Architecture Vision

<Project>

Version 0.1

For

<CLIENT>

Contents

[1 Introduction 6](#_Toc509931823)

[1.1 Purpose 6](#_Toc509931824)

[1.2 Scope 6](#_Toc509931825)

[1.3 Definitions 6](#_Toc509931826)

[1.4 References 7](#_Toc509931827)

[1.5 Documentation Roadmap 7](#_Toc509931828)

[2 Executive Summary 9](#_Toc509931829)

[2.1 Key Decisions 9](#_Toc509931830)

[2.2 Key Risks and Open Issues 9](#_Toc509931831)

[3 Architectural Drivers 10](#_Toc509931832)

[3.1 Business Case 10](#_Toc509931833)

[**3.1.1** Business Goals 11](#_Toc509931834)

[**3.1.2** Major Features 11](#_Toc509931835)

[3.2 Service Level Agreement 11](#_Toc509931836)

[**3.2.1** Scope and Applicability 12](#_Toc509931837)

[**3.2.2** Service Quality Guarantees 12](#_Toc509931838)

[**3.2.3** Definitions 12](#_Toc509931839)

[3.3 Use Case Model 12](#_Toc509931840)

[**3.3.1** Use Case View <View Name> 12](#_Toc509931841)

[3.4 Domain Model 14](#_Toc509931842)

[**3.4.1** Domain View <View Name> 14](#_Toc509931843)

[3.5 Design Constraints 15](#_Toc509931844)

[3.6 Quality Attribute Scenarios 15](#_Toc509931845)

[4 Solution Architecture 17](#_Toc509931846)

[4.1 Big Picture 17](#_Toc509931847)

[**4.1.1** Solution Context 17](#_Toc509931848)

[**4.1.2** Solution Decomposition 19](#_Toc509931849)

[**4.1.3** Layered Application Structure 21](#_Toc509931850)

[4.2 Development Technology Stack 23](#_Toc509931851)

[**4.2.1** Development Languages, Frameworks, and Libraries 23](#_Toc509931852)

[**4.2.2** Development Tools 24](#_Toc509931853)

[**4.2.3** External Integration Points 25](#_Toc509931854)

[4.3 Architecture Part X 26](#_Toc509931855)

[**4.3.1** Decision View <View Name> 26](#_Toc509931856)

[5 Operation Plan 28](#_Toc509931857)

[5.1 Transition Phase 28](#_Toc509931858)

[**5.1.1** Infrastructure 28](#_Toc509931859)

[**5.1.2** Environments 29](#_Toc509931860)

[**5.1.3** Provisioning 29](#_Toc509931861)

[**5.1.4** Management 29](#_Toc509931862)

[**5.1.5** Monitoring 30](#_Toc509931863)

[5.2 Operation Phase 30](#_Toc509931864)

[**5.2.1** Service Operation Support 31](#_Toc509931865)

[**5.2.2** CMOD Activities 31](#_Toc509931866)

[**5.2.3** Continuous Improvement Process 31](#_Toc509931867)

[6 Implementation Roadmap 32](#_Toc509931868)

[6.1 Implementation Deliverables 32](#_Toc509931869)

[6.2 Implementation Milestones 32](#_Toc509931870)

[6.3 Estimate 33](#_Toc509931871)

[**6.3.1** Assumptions and Limitations 33](#_Toc509931872)

[**6.3.2** Estimate 33](#_Toc509931873)

[6.4 Team 33](#_Toc509931874)

[**6.4.1** Team Skillset 34](#_Toc509931875)

[**6.4.2** Team Structure 34](#_Toc509931876)

[7 Appendix A – Cross-Reference 36](#_Toc509931877)

[7.1 Driver Fulfillment Table 36](#_Toc509931878)

[7.2 View Reference Table 36](#_Toc509931879)

[7.3 Trade-off Reference Table 37](#_Toc509931880)

[7.4 Element Catalog 37](#_Toc509931881)

[8 Appendix B – Architecture Design Methodology 38](#_Toc509931882)

[9 Appendix C – How View is Documented 40](#_Toc509931883)

[9.1 View Structure 40](#_Toc509931884)

[9.2 Graphical Notation 44](#_Toc509931885)

[10 Appendix D – Estimation Methodology 46](#_Toc509931886)

[11 Guide to Architecture Vision Documentation – DELETE ME! 47](#_Toc509931887)

[11.1 Documentation Guidelines 47](#_Toc509931888)

[11.2 Architecture Vision Effort Estimation 52](#_Toc509931889)

[11.3 Styles to Use in the Document 53](#_Toc509931890)

[**11.3.1** Custom Section 55](#_Toc509931891)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
|  | <x.x> | <details> | <name> |
|  | <x.x> | Approved | <name> |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This Architecture Vision elicits the significant architecture drivers such as business, functional, nonfunctional requirements and constraints, defines architecture, and develops a roadmap for Single Entry implementation. The document is intended as a primary technical guidance into solution implementation for the development team.

The solution architecture is designed following the guiding principles outlined in [Appendix A – Architecture Design Methodology](#_Appendix_A_–).

## Scope

The document describes the proposed Single Entry architecture towards development of the solution that will satisfy business, functional, non-functional requirements and constraints provided by the Client. This Architecture Vision covers the following information:

* Significant architectural drivers for the Solution
* Proposed software architecture of the solution based on these drivers
* Technology stack and environment definitions
* Operations specific perspectives
* Development road map and high level estimates for effort, team size and skill sets.

## Definitions

The Definitions section lists the acronyms and terms used in this document which might possess lesser familiarity or double meaning to the reader.

|  |  |  |
| --- | --- | --- |
| # | Term | Definition |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## References

The References section provides a complete list of all the documents referenced elsewhere in this document.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Version | Date | Document Name | Published by |
|  | <x.x> | <mm/dd/yyyy> | <Document Name> | <Publishing Organization> |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Documentation Roadmap

This section guides into the Architecture Vision document structure to help find the information of interest.

