Prudentia Book Store
Architecture Notebook

1. Purpose

The main purpose of this document is the understanding general design and the architectural structures with component interactions. This document describes the philosophy, decisions, constraints and the general view of project.

2. Architectural goals and philosophy

•The goal of this program is to provide a well-designed architecture that can be used easily by the employees who are in bookstore and the customers.
• The system is designed as the web application.
• The application approach is keep it simple
• The web services that we use for manipulate use case written in JAVA with MYSQL front-end part coded by ReactJS
• The web services are published on a platform called Heroku
• The system will respond to the member in less than two seconds from the time of the requested submittal.

3. Assumptions and dependencies

Users can buy books, via fast and easily understand way.
The application will run on the web without any break time
The storage capacity of the system not have to so big.
All information will be stored in the database. Passwords are encrypted.
The system will not be associated with any hardware.
This system requires the internet.

4. Architecturally significant requirements

• The system is a web application.
• Users can see the products without log in to the system but if they buy a book they have to be logged in
• For security reasons, every password encrypted with hashing algorithm
• The system keep track users operations like login, logout, buy something exc.
• The system have to response http request within a maximum of 10 seconds
• If a request perform an error that request info's about that will be stored

5. Decisions, constraints, and justifications

- The system is a website only needed to work that is internet connection
- There is no specific operating system limit.
- The system is flexible, adding and removing new things is easy.
- The system developed using JAVA, MYSQL and ReactJS
- Manipulating categories, products, users and exc. uses cases can only performing by admin.

6. Architectural Mechanisms

Stands for "Model-View-Controller." MVC is an application design model comprised of three interconnected parts. They include the model (data), the view (user interface), and the controller (processes that handle input).

The MVC model or "pattern" is commonly used for developing modern user interfaces. It is providing the fundamental pieces for designing a program for desktop or mobile, as well as web applications. It works well with object-oriented programming, since the different models, views, and controllers can be treated as objects and reused within an application.

Model

A model is data used by a program. This may be a database, file, or a simple object, such as an icon or a character in a video game.

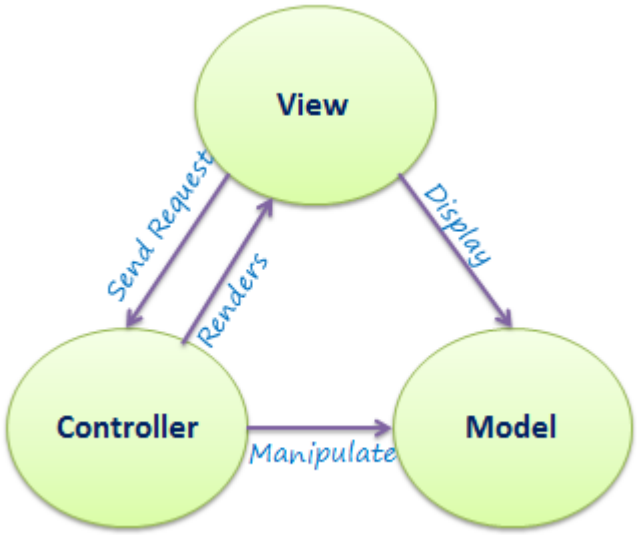
View

A view is the means of displaying objects within an application. Examples include displaying a window or buttons or text within a window. It includes anything that the user can see.

Controller

A controller updates both models and views. It accepts input and performs the corresponding update. For example, a controller can update a model by changing the attributes of a character in a video game. It may modify the view by displaying the updated character in the game.

