|  |  |
| --- | --- |
| **Version:** | 1.0 |
| **Last Updated:** | 20-June-2016 |
| **Author:** | Gururaj Munnalli |
| **Designation:** | Software Developer |

# High Level Technical Design and System Architecture Document of RAS247

## Document Reviewers/Approvers/Contributors

The following table identifies the individuals responsible for approving, contributing or reviewing this document.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Designation** | **Reviewer/**  **Approver/ Contributor** | **Approval Date** |
| **Jitendra** |  | Approver |  |
| **Vinod** |  | Approver |  |
| **Jitendra** |  | Reviewer |  |
| **Vinod** |  | Reviewer |  |
|  |  |  |  |
|  |  |  |  |

# 1 Introduction

## Purpose

The customers in distress while on road and the service providers nearby on the same platform for a better road experience for the customer. At the same time we are a platform for the service provider to actively and instantly connect with the customer and grow their business and reach of customers. The promise of availability at any time and place is RSA’s major offering for the customers and the opportunity to increase the business for the service provider..

### The Project includes:

The service provider and customer apps would be native apps developed for android platform while the admin app would be a web app meant for the server.

Customer app includes:

* Location Capture of the customer
* Vehicles information
* Request of service and get appointment
* Request fulfill confirmation
* Bill with SMS

Service provider app includes:

* Location Capture of the Service Provider
* Fetch the customer’s location
* View the issue of the customer
* Accept or reject the request
* Provide updates to the customer about ETA
* Contact support for confirmation and closure
* Create billing info along with an SMS
* Android development
* Web app development for admin app
* UAT

Admin app (Web Application) includes:

* View customer, vehicles, services, service provider etc.
* Approve customer and service provider.

The document is divided in to various chapters

* **Introduction** - covers the purpose and scope of the document. It also outlines the various objectives of the document and the audience targeted at.
* **System Overview** - covers the vision and mission of the project, technology and business drivers and also some brief overview of the candidate system under the purview of design.
* **Design Decisions** - covers the various Design decisions and Architectural and De-sign patterns used.
* **Rest of the Chapters** - covers the various architectural views and viewpoints.
* **Technology Stack** - captures various standards complied to, various frameworks and components used and productivity tools utilized for rapid development.

## Objectives

To capture the overall technical direction and evolution of the RSA system’s architecture

## Scope

This document captures high level design and system architecture to support Project requirements and objectives. This is high level design document only and will discuss approach at high level. The details mentioned are for the purpose of doing detail implementation effort estimation and planning. Low level details have to be worked while in Design and Implementation phase.

## Target Audience

The intended audiences of this document are **Uniqueware** Consulting

* Architects
* Development team
* Project management team(s)
* Support and maintenance team(s)
* Business stakeholders
* Technical stakeholders

## Assumptions

System design is created with following assumptions, should any of these change it will have impact and will be handled via proper project management change request.

* Customer provide service request with his/her information in detail.
* Customer provides the information about family members depend on RSA247 package.

# 2 System Overview

## Objective and Vision

Web enterprise application to address following high level use cases –

Android app to address following high level use cases –

## Business Drivers

The purpose of this project

## Technology Drivers

Design and build Android app and enterprise application

## Risks, Constraints & Assumptions

Architectural Risks

|  |  |  |
| --- | --- | --- |
| # | Risk | Implication and Mitigation Plan |
| 1 |  |  |
| 2 |  |  |

Constraints and Impacts

|  |  |  |
| --- | --- | --- |
| # | Constraints | Impact |
| 1 |  |  |
| 2 |  |  |

# 3 Design Decisions, Architectural & Design Patterns

## Design Decisions

|  |  |  |
| --- | --- | --- |
| # | Design Decision | Rationale |
| 1 | The system will be Java Enterprise application | JEE provides robust framework to build enterprise application |
| 2 | System will use Spring | Out-of-box components for rapid development AOP(Aspect based programing) to address Cross cutting concerns. DI(Dependency injection) |
| 3 | System will use MySql | Persistence details |
| 4 | System will use JBoss | JBoss is robust Enterprise application server |
| 5 | Android app | Android app for end users |

## Key Architecture Patterns

|  |  |  |
| --- | --- | --- |
| # | Architecture Patterns | Notes |
| 1 | Multi-Tier Architecture | The application processing, and the data management are logically separate processes. |
| 2 | Multi Layering | Multilayered software architecture is using different layers for allocating the responsibilities of an application. |
| 3 | Singleton | The singleton pattern is a design pattern used to implement the mathematical concept of a singleton, by restricting the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system. The concept is sometimes generalized to systems that operate more efficiently when only one object exists, or that restrict the instantiation to a certain number of objects. |
| 4 | Dependency Injection | Spring will be used to Object creation and lifecycle |
| 5 | AOP | Aspect-Oriented Programming (AOP) complements Object-Oriented Programming (OOP) by providing another way of thinking about program structure. The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. Aspects enable the modularization of concerns such as transaction management that cut across multiple types and objects. (Such concerns are often termed crosscutting concerns in AOP literature.) |
| 6 | IoC(Inversion of Control) | Decoupling of the execution of a certain task from implementation. Every system can focus on what it is designed for. The systems make no assumptions about what other systems do or should do. Replacing systems will have no side effect on other systems. |
| 7 | Factory Pattern | BeanFactory for creating instance of an object . |