* [Executive Summary](#_Executive_Summary) speaks about the proposed architecture on the highest level and covers:
  + [Key Decisions](#_Key_Decisions) about the architectural and operational choices made for the solution
  + [Key Risks and Open Issues](#_Key_Risks_and) detected with the analysis of the available context, requirements, and proposed decisions
* [Architectural Drivers](#_Architectural_Drivers) elicits the known essential requirements and constraints for the solution to design which play key role in forming archectural decisions and overall architecture.
  + [Business Case](#_Business_Case) describes the solution from the business standpoint including major features
  + [Service Level Agreement](#_Service_Level_Agreement) defines the important technical constraints and guarantees under which the solution will be serviced to its clients
  + [Use Case Model](#_Use_Case_Model) lists the key Use Cases for the designed solution
  + [Domain Model](#_Domain_Model) shows the key business entities with attributes and relationships between them.
  + [Design Constraints](#_Design_Constraints) include business, resource, technical and other constraints accounted for in the architecture of the solution.
  + [Quality Attribute Scenarios](#_Quality_Attribute_Scenarios) are a set of the testable scenarios clarifying non-functional requirements for the system quality attributes such as performance, maintainability, and others.
* [Solution Architecture](#_Solution_Architecture) defines the proposed architecture as a set of architectural views in the format defined in [Appendix C – How View is Documented](#_Appendix_B_–).
  + [Big Picture](#_Big_Picture) shows the solution architectural context, high level decomposition into components, and followed reference architecture.
  + [Development Technology Stack](#_Development_Technology_Stack) selects the tools, frameworks, libraries, external services and other technologies the solution implementation will rely on.
  + <Insert other sections of level 2>
* [Operation Plan](#_Operation_Plan) defines the environments, structures, toolsets, processes, and activites essential to build and operate the solution implementation.
  + [Transition Phase](#_Transition_Phase) talks about the part of the Operation Plan related to the solution implementation phase, including design and development of solution operation framework.
  + [Operation Phase](#_Operation_Phase) defines the processes to operate the solution in production using the framework built at the Transition Phase.
* [Implementation Roadmap](#_Implementation_Roadmap) proposes the plan for solution implementation including:
  + [Implementation Deliverables](#_Implementation_Deliverables) expected to be implemented and delivered at the implementation phase.
  + [Implementation Milestones](#_Implementation_Milestones) tied to the project timeline
  + [Estimate](#_Estimate_1) of complexity/size, effort, schedules for the implementation
  + [Team](#_Team) skillset and structure based on the technology, competence, and schedule requirements
* [Appendix A – Cross-Reference](#_Appendix_A_–_1) defines mappings between views, stakeholders, drivers, elements, etc.
* [Appendix B – Architecture Design Methodology](#_Appendix_A_–) briefly describes the used architecture design methodology.
* [Appendix C – How View is Documented](#_Appendix_B_–) explains the semantics and structure of an architectural view documentation
* [Appendix D – Estimation Methodology](#_Appendix_D_–) briefly describes the used estimation methodology.

# Executive Summary

The section Executive Summary highlights key architectural decisions made for the solution described in [Business Case](#_Business_Case). These decisions are defined and discussed in depth in [Solution Architecture](#_Solution_Architecture) while [Implementation Roadmap](#_Implementation_Roadmap) lays out the proposed milestones, estimates, and team to implement the provided architecture vision.

This section also summarizes the key business and technical risks related to the solution implementation. These risks are uncovered in depth in the rest of the document.

## Key Decisions

The section outlines key design decisions about the solution including the architecture big picture and most essential technologies and external services to rely on.

The solution will be implemented as a cloud based Java web service and deployed to the Cloud Platform Provider XYZ using it’s A, B, C services. The structured data will be stored in the scalable cloud RDBMS storage provided by the Cloud Platform Provider with multy-AZ replication to ensure data availability and backup as required by the solution’s SLA. The solution will be integrated as a REST-ful client with the third-party service ABC to store and retrieve the unstructured blob data.

## Key Risks and Open Issues

The section lists the key risks related to the solution design and implementation. It also lists key open issues where architectural decisions have not been made yet or are likely to change.

|  |  |
| --- | --- |
| Risk Description | Mitigation Strategy |
| Unauthorized data access | Employ robust IAM (Identity and Access Management) solutions and regular security audits. |
| Data leakage | Encrypt all sensitive information, both in transit and at rest. Use secure protocols for all data transmissions. |
| Downtime during high traffic (e.g., tax season) | Implement auto-scaling, load balancing, and have a fault-tolerance strategy, perhaps through the use of AWS SQS. |
| Slow response time during recalculation | Use caching solutions like Redis and optimize the tax calculation engine for performance. |
| Inaccurate tax calculations | Extensive unit testing and QA. Regular updates to the rule engine based on new or updated tax laws. |
| Complexity in updating tax rules | Use a well-documented rule engine like Drools, and provide sufficient training to administrators. |
| Data corruption | Implement ACID transactions in the database layer. Regular backups and integrity checks. |
| Inadequate audit trail | Integrate comprehensive logging and monitoring solutions like ELK Stack or AWS CloudWatch. |
| Vendor lock-in | Use containerization and microservices to ensure that services can be moved to another provider if necessary. |
| Data loss due to system failures | Regular data backups and implement disaster recovery strategies. |

# Architectural Drivers

The section captures significant requirements driving the solution architecture and road map. The requirements which are not influencing the solution architecture in major ways and low level requirement details and scenarios are typically excluded from this section and can be found in the requirement specification or the product backlogs.

## Business Case

The section lays out the business case for the solution <Project>.



Figure . Business level view of <Project>

The envisioned solution will enable the users to manage their payment transactions from their mobile devices whily allowing access to the rich reporting from the desktop browsers and backing up their transaction logs in the form of the pdf files to their accounts open with the third-party cloud storage services such as Dropbox for later access. The solution will be deployed on the Amazon cloud as SaaS web application.

