Documentation

Technical Documentation

Riadh Amami

November 12, 2020

Title: Documentation
Version: 1.0
Company: CompanyName
Creation date: November 12, 2020

File history:

Date	Author	Description
November 12, 2020	Riadh Amami	Document creation

Contents

List of Figures

List of Tables

Listings

1 ChapterI:Introduction

1.1 Context of the Project

1.1.1 Project context

1.1.2 Problem

1.1.3 Existing Solutions

1.1.4 Our Solution

1.2 Functional Study

1.2.1 Functional Needs

1.2.2 Non-Functional Needs

2 Flutter Development Phase

2.1 Use-Case Diagram

2.1.1 Global Use-Case Diagram

In real life, the Use Cases for a small system will be fairly well worked out before work begins on building the system. Large system tends to work more in parallel than linearly. So, some people will work on Use Cases, while others start building code for the Use Cases already completed. Since our Order Processing system is relatively small, we have completed all the Use Cases for it before building the system and we have detailed some basic Scenarios as well. They seem pretty simple. To fully understand the system's purpose, you must know who the system is for, that is, who will be using the system. Different user types are represented as actors. An actor is anything that exchanges data with the system. An actor can be a user, external hardware, or another system. So, finding actors is one of the first steps in defining our system use. Each type of external phenomenon with which the system must interact is represented by an actor. We defined four actors for our system and we determined in the next step, with the use case diagram, the functionalities that are related to each of them.

2.1.2 Roles

This figure allows to draw up the actors within the system, there will be four actors and each user will have different functionality to the others.

2.1.3 Global Use-Case Diagram and Description

The rest of this chapter is the Use Case diagram for our application's system. You'll see a lot of uses relationships between Use Cases. Uses are found early in the process, and allow you to show commonality between parts of the system. Extends tends to be added later, when we find some new requirement or functionality that extends the current system. Since we haven't built the first system yet, we don't have anything to extend.

2.1.4 Detailed Use-Case Diagrams

administrator management module

recruiter management module

candidate management module

2.2 Class Diagrams

2.2.1 Global Class Description

2.2.2 Detailed Class Diagrams

administrator management module

recruiter management module

candidate management module

chatbot module

live chat module

Mailing Module

2.3 Sequence Diagrams

2.3.1 Global Sequence Description

2.3.2 Detailed Sequence Diagrams

administrator management module

recruiter management module

candidate management module

chatbot module

live chat module

Mailing Module

2.4 Mockup

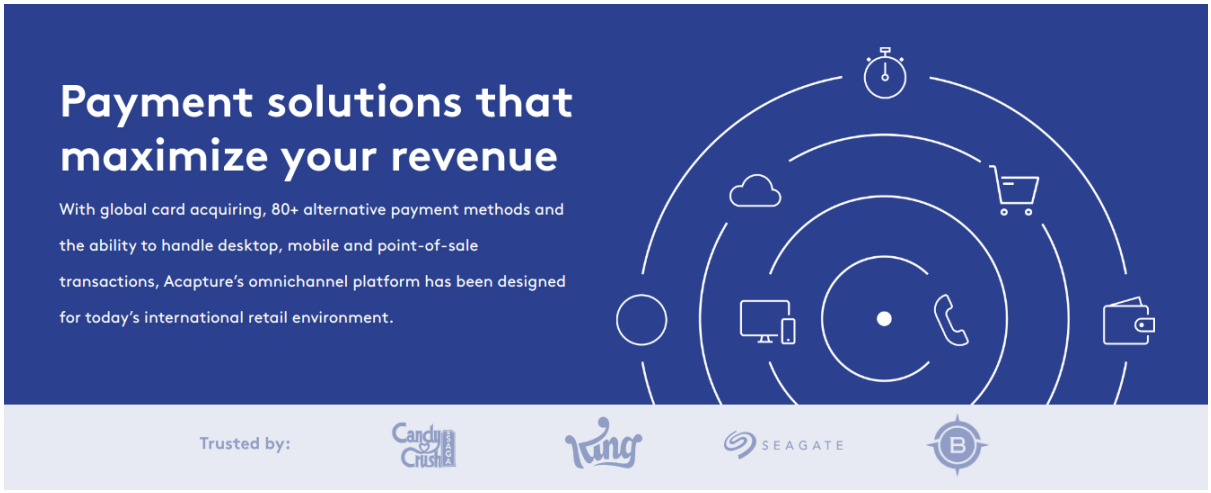

Figure 1: My figure

3 Tables

For the creation of tables it is recommended to use the package `ltablex`, it modifies the `tabularx` environment to combine the features of the tabularx package (auto-sized columns in a fixed width table) with those of the longtable package (multi-page tables)

3.1 Example

This is an example of table, table 2 is a dummy table.

label 2	**label 2**	**label 3**
item 1	item 2	item 3
item 1	item 2	item 3

Table 2: My table