MVC Architecture Pattern



Pros of MVC Pattern

1. Faster development process:
2. Ability to provide multiple views:
3. Support for asynchronous technique:
4. Modification does not affect the entire model:
5. MVC model returns the data without formatting:

7. Key abstractions

Registered Users: The persons that are using bookstore website, they can perform buying book, looking book etc. uses cases

Non-Registered Users: The persons that are using bookstore but not registered, as an example they cannot perform buying book uses case

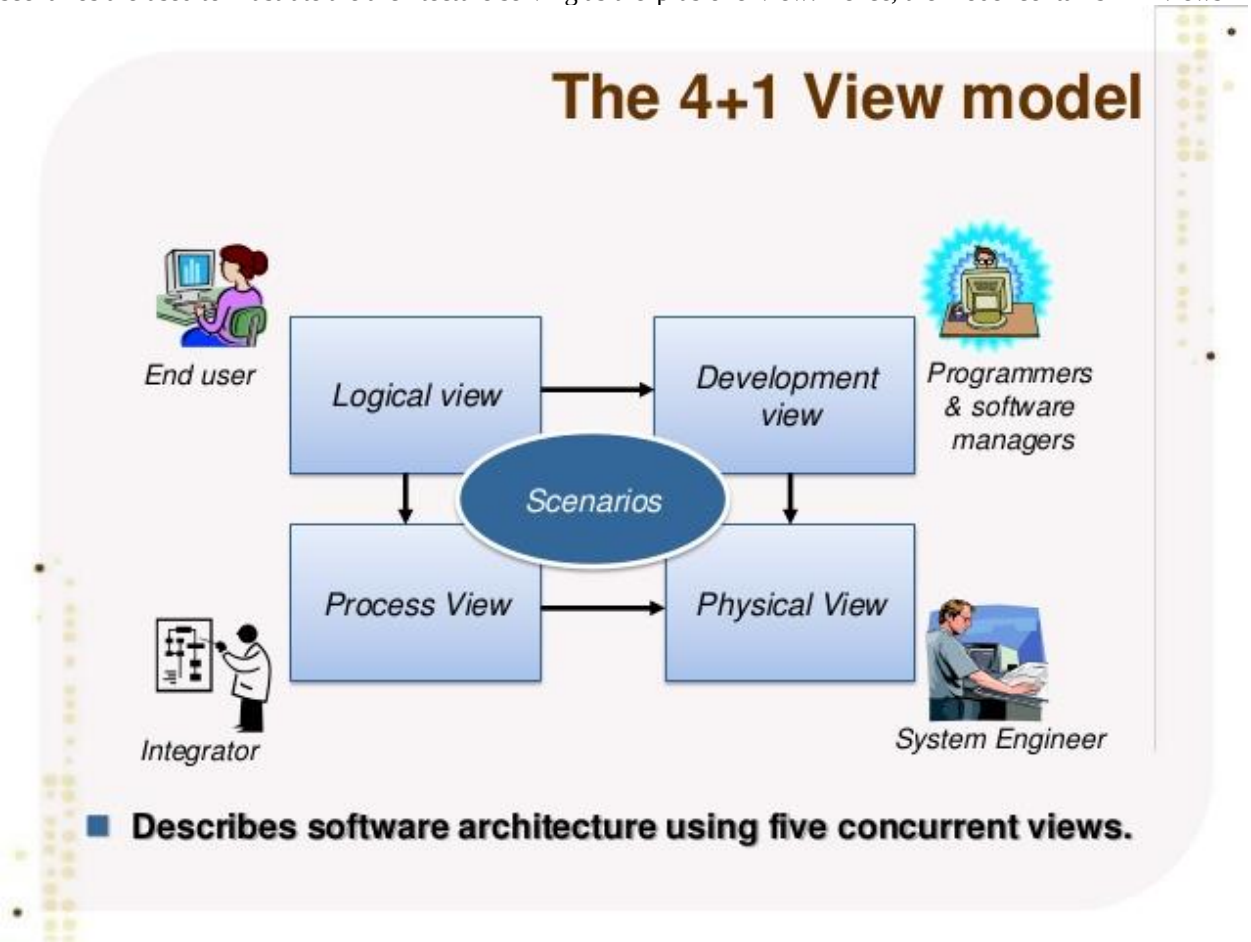
Admin: The person that manage categories, products, users, orders etc.

Interfaces: Interaction point between user and system.

Database: The abstraction for storing book information and user information.