### Business Goals

The section enumerates essential business goals for the solution

|  |  |
| --- | --- |
| # | Description |
| BG-1 | Modernize the company’s current approach of interfacing with our existing users |
| BG-2 | Achieve competitive advantage in TCO by moving to the Cloud based SaaS model |
| BG-3 | Enable access to the system from wider range of devices to target larger user base |
| <goal id> | <business goal description> |

### Major Features

The section enumerates solution major features.

|  |  |
| --- | --- |
| # | Description |
| F-1 | Centralized transaction data storage and management on the cloud |
| F-2 | Seamless integration with third-party cloud storage providers |
| F-3 | Access from the mobile devices and desktop browsers without loss of quality in user experience |
| F-4 | Secure data access, transmission, and storage protected from the unauthorized access |
| F-5 | Service uptime not less than 99.9% |
| <feature id> | <feature description> |

## Service Level Agreement

The SLA section provides key parts of the Service Level Agreement applicable to the designed solution ato be supported by the proposed solution architecture.

### Scope and Applicability

This Service Level Agreement (“SLA”) establishes the service quality parameters that are to be applied to the use of <Project> services, and is part of the <Project> Services Customer Agreement.

The terms and conditions specified in this agreement apply solely to the services provided as callable API hosted by the company, herein called “Covered Services.” This SLA applies separately to each service consumer (“Consumer”) that is using the Covered Services. <CLIENT> reserves the right to change the terms of this SLA in accordance with the <Project> Services Customer Agreement at any time.

### Service Quality Guarantees

The Covered Services will be operational and available to Consumers at least 99.95% of the time in any calendar month. If <CLIENT> does not meet this SLA requirement while the Consumer succeeds in meeting its SLA obligations, the Consumer will be eligible to receive Financial Credits as compensation. This SLA states the Consumer’s exclusive right to compensation for any failure on <CLIENT>’s part to fulfill the SLA requirements.

### Definitions

The following definitions are to be applied to <CLIENT>’s SLA:

* “Unavailability” is defined as the entirety of the Consumer’s running instances as having no external connectivity for a duration that is at least five consecutive minutes in length, during which the Consumer is unable to launch commands against the remote API through either the Web application or Web service API.
* “Downtime Period” is defined as a period of five or more consecutive minutes of the service remaining in a state of Unavailability. Periods of “Intermittent Downtime” that are less than five minutes long do not count towards Downtime Periods.

## Use Case Model

### Use Case View <View Name>

#### View Context

The interaction outlined in the feature [F-3](#Feature_F_3) happens between the user and the user facing we application

#### Representation



Figure 2. <Use Case View Name>

This view defines primary Use Cases for the feature [F-3](#Feature_F_3), corresponding to the types of interaction between the user and the web application supported by the solution.

#### Element Catalog

|  |  |  |
| --- | --- | --- |
| # | Name | Description |
| ACT-1 | User | A business user of the system |
| UC-1 | Use Case1 | The user logs into the system. |
| UC-2 | Use Case2 | The user pays with the registered credit card. |
| <id> | <element name> | <Use Case description or scenario> |

## Domain Model

The section captures the solution domain model parts essential for the solution architecture including entities, their attributes, and the relationships between them. These elements are captured as part of the solution requirement analysis and do not directly map to the modules, class files, or other similar elements on the implementation level.

### Domain View <View Name>

#### View Context

User facing functionality involves the entities and relationships captured by this view.

#### Representation



Figure 3. <Domain View Name>

This view addresses the primary entities participating in interaction defined by the Use Cases [UC-1](#UseCase_UC_1) and [UC-2](#UseCase_UC_2). The important concern of is-a relationship between Class1 and Class3 is addressed with the logical extension on the view.

#### Element Catalog

The catalog defines the business entities and relationships included in the representation of this Domain View.

|  |  |
| --- | --- |
| # | Description |
| Class1 | Responsible for a, b, c |
| Class2 | Responsible for a, b, c |
| Class3 | Responsible for a, b, c |
| <entity name> | <entity description> |

## Design Constraints

The section lists the constraints accounted for in the designed solution. These can be of business, technical, resource, and other types.

|  |  |
| --- | --- |
| # | Description |
|  | A minimum of 1000 simultanious users must be supported |
|  | Time to market must be within one year |
|  | Java+RDBMS on Amazon technology stack must be used |
|  | The existing client’s dev team skill set is based on Spring framework and JPA |
| <constraint id> | <constraint description> |

## Quality Attribute Scenarios

A Quality Attribute Scenario is an unambiguous and testable requirement for one or more Solution Quality Attributes such as Performance, Usability, Maintainability and others. The scenario consists of six parts: Source of Stimulus, Stimulus, Environment, Artifact, Response, testable and accurate Response Measure.

This section lists and prioritizes the scenarios pertinent to the designed solution.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Quality Attribute | Scenario | Business Priority | Related  To |
|  | Security  (credentials transport security) | At all times the credentials entered by the user during log-in are transferred to the server over encrypted, secure channel without the chance of sniffing by third party. | High | [UC-1](#UseCase_UC_1) |
|  | Usability  (easiness of payment) | When logged in and navigated to the payment page it takes the user up to 3 clicks to pay with the preregistered valid credit card | Medium | [UC-2](#UseCase_UC_2) |
| <scenario id> | <attribute name  (scenario meaning)> | <quality attribute scenario description> | <priority level: High, Medium, Low> | <use case, feature, constraint ids> |

Or alternative notation:

**Scenario:** QA-1

**Quality Attributes:** Security

**Business Priority:** High

**Related To:** [UC-1](#UseCase_UC_1)

**Description:** At all times the credentials entered by the user during log-in are transferred to the server over encrypted, secure channel without the chance of sniffing by third party.

**Environment:** Normal operation conditions

Source

Login Form

Artifact

Client-Server Channel

Response Measure

Traffic encrypted and protected from sniffing

Stimulus

Response

Enter and submit credentials

Credentials transferred to server

**Environment**: The system is online and accessible from outside

# Solution Architecture

The section Solution Architecture is primary for the Architecture Vision document. It defines and reasons about the solution architecture design based on the architecturally significant requirements and constraints identified in the section [Architectural Drivers](#_Architectural_Drivers).

## Big Picture

The section includes a list of architectural views covering the designed solution along with the context it runs in on the high level.

### Solution Context

#### Intent

The view defines the primary solution components collaborating with the external systems and services. It is driven by the [Business Case](#_Business_Case).

