|  |
| --- |
| **Disclaimer**  This is a **template** for the Software Architecture Document (SAD) that students may use. It provides a starting point for the **preparation** of the SAD.  **Note to authors**  If you add any new sections to the document please make sure that you maintain the header and text styles.  Before submission of the first draft of this document please make sure to update the Table of Contents and to delete this page.  **Author**: Dr. C. Constantinides <cc@cse.concordia.ca> |

**Software Architecture Document**

Version 1.0

for

LOTUS Calendar

Prepared by

|  |  |  |
| --- | --- | --- |
| Saif Mahabub | 27392974 | rhythmsaif@gmail.com |
| Alexander Rosser  Philippe Kuret  An Ran Chen  Adriel Fabella  Costa Papadakos | 27543069  27392680  27277385  27466005  26665691 | arosser95@gmail.com  philippekuret@gmail.com  archen94@gmail.com  adriel.fab@gmail.com  cotsop@gmail.com |

|  |  |
| --- | --- |
| Instructor: | Dr. C. Constantinides |
| Course: | SOEN 343 |
| Date: | 25/11/2016 |

**Document history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 13/11/2016 | 1.0 | Rough Draft, Introduction | Saif Mahabub |
| 16/11/2016 | 1.1 | Functional requirement added | Philippe Kuret |
|  |  |  |  |
|  |  |  |  |

**Table of contents**

[1. Introduction](#_30j0zll)

[Purpose](#_1fob9te)

[Scope](#_3znysh7)

[Definitions, acronyms, and abbreviations](#_2et92p0)

[2. Architectural representation](#_tyjcwt)

[3. Architectural requirements: goals and constrains](#_4d34og8)

[Functional requirements (Use case view)](#_2s8eyo1)

[Non-functional requirements](#_17dp8vu)

[4. Use case view (Scenarios)](#_3rdcrjn)

[5. Logical view](#_26in1rg)

[Layers, tiers etc.](#_lnxbz9)

[Subsystems](#_35nkun2)

[Use case realizations](#_1ksv4uv)

[6. Data view (optional)](#_1y810tw)

[7. Size and performance](#_4i7ojhp)

[8. Quality](#_2xcytpi)

**List of figures**

# Introduction

The following introduction provides an overview of the Software Architecture Document of the project.

## Purpose

This document provides a detailed architectural overview of the LOTUS Calendar system with the use of a few different architectural views, in order to represent different aspects of the system. It is intended to represent the significant architectural decisions that are made on the system. This is viewed by stakeholders and the development team.

## Scope

The scope of the document is to depict the architecture of the LOTUS Calendar system. It describes the architectural goals and constraints, the Use case view, Logical view and Data view.

## Definitions, acronyms, and abbreviations

Provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the Software Architecture Document. This information may be provided by reference to the project’s Glossary. For example:

**RUP**: Rational Unified Process

**UML**: Unified Modeling Language

**SAD**: Software Architecture Document

# Architectural representation

Describe the top-level architectural style of the system and the view model you will adopt. Additionally describe what each individual view will provide. Many enterprise software systems are modeled using the 4+1 view illustrated in Figure 1.

**Logical view** : Audience: Designers. The logical view is concerned with the functionality that the system provides to end-users. UML Diagrams used to represent the logical view include **Class diagram**, and **interaction diagrams** (**communication diagrams**, or **sequence diagrams**).

1. **Use case view** (also known as Scenarios) : Audience: all the stakeholders of the system, including the end-users. The description of an architecture is illustrated using a small set of use cases, or scenarios which become a fifth view. The scenarios describe sequences of interactions between objects, and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype. Related Artifacts : **Use-Case Model**.

**Data view** (optional): Audience: Data specialists, Database administrators. Describes the architecturally significant persistent elements in the data model . Related Artifacts: **Data model**.

# Architectural requirements: goals and constraints

Requirements are already described in SRS. In this section describe *key* requirements and constraints that have a significant impact on the architecture.

## Functional requirements (Use case view)

The system will allow the user to securely login.

The system will allow all users to see the reservations made in all room.

The system will allow the user to change the date while looking at the reservations.

The system will allow only one user per room to add, cancel and modify reservation their reservation.

The overview below refers to architecturally relevant Use Cases from the Use Case Model (see references).

