*Bookstore website*

*Version 1.0*

*Software Architecture Document*

*3/2023*

**Revision History**

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

**Table of Contents**

[**1.**](#_heading=h.gjdgxs) **Introduction 3**

[1.1.](#_heading=h.30j0zll) Scope 3

[1.2.](#_heading=h.1fob9te) Definitions, Acronyms, and Abbreviations 3

[1.3.](#_heading=h.3znysh7) References 3

[1.4.](#_heading=h.2et92p0) Overview 3

[**2.**](#_heading=h.tyjcwt) **Architectural Representation 3**

[**3.**](#_heading=h.1t3h5sf) **Architectural Goals and Constraints 3**

[**4.**](#_heading=h.4d34og8) **Use-Case View 3**

[**5.**](#_heading=h.2s8eyo1) **Logical View 3**

[5.1.](#_heading=h.17dp8vu) Overview 3

[5.2.](#_heading=h.3rdcrjn) Architecturally Significant Design Packages 3

[**6.**](#_heading=h.26in1rg) **Process View 3**

[**7.**](#_heading=h.lnxbz9) **Deployment View 3**

[**8.**](#_heading=h.35nkun2) **Implementation View 3**

[8.1.](#_heading=h.1ksv4uv) Overview 3

[8.2.](#_heading=h.44sinio) Layers 3

[**9.**](#_heading=h.2jxsxqh) **Data View 3**

[**10.**](#_heading=h.z337ya) **Size and Performance 3**

[**11.**](#_heading=h.3j2qqm3) **Quality 3**

[11.1.](#_heading=h.1y810tw) Applicability to Technical Reference Model/Standards Profile (TRM/SP) 3

[11.2.](#_heading=h.4i7ojhp) Applicability to Common Services Architecture 3

[11.3.](#_heading=h.2xcytpi) Applicability to HSD&D Core Specifications for Rehosting Initiatives 3

[**12.**](#_heading=h.1ci93xb) **Architectural Mechanism 3**

[12.1.](#_heading=h.1pxezwc) Analysis Mechanisms 3

[12.2.](#_heading=h.2bn6wsx) Analysis-to-Design-to-Implementation Mechanisms Map 3

[12.3.](#_heading=h.qsh70q) Implementation Mechanisms 3

[12.3.1.](#_heading=h.3as4poj) <Mechanism subsection> 3

**Software Architecture Document**

# **Introduction**

## **Scope**

This document is applied to the developing bookstore website.

## **Definitions, Acronyms, and Abbreviations**

## **References**

## **Overview**

The rest of this document contains a description of the whole system which is illustrated in diagrams or written form that will loop through architecture views.

# **Architectural Representation**

## This section details using the views defined in the 5 models to build the architecture: use case view, process view, deployment view, and implementation view. The following are the views used to document the smart banking application:

## 2.1. Use case view

## Audience:

## All the stakeholders of the system, including the end-user.

## Area:

## Describes a number of use examples and/or scenarios that illustrate some of the most crucial and fundamental aspects of the system. Describe the actors and cases in the system. Along with the basic workflow, the article also covers exception cases, exception outputs, and other relevant use cases.

## 2.2. Logical view

## Audience:

## Designers, programmers, testing staff

## Area:

## Functional requirements, object hierarchy, system layers

## Describes the object model's design. The subsystems and their interactions are also described.

## 2.3. Process view

## Audience:

## Integrators, programmers

## Area:

## Non-functional requirement

## Describes the concurrency and synchronization features of the design. Define the system's behavior at runtime.

## 2.4. Deployment view

## Audience:

## Programmers, code testers

## Area:

## Software component: Describes the modules and subsystem divisions of the system.

## 2.5. Implementation view

## Audience:

## Database administrators, system engineers, development managers

## Area:

## Persistence: Describes the data model's architecturally significant persistent element. The mapping of software to hardware is described, as well as the distributed features of the systems.

## 

## 

# **Architectural Goals and Constraints**

* Security: All passwords provided by users will be encoded for the case of leaking database.
* Privacy: Other users’ information in the database will be secured to not be used for any other purposes.
* Distribution: due to being deployed on firebase, the website can be accessed any where and any device provided internet connection
* Reusability: the software architecture is designed for the purpose of reusability when the domain is separated from application layer and infrastructure, which means with any database and/or protocol changes don’t affect domain of software
* Development tools:

Coding: intellij, visual studio code

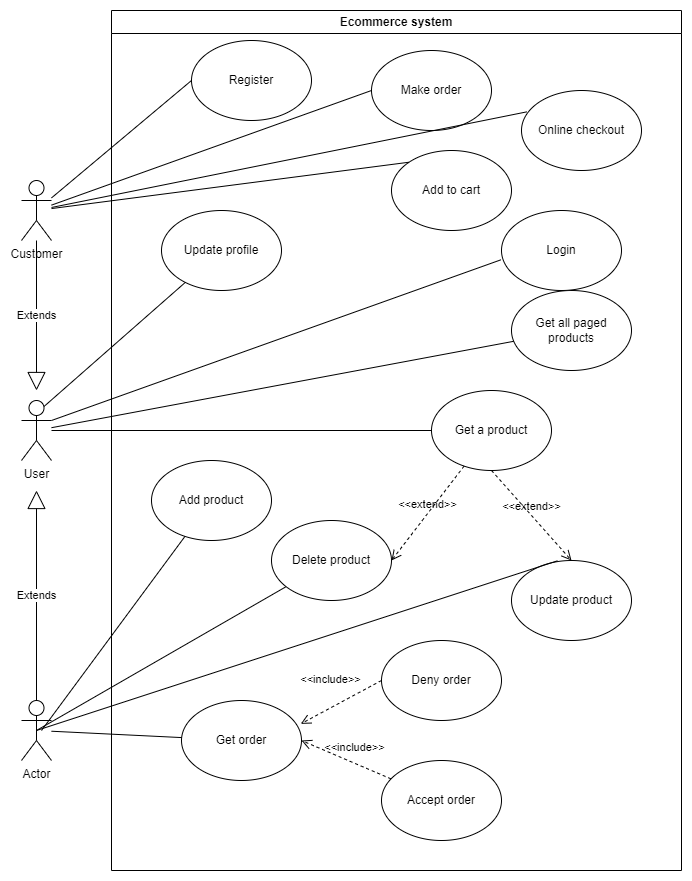
Testing: postman

Designing: figma, draw.io

Managing: trello

Storing: github, google drive

# **Use-Case View**

**

# **Logical View**

## **Overview**

The system is divided into three main subsystems: FE, BE and Network. Additionally, as an executed system, it has the interaction from the user via Browser to access FE and a Database for storing all the data of the system, which is used by Application.

5.1.1 FE

This is the front-end part of the system which contains all the views of the system. The purpose of this subsystem is to display UI to users and perform the functions that do not need to access the database.

5.1.2 BE

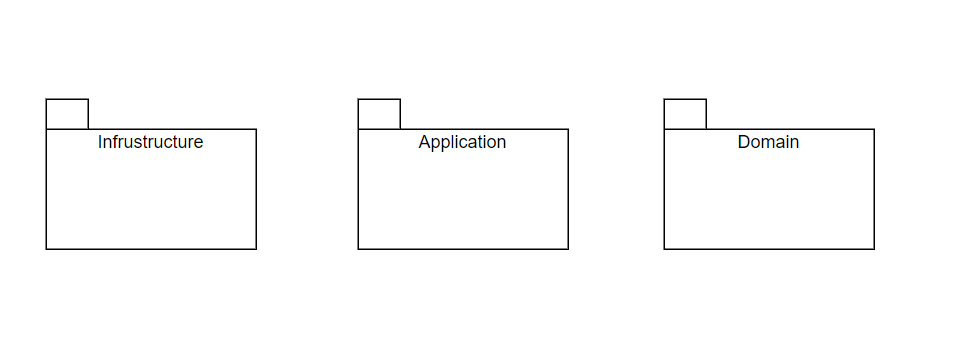
This is the back-end part of the system which performs the functions that need to work with the database. It can be divided into Controller, Repository and Domain packages for specific functions

5.1.3 Network

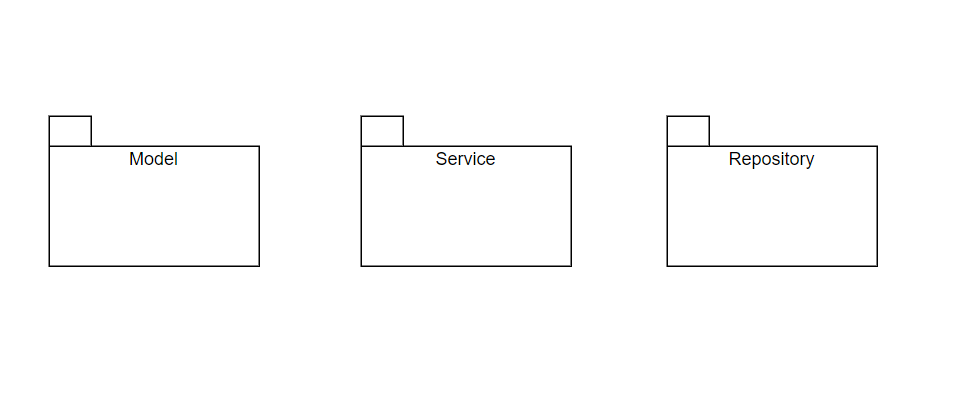
This is the connection to connect Presentation and Application with the user via Browser. The user’s experience, whether good or not, partly depends on this subsystem. It conforms to the TCP/IP model, which consists of four layers: Transport layer, Application layer, Internet layer and Network access layer.