#### Context

Context of the decision as diagram or text

#### Representation



Figure 4. Solution Context

Diagram and text documenting decision

#### Element Catalog

Table of annotated elements

|  |  |
| --- | --- |
| Name | Description |
| Cloud Based Solution | Responsible for implementing the REST API to serve data on request from the mobile and web clients |
| Element2 | Responsible for a, b, c |
| Element3 | Responsible for a, b, c |
| <element name> | <element description and responsibilities> |

#### Behavior

Behavior diagrams and scenarios

#### Variability

Points of decision’s configuration, customization, etc.

#### Reasoning

Reasoning

### Solution Decomposition

#### Intent

The view defines the runtime decomposition of the server-side part of the solution. It is driven by the [Business Case](#_Business_Case) and the architecture best practices applicable to the cloud-based applications.

#### Context

The view context is defined by the view [Solution Context](#_Solution_Context) where this section represents decomposition of [Cloud Based Solution component](#Element_Cloud_Based_Solution).

#### Representation



Figure 5. Cloud Solution Decomposition

The cloud based part of the solution is decomposed into several parts documented below combining several standard architectural patterns applicable to the highly loaded cloud based SaaS applications: Load Balancer, Data Cache, Background Process, Shared Storage, and Static Content Provider. The subsection resoning provides detailed discussion of these choices.

#### Element Catalog

Table of annotated elements

|  |  |
| --- | --- |
| # | Description |
| Element1 | Responsible for a, b, c |
| Element2 | Responsible for a, b, c |
| Element3 | Responsible for a, b, c |
| <element name> | <element description and responsibilities> |

#### Behavior

Behavior diagrams and scenarios

#### Variability

Points of decision’s configuration, customization, etc.

#### Reasoning

Reasoning

### Layered Application Structure

#### Intent

The view defines the runtime decomposition of the server-side part of the solution. It is driven by the [Business Case](#_Business_Case) and the architecture best practices applicable to the cloud-based applications.

#### Context

The view context is defined by the view [Solution Context](#_Solution_Context) where this section represents decomposition of [Cloud Based Solution component](#Element_Cloud_Based_Solution).

#### Representation



Figure 6. Layered Application Structure

The cloud based part of the solution is decomposed into several parts documented below combining several standard architectural patterns applicable to the highly loaded cloud based SaaS applications: Load Balancer, Data Cache, Background Process, Shared Storage, and Static Content Provider. The subsection resoning provides detailed discussion of these choices.

#### Element Catalog

Table of annotated elements

|  |  |
| --- | --- |
| # | Description |
| Element1 | Responsible for a, b, c |
| Element2 | Responsible for a, b, c |
| Element3 | Responsible for a, b, c |
| <element name> | <element description and responsibilities> |

#### Behavior

Behavior diagrams and scenarios

#### Variability

Points of decision’s configuration, customization, etc.

#### Reasoning

Text

## Development Technology Stack

The section includes a list of architectural views covering <List of Concerns, Solution Part, etc.>.

### Development Languages, Frameworks, and Libraries

#### Intent

The view lists the set of programming languages, frameworks, and libraries the solution implementation will depend on.

#### Context

The context is provided by the view [Solution Context](#_Solution_Context).

#### Element Catalog

Table of annotated elements

|  |  |  |
| --- | --- | --- |
| Name | Version | Description |
| Framework1 | x.x | Responsible for a, b, c |
| Library2 | 3.0-RC1 | Responsible for a, b, c |
| Library3 | x.x.x.x | Responsible for a, b, c |
| <name> | <version> | <description and responsibilities> |

#### Variability

Points of decision’s configuration, customization, etc.

#### Reasoning

Text

### Development Tools

#### Intent

The view lists the set of tools the development team will rely upon in solution implementation.

#### Context

The context is provided by the view [Solution Context](#_Solution_Context).

#### Element Catalog

Table of annotated elements

|  |  |  |
| --- | --- | --- |
| Name | Version | Description |
| Framework1 | x.x | Responsible for a, b, c |
| Library2 | 3.0-RC1 | Responsible for a, b, c |
| Library3 | x.x.x.x | Responsible for a, b, c |
| <name> | <version> | <description and responsibilities> |

#### Variability

Points of decision’s configuration, customization, etc.

#### Reasoning

Text

### External Integration Points

#### Intent

The view lists the set of programming languages, frameworks, and libraries the solution implementation will depend on.

#### Context

The context is provided by the view [Solution Context](#_Solution_Context).

#### Element Catalog

Table of annotated elements

|  |  |  |
| --- | --- | --- |
| Name | Version | Description |
| Framework1 | x.x | Responsible for a, b, c |
| Library2 | 3.0-RC1 | Responsible for a, b, c |
| Library3 | x.x.x.x | Responsible for a, b, c |
| <name> | <version> | <description and responsibilities> |

#### Variability

Points of decision’s configuration, customization, etc.

#### Reasoning

Text

## Architecture Part X

The section includes a list of architectural views covering <List of Concerns, Solution Part, etc.>.

### Decision View <View Name>

#### Intent

Use Cases, QAS, etc

#### Context

Context of the decision as diagram or text

#### Representation

Diagram and text documenting decision

#### Element Catalog

Table of annotated elements

|  |  |
| --- | --- |
| # | Description |
| Element1 | Responsible for a, b, c |
| Element2 | Responsible for a, b, c |
| Element3 | Responsible for a, b, c |
| <element name> | <element description and responsibilities> |

#### Interfaces

Element interfaces

#### Behavior

Behavior diagrams and scenarios

#### Variability

Points of decision’s configuration, customization, etc.

#### Reasoning

Reasoning

# Operation Plan

Typically, each service has two key phases of its lifecycle: Transition Phase and Operation Phase.



The state while the solution stays in active development mode is called **Transition Phase**. During this period main activities related to Infrastructure management will be performed by system engineering team, especially by Configuration Managers (CMs). Their goal is to implement all technical solutions to make the product ready to enter the Operation Phase.

Transition Phase ends up with the Production Release.