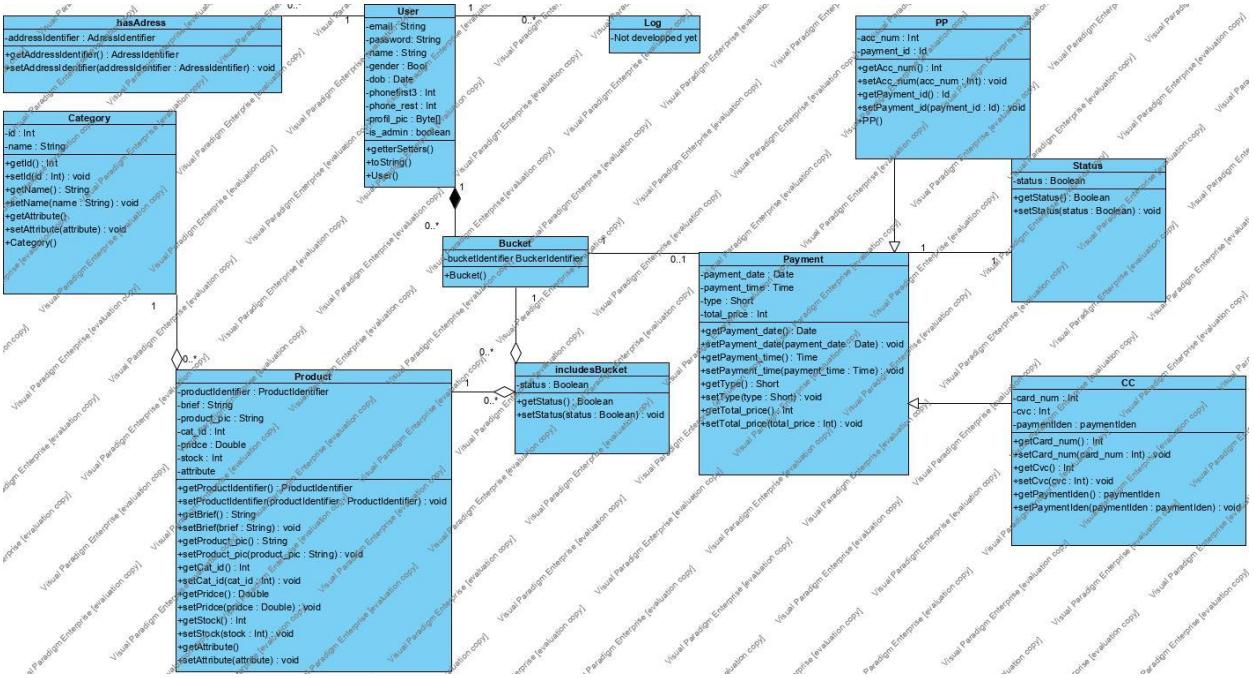
8. Layers or architectural framework

In our project we use 4+1 view model, 4+1 is a view model used for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views". The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers, system engineer, and project managers. The four views of the model are logical, development, process and physical view. In addition, selected use cases or scenarios are used to illustrate the architecture serving as the 'plus one' view. Hence, the model contains 4+1 views



