<Project Name>

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |

Note: When you include figures

* Use figure numbers and figure captions.
* Use diagrams/ images/ screen shots with high resolution to get a clear figure
* Use the figure captions in the form of Figure 1. <<caption>> and when explain it in the text, use the abbreviation “Figure 1,” even at the beginning of a sentence.
* When you use a tool to draw diagrams, change the font settings of the diagram; It is better to have font in black colour/ large in size (12pt) and if possible do not fill the objects/ elements in the diagram with a colour (keep the background colour white, for clear visibility)
* Also, try to re-locate the objects closely in a way that take less space (you may drag elements/objects close to each other).
* This will be useful to get a clear image; in order to make the diagram small in size without reducing the resolution quality.
* When you include images in your reports, make them “in line with text” ( picture tool bar 🡪 wrap text 🡪 in line with text) and include the caption accordingly. If you include two or more images together ( in a row), group them.
* Describe each diagram with few sentences.

When you draw diagrams (eg. Sequence diagram) do not include only two object called “user” and “system”. Include all the internal objects within the system, without considering the system as a black box. For example: for a mobile application the main system may consists of sub objects such as , <<UI>>:main\_Interface, :controller, <<UI>>:analysis\_Interface, :local\_DB, etc. (this is only an example; use your own terms).

References:

* Indicate the tools you have used to draw the diagrams

Useful theory for design diagrams : (relationships in Class diagram)

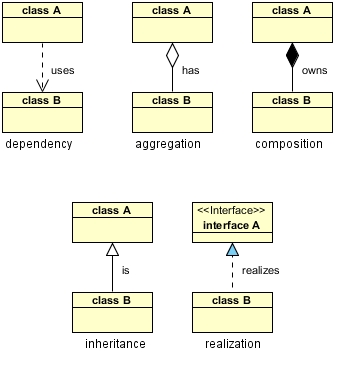


Table of Contents

1. Introduction 2

1.1 Purpose 2

1.2 Scope 2

1.3 Definitions, Acronyms, and Abbreviations 2

1.4 References 2

1.5 Overview 2

2. Architectural Representation 2

3. Architectural Goals and Constraints 2

4. Use-Case View 2

4.1 Use-Case Realizations 2

5. Logical View 2

5.1 Overview 2

5.2 Architecturally Significant Design Packages 2

6. Process View 2

7. Deployment View 2

8. Implementation View 2

8.1 Overview 2

8.2 Layers 2

9. Data View (optional) 2

10. Size and Performance 2

11. Quality 2

# Introduction

This **Document** provides an overview of the “Smart Planner” application. It includes different perspectives of the application being developed.

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system. The specific audiences for the document is identified, with an indication of how they are expected to use the document.

[This section defines the role or purpose of the **Software Architecture Document**, in the overall project documentation, and briefly describes the structure of the document. The specific audiences for the document is identified, with an indication of how they are expected to use the document.]

## Scope

[A brief description of what the Software Architecture Document applies to; what is affected or influenced by this document.]

This document will describe the “Smart Planner” application in more detailed and in an architectural manner. The developer can use this a manual to get a clear understanding of the application and to implement the system. Further the clients and the other stakeholder can get a clear understanding of the system being developed by referring this document.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User | Person who use the Android application to schedule tasks |
| Server | Web server coded using PHP |
| System | Web server together with the Android application |

## References

[This subsection provides a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

- Indicate the tool you have used to draw the diagrams

## Overview

[This subsection describes what the rest of the **Software Architecture Document** contains and explains how the **Software Architecture Document** is organized.]

The rest of the document includes Architectural representation of the system by dividing it into logical view, process view, implementation view, deployment view and use-case view. The above views will be described with the use of relevant diagrams. The architectural goals and constraints will also be described. Later the size and performance and the quality of the system will described.

# Architectural Representation

[This section describes what software architecture is for the current system, and how it is represented. Of the **Use-Case**, **Logical**, **Process**, **Deployment**, and **Implementation Views**, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]

This section describes the software architecture of the “Smart Planner” application with respect to several views. They are,

* Use-case view

This view describes the main set of functionalities of the system using use-cases and scenarios. The interactions between the users with the system which is considered as a black box will described in this view.

* Logical view

This view describes how the system is structured. the model will be used to understand the system properly.

* Process view

This view describes how the process flow within the system works. With or without the interaction of user an activity will start. Then the data and instructions will be exchanged within the components of the system. This view will consider detail about those processes.

* Deployment view

This view describes the environment which the final system will function. The specific requirements of the environment will be considered here.

* Implementation view

This view describes about the necessary components and how they are bundled to be needed for the implementation of the system.

# Architectural Goals and Constraints

[This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

# Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

Identify sufficient number of use cases for your system. (eg. Less than five use cases are not sufficient for the scope of the project)

## Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality.]

**Include usecase diagrams and their scenarios.**

For each important/ main/ selected usecase include the realization of a scenario. For example :

|  |  |
| --- | --- |
| **Use case name** | <<usecase name>> |
| **Actor** | <<actor names>> |
| **Description** | <<Describe the purpose of the use case>> |
| **preconditions** | <<any preconditions that should be satisfied before the use case happens>> |
| **Main flow** | << stepwise description>> |
| **Successful end/post condition** |  |
| **Fail end/post condition** |  |
| **Extensions** |  |
|  | If one of the items is not available you may indicate <<N/A>> |

# Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

## Overview

[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

## Architecturally Significant Design Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

**Include the Class diagram and describe it**

# Process View

[This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.]

**Include the Activity diagram and Sequence diagram and describe**

# Deployment View

[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.]

**Include the Deployment diagram if available and describe**

# Implementation View

[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]

## Overview

[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers. ]

## Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

**Include the Package diagram and describe**

# Data View (optional)

[A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.]

# Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

# Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]

**12. References**

Refer any data/ information in a standard format (eg. IEEE referencing style)

For different algorithms/ techniques/ theories you can refer text books.

For tools you can refer web pages.

For similar work you can refer research paper articles that describe the work.

You may include white paper articles for the description of technologies; web URL for the tool references. When you refer such a web page, you have to indicate the (Accessed on <<date>>)