Operations Team will be responsible for performing the activities during the **Operation Phase**. Key goal at this phase is to maintain the desired quality of service and Service Level Agreement.

## Transition Phase

The section includes a list of architectural views covering <List of Concerns, Solution Part, etc.>.

### Infrastructure

#### Hosting Platform

Text

#### Hardware Resources

Text

#### Virtualized Resources

Text

### Environments

Text

#### Development Environment

Text

#### Continuous Integration Environment

Text

#### Testing Environment

Text

#### Staging Environment

Text

#### Production Environment

Text

### Provisioning

#### Packaging

Text

#### Deployment

Text

#### Undeployment

Text

### Management

#### Application Management

Text

#### Data Management

Text

#### Infrastructure Management

Text

#### System Security Management

Text

#### Backup and Restore

Text

#### Disaster Recovery

Text

#### Incident Management

Text

### Monitoring

#### Availability and Capacity Monitoring

Text

#### Logging and Audit

Text

#### Performance Monitoring

Text

#### Health Monitoring

Text

## Operation Phase

All activities during Operation Phase are performed in cycle.

Some of the Transition Phase tasks can be transferred to Operation Phase if they are considered as not vitally important to start Service Operation Phase.

### Service Operation Support

Text

### CMOD Activities

Text

### Continuous Improvement Process

Text

# Implementation Roadmap

Implementation Roadmap defines the solution implementation road map including list of implementation deliverables, major milestones, effort estimates, and recommended team skillset, structure, and size.

## Implementation Deliverables

The implemented solution will include the following parts as deliverables:

|  |  |  |
| --- | --- | --- |
| Name | Refer to Requirements | Refer to Design |
| Application 1 |  |  |
| Service 2 |  |  |
| Mobile Client 3 |  |  |
| Production Deployment 4 |  |  |

## Implementation Milestones

The section proposes the major milestones to guide the solution implementation.

|  |  |  |
| --- | --- | --- |
| Milestone | Description | Outcomes |
| M0 | Project bootstrap | * Dev Team ramped up * Dev and Testing Environment set up * Project Skeleton built |
| M1 | Main application with web front-end development | * Alpha version of the web application * Deployment to the staging |
| M2 | Development and stabilization of the web application, development of the deployment and monitoring procedures, release | * Release 1.0 of the application prepared and stable * DevOps framework functional * Release 1.0 goes to production |

## Estimate

The Estimate section provides the estimates for the solution implementation based on the proposed architecture and selected estimation methodology.

### Assumptions and Limitations

This section describes the known limitations imposed by selected platform, technology, hardware, operating system, third party components, etc. which might affect implementation effort and schedule estimates.

|  |  |  |
| --- | --- | --- |
| # | Description | Responsible |
| A1 | Assumption 1 | <Client Name> |
| A2 | Assumption 2 | SoftServe Team |
| L1 | Limitation 1 | <Third Party Name> |

### Estimate

The Estimate section provides effort and, optionally, schedule estimates for the implementation based on the architecture proposed in this Vision document and the selected estimation methodology defined in [Appendix C](#_Appendix_C_–).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Item | Complexity | Min Effort | Max Effort |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | **Total:** |  |  |  |

## Team

The Team section outlines the proposed team skillset and structure based on the proposed technology stack and estimates.

### Team Skillset

The section defines skills required from the different team member roles to implement the solution.

|  |  |
| --- | --- |
| Role | Skillset |
| Business Analyst | * Business domain 1 * Business domain 2 |
| Backend Developer | * Programming Language 1 * Technology 2 * Framework 3 * Tool 4 * Standard 5 |
| UI Developer | * Programming Language 1 * Technology 2 * Framework 3 * Tool 4 * Standard 5 |
| DevOps Engineer | * Programming Language 1 * Technology 2 * Framework 3 * Tool 4 * Standard 5 |

### Team Structure

The section proposes a team/sub-team structure and work allocation for the implementation phase.

#### Core Team

The core team is responsible for:

* Development of <Component 1>
* Development of <Component 2>
* Support of <Application 1>

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Responsibility | Count | FTE |
| Project Manager | * Project management (coordinate the team, status reporting, communications with the Client team) * Gap analysis | 1 | 0.5 |
| Business Analyst | * Business analysis * Business requirements specification * Software requirements specification | 1 | 1.0 |
| Solutions Architect | * System analysis and design * System requirements specification | 1 | 0.2 |
| Technical Leader | * Technical leadership and communication * SCRUM Master * Code Reviews * Backend implementation | 1 | 1.0 |
| Sr. Backend Developer | * Technical communication * Code reviews * Backend implementation | 2 | 1.0 |
| Jr. Backend Developer | * Backend implementation | 3 | 1.0 |
| Int. UI Developer | * Web frontend design and implementation * UX prototyping | 3 | 1.0 |

#### DevOps Team

The DevOps team is responsible for:

* Configuration of <Environment 1>
* Configuration of <Environment 2>
* Deployment and runtime monitoring of <Application 1>

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Responsibility | Count | FTE |
| DevOps Architect | * Operations analysis and design | 1 | 0.2 |
| Int. DevOps Engineer | * System deployment * Environment setup * Production operations support | 3 | 1.0 |

# Appendix A – Cross-Reference

The Cross-Reference section provides convenient lookup tables mapping views, drivers, and risks.

