Software Design Specification

The Software Design Specification Outline

1. Introduction

1.1 Purpose of this document

A software design specification is a written description of a software product, that a software designer writes in order to give a software document team overall guidance to the architecture of the software project. An SDS usually accompanies an architecture diagram with pointers to detailed feature specifications of smaller pieces of the design. Practically, the description is required to coordinate a large team under a single vision, needs to be a stable reference, and outline all parts of the software and how they will work. .

1.2 Scope of the development project

The purpose of this project is for each restaurant to have its own personalized app with multiple uses like: Personalized Loyalty Programs, Delivery management and tracking, Inventory management, comprehensive menu display.

1.3 Definitions, acronyms, and abbreviations

IEEE : Institute of Electrical and Electronics Engineers
SDS : Software Design Specification
FODO : Food Ordering Application

1.4 References

1.4.1: IEEE SDS template

1.5 Overview of document

This SDS is divided into seven sections with various sub-sections. The sections of the Software Design Document are:

1. Introduction: describes about the document, purpose, scope of development project definitions and abbreviations used in the document.
2. Conceptual Architecture/Architecture Diagram: describes the overview of components, modules,structure and relationships and user interface issues.
3. Logical Architecture:describes Logical Architecture Description and Components.
4. Execution Architecture: defines the runtime environment, processes,deployment view.
5. Design Decisions and Trade-offs: describes the decisions taken along with the reason as to why they were chosen over other alternatives.
6. Pseudocode for components: describes pseudocode, as the name indicates.
7. Appendices: describes subsidiary matter if any.

2. Conceptual Architecture/Architecture Diagram

Conceptual Architecture is “Context” for the system’s use.

2.1 Overview of modules / components

This subsection will introduce the various components and subsystems.

2.2 Structure and relationships

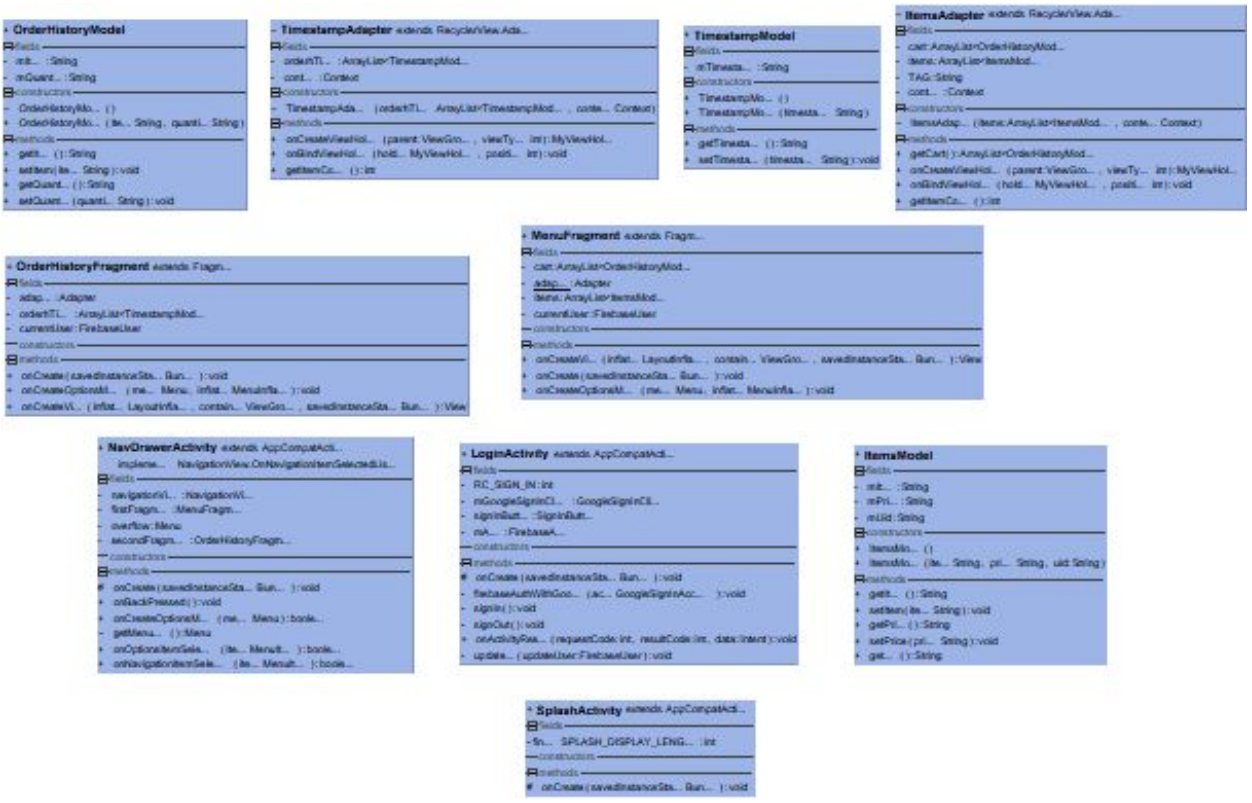
Make clear the interrelationships and dependencies among the various components. Structure charts can be useful here. A simple finite state machine can be useful in demonstrating the operation of the product. Include explanatory text to help the reader understand any charts.

2.3 User interface issues

This section will present the main principles of the product's user interface. Use the personas defined in section 2.1 of your SRS to make specific examples. This section should not touch on technical details. You may want to include sketches and specific text messages.

3. Logical Architecture

Class Diagram:



3.1 Logical Architecture Description

Discuss some details(generic) of Logical Architecture

3.2 X Component (or Class or Function ...)
Use exactly the template you define in 3.2. If a part of the template is not applicable,

4.0 Execution Architecture
Runtime environment required is any device supporting Android Operating System with the minimum version of KitKat(4.4), Android Studio as a deployment platform.

4.1 Reuse and relationships to other products
This application is designed as a generic app with full functionalities and generic in the sense that we can manually set the name and branding of the respective restaurant. After we have delivered the app to the restaurant, the restaurant can set their app contents like menu and photos. Reusing the app is an important strategy we have used in our project and this will help us to make app for different restaurants easy and quick.

5.0 Design decisions and tradeoffs
The design decision to use two apps separately for restaurant and restaurant's customer is to provide encapsulation. It may have been possible to make one application for both the restaurant and the customer, but it is easier to manage for the restaurant its own app for itself and its customer.

6.0 Pseudocode for components

7.0 Appendices (if any)

/*SDS component template
The template given below suggests a reasonable structure for giving a thorough description of each component described in Part 3 of the SDS. The specific information depends in part on the design approach. Your team must adapt this template to your needs and describe it in section 3.1 of your SDS.

Identification	The unique name for the component and the location of the component in the system.
Type	A module, a subprogram, a data file, a control procedure, a class, etc
Purpose	Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements.

Function	What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.
Subordinates	The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part.
Dependencies	How the component's function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components.
Interfaces	Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here.
Resources	A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services.
Processing	The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic.
Data	For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary.

*/