# 4 Requirements View

## Functional Requirements

## Use Case Diagram



Customer

Service Partner

Administrator

## Customer point view



Customer

## Actors

System will have following users interacting with system

* Customer
* Service Partner
* Administrator

## Business Flow

### Non Functional Requirements

The following table summarizes various non-functional requirements that the system should adhere to

# 5 Conceptual Architecture

**Client Tier**

**External Inerface**

**Browses**

**External**

**Internal Users**

**External Service Interface**

**Presentation Tier**

**Interface Layer**

**Presentation**

**Managed Bean**

**Delegates**

**Business Tier**

**Persistence Tier**

**Business Layer**

**Service API**

**Business Logic**

**DAO**

**Entity**

**Java Beans**

The diagram above shows the conceptual Architecture of RSA247 application.

System can be accessed through following client

- Internal Users

- External systems

System has been divided in different Tiers so that can be physically managed

## Business Tier

Business tier is responsible for handling business processing, Interacting with database and providing services in the form of REST resources. External system and Presentation will communicate with business tier using xml’s. Business services will be REST services.

## Presentation Tier

Presentation tier is responsible to interact with client tier (User). Any business processing needs will be served through business tier. Presentation tier will also be responsible for security, User authentication and authorization.

## Client Tier

Client tier is responsible for rendering user interface where user can interact.

## Persistence Tier

Persistence tier will be responsible to store transactional data for RSA247 code. Meta data for packages and pricing, services and pricing and user will be stored in persistence data base. Transactional data will be persisted in Data base. DAO will be responsible for persistence and perform CRUD operation.

# 6 Logical & Technical Architecture

**Client Tier**

**External Interface**

**Presentation Tier**

**Business Tier**

**Data Base**

**Browsers**

**External Systems**

**Presentation Layer**

**Business Layer**

**Internal Users**

**Interface**

**Presentation**

**Managed Bean**

**Delegates**

**Logging**

**Security**

**Cache**

**Bean Mapper**

**Exception**

**Service API**

**Business Logic**

**DAO**

**Entity**

**Java Beans**

Rest/Xml

Rest/Xml

Java API

JDBC

RSA247 application is conceived to be a multi-tiered, multi-layered Enterprise System.

As depicted, the system consists of three layers Presentation layer, the Business layer and Persistence layer. Each layer performs speciﬁc functionalities.

The various modes of communication protocols/medium/data that includes HTTP, JDBC, JSON, HTML.

## Presentation Layer

This layer is responsible for

* Security
* Presentation
* Request handling
* Marshaling/Un-Marshaling
* Communication with Services

## Business Layer

This layer is responsible for

* Business processing
* Data access Object
* Data persistence

This layer consists of Business Logic components that will be utilized by various delegator APIs to perform or trigger business functionalities. This layer will also consist of utility and helper class that help in performing business transactions.

Service API’s will be responsible for service request processing, security, content negotiation.

## Common components Layer:

This layer consist all the relevant commonly used components across the business tier. Logging, Security, Bean mapper and other frequently used components. This will be developed as pluggable archive which can be plugged in with presentation layer or business layer.

# 7 Deployment view

# 8 Technology Stack

## Standards

The system follows various open standards as listed.

* HTTP
* XML
* JSON
* HTML
* JDBC
* JAVA EE Specifications
* Hibernate
* Restful Services
* Spring
* Tomcat

### Open Source Components and Frameworks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Tools | Provider | Version | License | Source |
| 1 | JEE specification components | Sun/Oracle | 7/8 | Sun/Oracle |  |
| 2 | Log4j | Apache | 1.x | Apache |  |
| 3 | Maven | Apache | 3.x | Apache 3 |  |
| 4 | JDBC | Sun | 4.x | Sun |  |
| 5 | Spring | SpringSource | 3.x | SpringSource |  |
| 6 | JBoss | JBoss | 7.x | JBoss |  |
| 7 | SQLServer | SQLServer |  | SQLServer |  |
| 8 | Hibernate | Hibernate | 4.x | JBoss |  |
| 9 | JAXB | Sun | 2.x | Sun |  |

### Productivity Tools

|  |  |  |  |
| --- | --- | --- | --- |
| # | Tools | Rational | Notes |
| 1 | Eclipse | Popularly and widely used JEE development environment | Rapid Development. Out of the box code generators. Easy to use Assists programmers with quick and easy to use tools. Pluggable utilities |
| 2 | Maven | Integrated and continuous build system for Java | Easy to identify and centralize dependent artifacts. Team builds. Integration with SVN. Automated reports for style and static code analysis SQL Developer Widely used data base client Easy to use and debug |
| 3 | Findbugs | Static code analyzer | Easy plugins to integrate with eclipse and maven |
| 4 | Checkstyle | Style check for source code | Easy plugins to integrate with eclipse and maven |
| 5 | SVN plugin for eclipse | Integrated and continuous source control | Easy plugins to integrate with eclipse and maven |

# 9 Appendix

|  |  |  |
| --- | --- | --- |
| # | Term | Definition |
| 1 | JEE | Java Enterprise Edition |
| 2 | REST | Representational State Transfer |