## Driver Fulfillment Table

The section maps the architectural drivers elicited in this document to the decision views. When a driver is not addressed by the Vision it is marked as not fulfilled.

|  |  |  |
| --- | --- | --- |
| Driver | Fulfilled? | Decision Views |
| <driver hyperlink> | <Yes/No> | <Links to relevant decision views> |

## View Reference Table

The section lists the documented requirement and architectural views of the solution accompanied with the brief description and the stakeholders who should be interested in using this view at the implementation phase. The views are documented following the architecture documentation guidelines defined in [Appendix B – How View is Documented](#_Appendix_B_–).

|  |  |  |
| --- | --- | --- |
| View Name | Description | Concerned Stakeholders |
| [Use Case View <View Name>](#_Use_Case_View) |  |  |
| [Domain View <View Name>](#_Domain_View_<View) | <view brief description> | <stakeholder roles> |
| [Decision View <View Name>](#_Decision_View_<View) | The view defines this and that architectural decision targeting those and those drivers | Product Manager, Architect, Dev Team |
| [Implementation Milestones](#_Implementation_Milestones) |  |  |
| [Estimate](#_Estimate) |  |  |
| [Team Structure](#_Team_Structure) |  |  |
| <view hyperlink> | <view brief description> | <stakeholder roles> |

## Trade-off Reference Table

The section maps trade-offs, risks, and sensitivity points to the decision views where they are analyzed.

|  |  |  |
| --- | --- | --- |
| Point | Description | Decision Views |
| <point hyperlink> | <type: Risk, Nonrisk, Sensetivity Point, Trade-off and description> | <Links to relevant decision views> |

## Element Catalog

The section maps elements defined in the documentation to the views where they can be found.

|  |  |  |
| --- | --- | --- |
| Driver | Description | Decision Views |
| <element hyperlink> |  | <Links to relevant decision views> |

# Appendix B – Architecture Design Methodology

The architectural design of the solution follows the formal methodology called “Augmented Attribute Driven Design” (Augmented ADD). This methodology is developed by Carnegie-Mellon Software Engineering Institute to standardize on the industry best practices and to build the software architectures robustly and rationally within practicable schedules.

The figure defines Augmented ADD in detail.



Figure A-1. Augmented Attribute Driven Design

There are a few major principles guiding this process:

* The solution architecture design must be driven and rationalized by the explicit set of the functional and business requirements, non-functional qualities of the system (such as performance, availability, usability, maintainability, etc.) defined by means of the Quality Attribute Scenarios, and various design constraints. All these are called Architectural Drivers. The architectural drivers are elicitated, prioritized, and approved by the solution stakeholders.
* The process is iterative. Each iteration starts with selection of the subset of architectural drivers to address and completes with documenting the design decisions made to satisfy those.
* The design decisions are based on selection among design concepts such as Reference Architectures, Patterns, Frameworks, Tools, etc. Selection relies on reasoning how well aparticular candidate concept matches the architectural drivers, what pros, cons, and risks it brings to the table in comparison with other candidates.
* The selected design concepts are instantiated, clarified, and transformed to seamlessly work together within the frame of the solution design and implementation.
* The results of the design process are recorded with the appropriate level of detail in the documentation called “Architecture Vision” which serves as an architectural entry point into the solution implementation.

Architectural Driver Elicitation is one of the most critical parts in preparation and execution of this process. While designed product features, functional requirements, and various business, resource, and other constraints are typically well understood by the stakeholders the non-functional requirements concerning System Quality Attributes such as performance, capacity, resource utilization, reusability of the specific components, maintainability of the solution implementation, its deployability and others can remain more obscure. At the same time these Quality Attributes drive much of the design decisions about the future solution architecture.

To improve the process of the quality attribute related requirements elicitation the Quality Attribute Workshop (QAW) methodology is used. In essence, QAW is the process of collaborative generation and prioritization of the scenarios defining how well the solution or its parts are expected to work under certain conditions executed by the architects along with the group of the solution stakeholders. Each scenario must be specific enough and provide a testable measure of response by the system.

# Appendix C – How View is Documented

## View Structure

A view is the main unit of the Architecture Vision . Its purpose is to document, explain, and rationalize about a particular design decision (or multiple related design decisions). In its simplified form the view is also used to document a set of requirements in the form of Use Case and Domain Model diagrams.

The view typically includes these seven parts, some of them can be omitted:

Intent

The information about the purpose of this view and the list of the architectural drivers addressed by this view provide the Intent of the view. The architectural driver references link to those defined in the [Architectural Drivers](#_Architectural_Drivers) section and can include features, use cases, quality attribute scenarios, constraints, etc. This section can also directly describe the problem(s) to solve with the view.

Context

The context of the view means the larger part of the designed solution within which this view exists. For instance, the view can define the decomposition of the application source code into layers. In this case the context of this view will be the solution itself or its subsystem including the application as a black box. This section would often link to another view defining this context or in other cases could directly embed the diagram and textual description of the context.

Representation

The Representation defines and depicts the design decisions made to address the specified intent in the specified context. It usually includes a diagram drawn with UML or another graphical notation and textual description of the design decisions made. In some cases a table or some other form of representation can be used instead of or in addition to the the diagram.

Element Catalog

The Element Catalog lists the elements defined in the view defining their types, responsibilities, semantics, constraints, and other important traits. When needed the relationships, operations, and attributes can be defined as well. When an element is defined on more than one view there is no need to duplicate its definition. A link to its original definition can be used instead.

Interfaces

The section defines important element interfaces and integration points in as much detail as needed to enforce the clear contracts between elements and subsystems. These interfaces are typical candidates for inclusion:

* REST API
* SOAP Web Services via linked or embedded WSDL and XSD
* Message Broker queue and exchange names to bind
* Message formats for XML, JSON, CSV, and other types of messages
* RPC
* Important code module (classes, java interfaces) interfaces

The documented interfaces can be provided and/or required.

This is an example what the documentation for a REST resource might look like.

|  |  |
| --- | --- |
| Title | Start indexing documents. Indexing will be performed asynchronously. Use status call to get status of the index |
| URL | /documents/index |
| Method | POST |
| Request Body | {“startDate”: “2012-01-30”, “endDate”: “2013-01-30” }  Also you can miss date parameters and just use: {} |
| Success Response | Response Headers:  Status Code: 200 OK |
| Error Response | Code: 401 UNAUTHORIZED |
| Sample Call | http://host/documents/index |

These parts pertinent to the REST web service definition cen be included into the resource documentation:

* Resource path sans the base path and version (<http://host/application/v1>)
* HTTP method
* Headers
* Cookies (optional)
* Query (optional)
* JSON passed in the request
* Response code. For more details, please refer to [section 6 of RFC 2616](http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.1.1)
* Response error codes
* JSON passed back in the response (optional)

Variability

The Variability section documents possible ways to configure, customize, and substitute elements and relationships which are parts of this view in response to potential requirement and external dependency changes as well as in the interest of deployment, integration, and other operations on the solution. Typical variability candidates:

* Configuration options in files, data stores, etc.
* Scripting
* Substitution of components, classes, third-party services
* Using alternative frameworks, libraries, servers, pieces of infrastructure, and other technologies.
* Limits within which the resource amounts can vary to provide for the system requirements

Reasoning

The Reasoning section rationalizes about the decisions taken and described in the view. It can also document the alternative decisions and compare them to the selected ones to show why they were discarded.

