Spor Cantam
Architecture Notebook

1. Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

2. Architectural goals and philosophy

The Spor Cantam online shopping app project does not have any complex deployment concerns. The project is aimed to work on approximately %90 percent of the mobile phones, android 5.0 or higher, so it does not have concern about adapting the legacy android versions. The app is going to be lightweight, so performance and battery/internet usages are going to be optimal. The app does need a long term maintenance as admins to check items that added to server or payments. Other than that the app does not have any serious maintenance operations.

Project goals are:

- Work on different environments (different android phone brands and versions)
- Has +95% crash free users

3. Assumptions and dependencies

The app has basic assumptions for the users such as: Be able to read/write, be able to use android phones, be able to use online payment methods, be able to understand basic tasks.

Dependencies are mostly about the team skill. Every team member must have knowledge about OOP programming and team work experience. Also the app totally under dependency of android libraries. Also a server provider (currently digitalocean) for all the back end work.

4. Architecturally significant requirements

- The system must respond within 5 seconds
- The system must deploy on Android
- The system must be reliable about payments and sales
- The system must control all the sales have been made

5. Decisions, constraints, and justifications

- The database must be a SQL type database
- Mobile app should be developed native and use kotlin language unless java is a better option for specific tasks
- The code must be clean and object oriented
- The project should not waste so much system resources on both ends

6. Architectural Mechanisms

Documentation

For all classes you wrote, you have to identity:
- The main purpose of this class
- The clean purpose of the function you implemented
- Special requirements and cases

Cleanliness

For all code you wrote, you have to follow:

- Code must be written in functions, no duplicated code.
- Variable names should be understandable and informal.
- Spacing should be clean and correct
- Algorithms should be implemented correctly

7. Key abstractions

Customer: The user that going to use the system and place orders
Seller: The user who is gonna sell products and receive the payments
Admin: The person who checks all payments
-3rd party credit card payment service: The service that allows system to take payments
-Account: User identifier, can be seller, customer or admin

8. Layers or architectural framework

The system is going to use MVC pattern.

9. Architectural views

All the diagrams below can be found in the same folder of this document, in pdf format.

- Use Case Diagram: Simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.
- Component Diagram: Illustration of the structure of arbitrarily complex system.
- Package Diagram: Depicts the dependencies between the packages that make up a model.
- Deployment Diagram: Models the physical deployment of artifacts on nodes.