## **Architecturally Significant Design Packages**

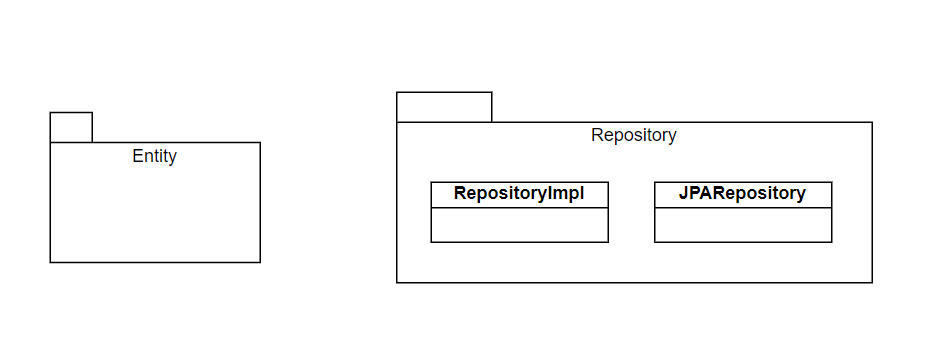
The BE of the system is divided into 3 parts according to DDD (Domain Driven Design). Domain layer contains the most important logic of the whole project. Infrastructure consists of connection to Database. Application creates api to connect to FE. They are all separated into components.



In Domain, model represents an object and also entity, repository contains only the interface of repository implementation which is held in infrastructure and service for doing logic to model.



Infrastructure containing implementation of a repository, which connects to a database using ORM. The purpose of separating this part from the domain is because with database changes such as from MySQL to PostgreSQL or Mongo, we don’t need to modify the domain.

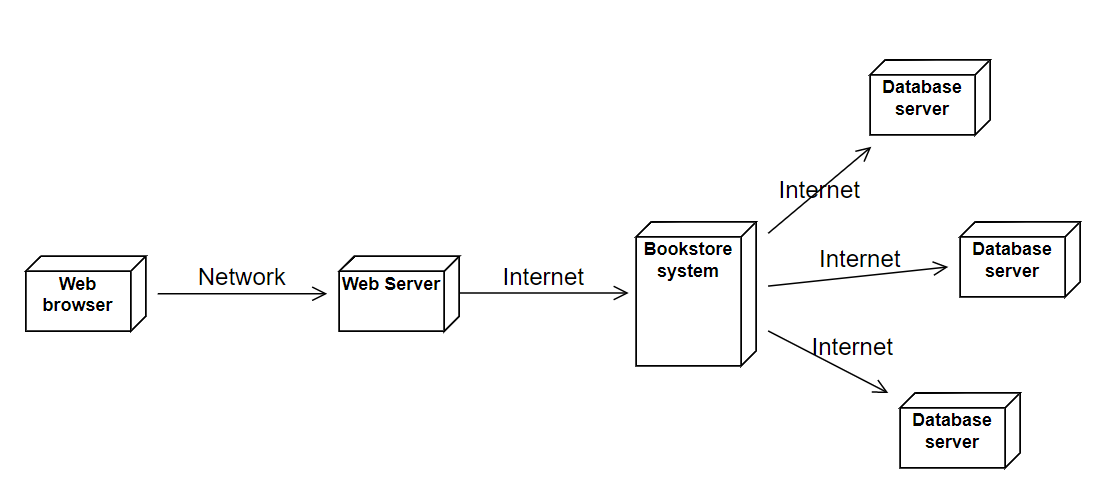


Application contains controllers which use services in the domain and create rest api to connect to FE. The purpose of separating this part from the domain is because of protocol changes.

# **Process View**

# **Deployment View**

With any mobile device capable of running a web browser with the internet provided, when the user calls any api to the web server, which has the bookstore system deployed on, then the user will get a response from the system right after the bookstore system executed (may use Database server with the internet connection). The database will be hosted in some other hosting space.

**

# **Implementation View**

There are no considerable changes when implementing the system. Then, this view will be ignored in this document.

# **Data View**

# **Size and Performance**

The bookstore is going to contain about 200-400 products and support 10000 concurrent users.