The section points out the sensitivity points, risks, trade-offs, unmade decisions and more.

Typical approaches to Reasoning section completion:

* Alternative options analysis
* Trade-off Analysis
* References to authoritative sources
* Calculations
* Cost-Benefit Analysis
* Reference to Proofs of Concept, Prototypes, Simulations
* Reference to previous experience

**Alternative options analysis** can be presented in the form:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Alternatives | Driver 1 | Driver 2 | Driver 3 | Reason for discarding |
| Alternative1 | ++ | - | + | Accepted |
| Alternative2 | + | o | o | Reasons a, b, c |
| Alternative3 | **--** | **+** | **o** | Reasons x, y, z |
| <alternative> |  |  |  | <reasons why alternative is discarded> |

Where:

* ++: a very positive influence on the quality goal can be expected
* +: a positive influence on the quality goal can be expected
* o: no influence on the quality goal can be expected
* -: a negative influence on the quality goal can be expected
* --: a very negative influence on the quality goal can be expected

**Trade-off analysis** is another way to analyze the decisions for risks and trade-offs. Typically a form like the one below is used to capture its results in the reasoning.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario #: Number | Text of quality attribute scenario | | | |
| Architectural Decisions | Sensitivity | Tradeoff | Risk | Nonrisk |
| Architectural decisions relevant to this scenario that affect quality attribute response | Sensitivity Point # | Tradeoff Point # | Risk # | Nonrisk # |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |
| Reasoning | Qualitative and/or quantitative rationale for why the list of architectural decisions contribute to meeting each quality attribute requirement expressed by the scenario | | | |

The method is described in depth in this document:



Other types of reasoning are expected to come in the form suited to a specific case.

## Graphical Notation

The section defines the graphical and color notation used in the document to draw the diagrams in the views.

This Vision primarily uses **UML 2.4.1** documented at [www.uml.org](http://www.uml.org/) as the graphical modeling language of choice.

These online resources can be used to look up the common ways of UML modeling:

* [UML 2 Tutorial](http://www.sparxsystems.com/resources/uml2_tutorial/index.html) from Sparx Systems
* [www.uml-diagrams.org](http://www.uml-diagrams.org/)

Additionally, these graphical elements are used through the documentation:

|  |  |
| --- | --- |
| Graphics | Meaning |
|  | Mobile Device with the client application installed |
|  | Third-party components are marked with blue color |
| <image> | <image meaning> |

# Appendix D – Estimation Methodology

This section refers to the estimation methodology used to produce the implementation estimates.

The approach to the estimation in the phase of initial solution architecture generation involves several steps:

1. Decompose the solution into sufficiently small and well-defined parts.
2. Estimate their size and/or relative complexity by analogy and involving, when possible, other experts.
3. Translate the obtained estimates into the effort estimates for an “average” developer of a specified level of knowledge and experience. This translation accounts for the company historical log of similar implementations.
4. Adjust the estimates for the identified key risks which can potentially affect the required effort or schedule.
5. Based on the project business priorities and the estimated effort
   1. either plan the Milestone delivery schedule (usually in project iterations or months) and deduce the proposed team size and structure from it
   2. or define the team size and structure and plan the schedule based on the amount of effort the team can cover over an iteration.
6. Project and approve with the stakeholders the iterative implementation process and delivery based on the above estimates.
   * <Amend or replace the methodology description if the different one was used>

# Guide to Architecture Vision Documentation – DELETE ME!

[DELETE THIS SECTION AND UPDATE THE TABLE OF CONTENTS BEFORE SENDING TO CLIENT]

## Documentation Guidelines

Architecture Vision is a document or documentation package defining the target solution architecture based on the significant architectural drivers. Depending on the actual project needs it may include items from this list:

* Executive Summary
  + Key Decisions
  + Key Risks
* Cross-references
  + View list with view names, references, brief summaries, interested stakeholders
  + Driver fulfillment list
  + Risk, trade-off, sensitivity point, unmade decision list
* Elicited Architectural Drivers
  + Solution business case and major features
  + Service Level Agreement
  + Functional requirements in the form of features and use cases
  + Domain model
  + Quality attribute scenarios
  + Design constraints including business and technical constraints
* Defined Solution Architecture
  + Solution context
  + Solution architecture big picture
  + Technology stack
    - Programming languages
    - Frameworks and libraries
    - Third-party services and other dependencies
    - External integration points
    - Development tools
  + Solution architecture decomposition views
    - Code organization, layers, design pattern instantiations (Modules views)
    - Runtime components, services, and processes, integrations (Components & Connectors views)
    - Packaging, deployment, installation, and other allocation views
  + Behavior scenarios and diagrams for the views (Sequence, Activity, and other diagram types)
  + Interfaces and variability (configuration, customization) for the elements defined in the views
  + Reasoning for the decisions defined by the views
* Operation Plan
  + Transition Phase
    - Environments
      * Development
      * Testing
      * Continuous Integration
      * Staging
      * Production
    - Infrastructure
      * Hosting Platform
      * Hardware resources
      * Virtualized resources
    - Deployment/Provisioning
    - Monitoring
    - Disaster Recovery
    - Incedent Management
    - Audit and Logging
    - Infrastructure Management
    - Data Management
    - Application Management
    - Backup and Restore
    - System Security Management
    - Availability Management
  + Operation Phase
    - Service Operation Support
    - CMOD Activities
    - Continuous Improvement Process
* Recommended Implementation Roadmap
  + Implmentation Milestones
  + Implementation Deliverables
  + Estimates
    - Complexity
    - Effort
    - Schedule
    - Team size
  + Team
    - Team member skillsets
    - Team structure

The actual content needs are to be determined by the architect in collaboration with the client.

