exceptions should be logged as ERROR or FATAL depending on criticality. Following log levels may be used in the application as per need.

|  |  |
| --- | --- |
| **Log Levels** | **Details** |
| FATAL | The FATAL level designates very severe error events that will presumably lead the application to abort. |
| ERROR | The ERROR level designates error events that might still allow the application to continue running. |
| WARN | The WARN level designates potentially harmful situations. |
| INFO | The INFO level designates informational messages that highlight the progress of the application at coarse-grained level. |
| DEBUG | The DEBUG Level designates fine-grained informational events that are most useful to debug an application. |
| TRACE | The TRACE Level designates finer-grained informational events than the DEBUG Since:1.2.12 |

1. Log Levels to be used

**Logging Pattern** –

SLF4J would act as a façade for the Log4J framework. The log4j needs to be configured such that

* Daily rolling of the log file – Use RollingAppender
* Application module/service name needs to be in the log pattern
* Date, time and log originating class name and logging level is listed – use logging pattern

**[Date] - [Log\_Level] - [Micro Service/Module] - [ClassName]-[Method Name] - [TransactionID]-[Log Message]**

* Transaction Identifier is a top level unique identifier of a transaction being worked on.

E.g. BPA: <BPA\_AUDIT\_TRAIL\_ID>, CERT:<CertificateNumber>, etc.

* Log files will use delimiter ’ (This delimiter will not be used in other services)
* Date format will use ‘/’ - i.e. YYYY/MM/DD
* Time Format will use - HH:mm:ss,SSS
* UTF-8 encoded
* File name should have module name and date.

### Application Exception Handling Architecture & Design

Application faults will be segregated into two parts.

**Custom Error:** To encapsulate and represent all services side business or validation errors. These type of errors will be shown to user as part of data validation. Error code will be passed from services to UI and UI will use configuration/properties file to retrieve error message before showing to the user. Externalization of error message to be shown to user for the error code will facilitate internationalization, through update of corresponding localized error message against the same error code.

**Custom Exception:** To encapsulate all application system exceptions. Exceptions will be normally be logged to application log files for facilitating support team to trace system error and take corrective measures.

**Application**

**Error/Exception**

**Custom Error**

(To show Business Errors to Users)

**Custom Exception**

(To track application exceptions)

1. Custom Exception

[TBD]

### Data caching options

[TBD]

### Code Quality/Testing Framework – UI

• JsHint/JsLint - CQ

• Karma (Test Runner) - UT

• Protractor - IT, dev smoke testing

• Jasmine

• PhantomJS

[TBD]

### Code Quality/Testing Framework – JAVA

• Sonar Qube

• TestNG & Mockito

• Emma

• Jmeter

[TBD]

### Application Monitoring

**Application** - Application level Custom or system generated exceptions or errors generated to log files by Log4j & Slf4j will be monitored and notifications will be sent out to support group to take necessary correction measures.

**JVM and Application Health** – JVM critical parameters will be monitored by JMX as detailed in Application Container design. JMX will be used to monitor response times of REST services.

**Technologies to be used**

* JMX Metrics

### Concurrency Management

* [TBD].

# Technology Assumptions

* 1. [TBD]
  2. All the environments can be connected from offshore. Source control and build mechanisms will also be accessible from offshore.
  3. Angular 5 final is considered for effort estimation and design. The latest and technically viable version of 5.x (final) will be considered for the development.
  4. Java version 1.8 would be used - Oracle JDK/JREs will be used and other implementations like OpenJDK will not be used.

# Best Practices and Guidelines

[TBD]

# Technology Risks and Mitigations

* + 1. Angular 5 is newly released and there might be risks of effort deviation with regards to any stability and support issue.
    2. [TBD]

1. Reference Documents
2. [TBD]
3. Environment Design:
4. Application Design:
5. Application Integration platform Design:
6. Guidelines for designing microservices:
7. The list of identified microservices and criteria for identification:
8. Java Coding standards:
9. Secure Coding standards:
10. Database Schema:
11. Security Architecture Document:

THANK YOU