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Name** | **Architectural relevance** | **Addressed in:** |
| UC1 | Log in | Securely log the user into the system | Section number where this use case or scenario is addressed in this document |
| UC03 | View Bookings | View the reservations and availabilities schedule. | ? |
| UC04 | Change View Booking Date | Change the viewed date. | ? |
| UC05 | Create Reservation | Create a new reservation | ? |
| UC06 | Cancel Reservation | Cancel a pre-existing reservation | ? |
| UC07 | Modify Reservation | Modify a pre-existing reservation | ? |

## Non-functional requirements

Describe the architecturally relevant non-functional requirements, i.e. those which are important for developing the software architecture. Think of security, privacy, third-party products, system dependencies, distribution and reuse. Also environmental factors such as context, design, implementation strategy, team composition, development tools, time to market, use of legacy code may be addressed.

Usually, the non-functional requirements are already in place and can be referenced here. This document is not meant to be the source of non-functional requirements, but to address them. Provide a reference per requirement, and where the requirement is addressed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Name** | **Architectural relevance** | **Addressed in:** |
| e.g. Vision, SRS. | Name of requirement. | Description on why this requirement is relevant to the software architecture. | Section number where this requirement is addressed in this document. |

# Use case view (Scenarios)

The scenarios (or functional view) represent the behavior of the system as seen by its actors. Use case scenarios describe sequences of interactions between actorsd and the system (seen as a black box) as well as between the system and external systemsTthe *UML use case diagram* is used to capture this view.

# Logical view

The logical view captures the functionality provided by the system; it illustrates the collaborations between system components in order to realize the system's use cases. Describe the architecturally significant logical structure of the system. Think of decomposition in tiers and subsystem. Also describe the way in which, in view of the decomposition, Use Cases are technically translated into Use Case Realizations.

## Layers, tiers etc.

Describe the top-level architecture style. Deploy a *UML class diagram*.

## Subsystems

Describe the decomposition of the system in subsystems and show their relation.

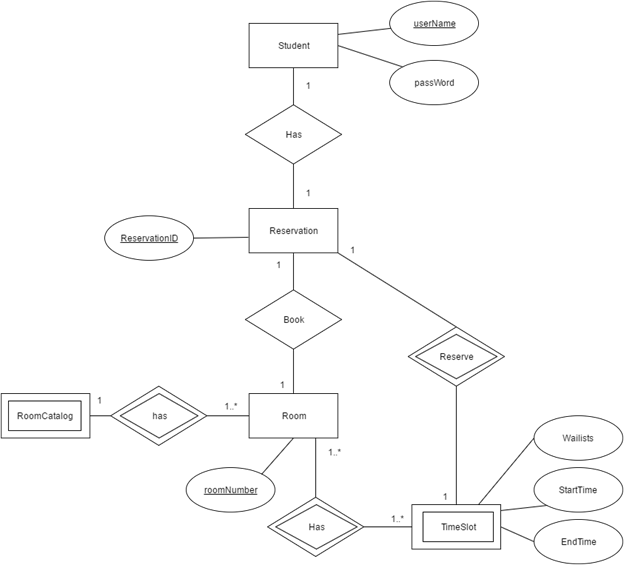
**Architecturally significant design packages**

Desribe packages of individual subsystems that are architecturally significant. For each package nclude a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

## Use case realizations

In this section you have to illustrate how use cases are translated into *UML interaction diagrams*. Give examples of the way in which the Use Case Specifications are technically translated into Use Case Realizations, for example, by providing a sequence-diagram. Explain how the tiers communicate and clarify how the components or objects used realize the functionality.

# Data view (optional)



**Relational schema for the ER diagram**

Student (userName, passWord)

Reservation (ReservationID, ***roomNumber***, ***userName***)

Room (roomNumber)

RoomCatalog (***roomNumber***)

TimeSlot (StartTime, EndTime, Waitlists, ***ReservationID***)

Foreign key is in bold.

# Size and performance

Describe how the architecture supports the key sizing and performance requirements, as described in the *Supplementary Specification*. For example:

Volumes:

* Estimated online orders : 100 a day, with peaks in the evening
* Registered individual customers : about 100
* Corporate customers : about 200

Performance:

* Time to process and online payment (credit card validation + confirmation) : less that 15 seconds required.

# Quality

A description of how the software architecture contributes to the quality attributes of the system as described in the ISO-9126 (I) standard. For example: The following quality goals have been identified:

Scalability:

* Description : System’s reaction when user demands increase
* Solution : J2EE application servers support several workload management techniques

Reliability, Availability:

* Description : Transparent failover mechanism, mean-time-between-failure
* Solution : : J2EE application server supports load balancing through clusters

Portability:

* Description : Ability to be reused in another environment
* Solution : The system me be fully J2EE compliant and thus can be deploy onto any J2EE application server

Security:

* Description : Authentication and authorization mechanisms
* Solution : J2EE native security mechanisms will be reused