The editors can drop the sections irrelevant or out of scope in the specific consulting or add new ones. They must follow the general document style and structure while doing so.

1. When editing the actual document always remove <> from the placeholders when substituting with the actual values.

Document Properties

The document properties (Word info page) should be updated for the actul vision document. Title must be edited to include the actual solution name, the authors should be added. The file name should include the solution name, document name (Architecture Vision), client name, and preferably, current revision.

[Title Page](#_top)

Fill in the actual Solution Name, date (title page footer), version, etc. Update the Solution Name and version in the page headers.

[Revision History](#Revision_History)

Add a new row with next revision to reflect the next iteration of work on this document. As a rule, there are at least the initial revision, final draft, revision approved for the client, final revision with the fixes introduced from the presentation to the client. Sometimes there are more revisions.

The revisions before final have 0.xx numbering. The final version has 1.0 revision. The final and amended version has 1.xx numbering, etc.

Version numbers on the [Title Page](#_top) and in the headers need to be updated for each new revision added to this table. Often it also makes sense to update the file name and properties to reflect the current revision number.

[Introduction](#_Introduction)

Introduction provides meta-information such as the purpose, scope, documentation road map, etc. about the Architecture Vision document.

[Purpose](#_Purpose)

Purpose specifies the purpose of this Architecture Vision.

[Definitions](#_Definitions)

This subsection should provide definitions of all terms, acronyms, and abbreviations required to properly interpret the Architecture Vision. This information may be provided by reference to the project Glossary.

[References](#_References)

This subsection should provide a complete list of all documents referenced elsewhere in the Architecture Vision. Each document should be identified by version, title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained.

Smaller documents provided by the Client can be attached to this table for easiness of reference and review.

These might be requirement specs, business level overviews, vision drafts, international/national standards, etc.

[Implementation](#_Implementation_Deliverables) Deliverables

List the essential deliverables referring to the subsections and items in the Architectural Drivers and Solution Architecture sections as appropriate

Assumptions [and](#_Assumptions_and_Limitations) Limitations

Describe dependencies, risks and key assumptions known or expected at this time. Touch on what would be expected and what should be excluded.

Short reference:

A risk is an uncertain event or condition that, if it occurs, has a positive or a negative effect on a proposal project objective. A risk has a cause and, if it occurs, a consequence.

An assumption is where there may be external circumstances or events that are to be considered to occur for the proposed project to be successful. If you believe these external events are likely to happen, then you have an assumption. If you have an external event that must occur, but you are uncertain that it will, it is a risk.

A dependency is a situation where some part of the proposed project requires some other event, activity, or effort to be completed before the proposed project can be successful. You may also ‘assume’ that such an event will happen, but tracking as a dependency calls out with more emphasis that the chain of events must be completed correctly.

[Team Skillset](#_Team_Skillset)

Define required skillset for the team members based on the technology stack proposed in the solution arcchitecture

Team [Structure](#_Team_Structure)

List team members that will be involved into implementation phase with their core responsibilities. There can be multiple sub teams as in example

## Architecture Vision Effort Estimation

You should have clear understanding of how much time can take architecture vision preparation. The entire process involves three major activities:

1. **Information Discovery** – requirement and constraint brainstorming and analysis, technology research, communication with the stakeholders, etc.
2. **Decision Making** – selection and analysis of the appropriate design decisions to include into the architecture vision.
3. **Documentation Preparation** – actual documentation of the architecture vision including all the parts of this document.

Information Discovery and Decision Making can vary in major ways depending on the client specifics, solution size and complexity, current state of requirement elicitation, required granularity of the design, on-site vs. off-site mode of the discovery, architect’s personal background, etc.

We recommend to use these numbers as minimal required architect’s effort for #1 + #2:

* Small Solution – 16+ hours
* Medium Solution – 40+ hours
* Large Solution – 80+ hours

Documentation Preparation can be estimated based on these numbers:

1. Introduction – 1+ hours
2. Executive Summary – 1+ hours
3. Business Case – 1+ hours
4. SLA – 1+ hours
5. Single Use Case or Domain View – 1+ hours
6. Constraints – 1+ hours
7. Quality Attribute Scenarios – 2+ hours
8. Single Decision View – 2+ hours
9. Implementation Roadmap – 1+ hours
10. Estimates – 4+ hours
11. Team – 1+ hours
    * As an example if it is required to document a vision including 2 Use Case views, 1 Domain Model view, 5 Decision Views with the full Implementation Roadmap the total preparation effort can be estimated as:
    * Total = 1+1+1+1+2\*1+1\*1+1+2+5\*2+1+4+1 = 26+ hours
    * If this is a small solution the grand total for the discovery and architecture vision creation can take 16+26 = 42+ hours or 1+ man/week.

## Styles to Use in the Document

Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text Body Text

Body Text Indented Body Text Indented Body Text Indented Body Text Indented Body Text Indented Body Text Indented Body Text Indented Body Text Indented Body Text Indented Body Text Indented

Bulleted List

* Item 1/Bulleted List

Item 2/Bulleted List Indented

Item 3/Bulleted List Indented 2

Numbered List

1. Item 1/Numbered List
2. Item 2/Numbered List
   1. Numbered List Letter
   2. Numbered List Letter

Indented List

* 1. Item 1/Numbered List Indented
  2. Item 2/Numbered List Indented

Note Text here Note Text here Note Text here Note Text here Note Text here Note Text here Note Text here Note Text here Note Text here Note Text here Note Text here Note Text here

1. Important Text here Important Text here Important Text here Important Text here goes Important Text here Important Text here Important Text here Important Text goes here
2. Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here Tip Text here

Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Column 1 | Column 2 | Column 3 | Column 4 |
|  | Text | Text | Text | Text |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. It is best to use vector graphics formats (.emf) for your diagrams to make sure the quality is not lost when adjusting the size and the document size is smaller. SVG format should be converted to .emf before embedding, otherwise it is converted to the raster image.

Figure



Figure . Type a name for your figure

Figure Indented



Figure . Type a name for your indented figure

### Custom Section

#### Custom Section