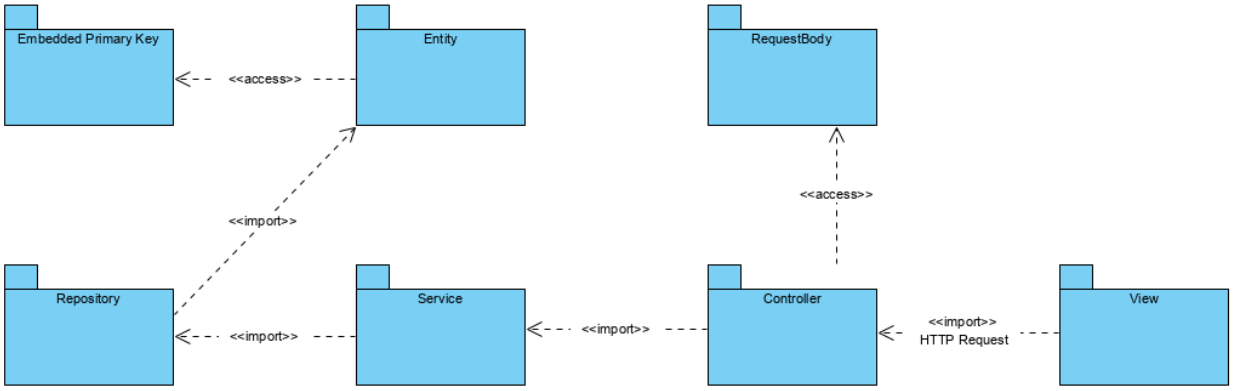
9. Architectural views

Logical View: The purpose of the logical view is to specify the functional requirements of the system. It's relevance is to developers. This contains information about the various parts of the system. In UML, logical view is modeled using class, object and composite structure diagrams

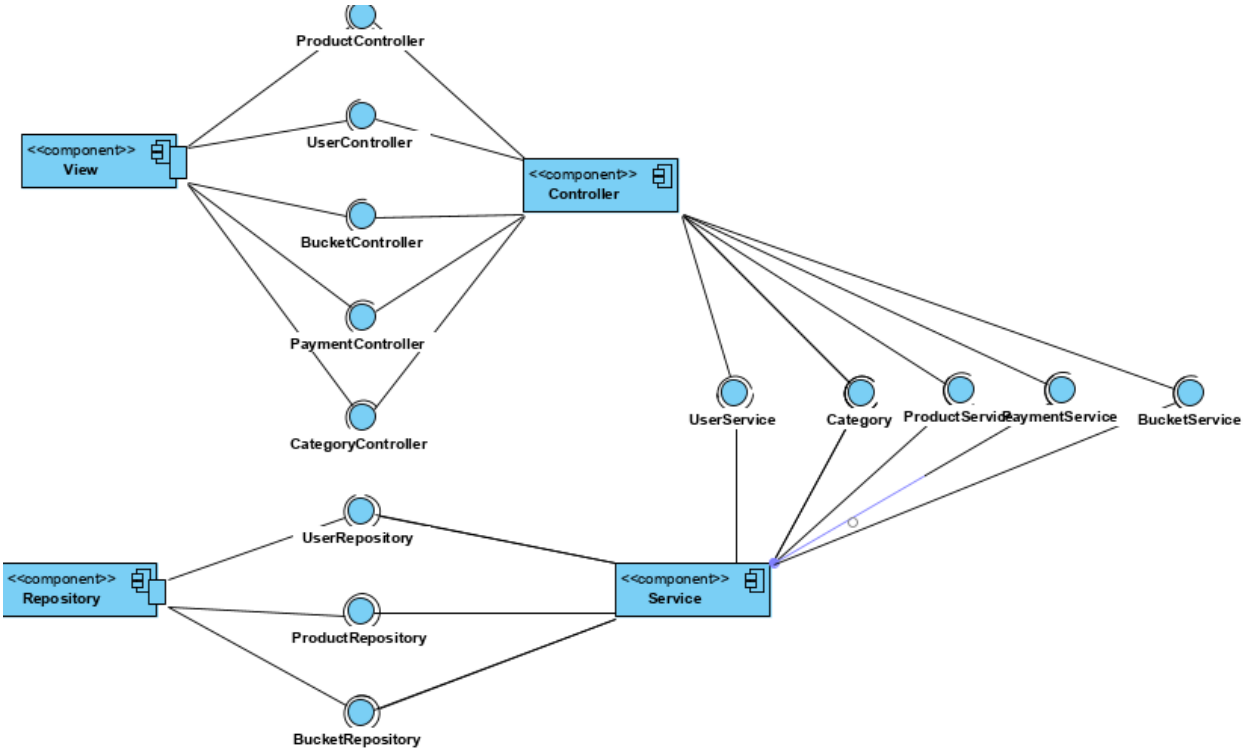


Class Diagram

Implementation View: The organization of software modules such as source code, data files, executable files in the project development environment. It focuses more on the subsystems. In UML, Package and Component diagrams are used to model the implementation view.

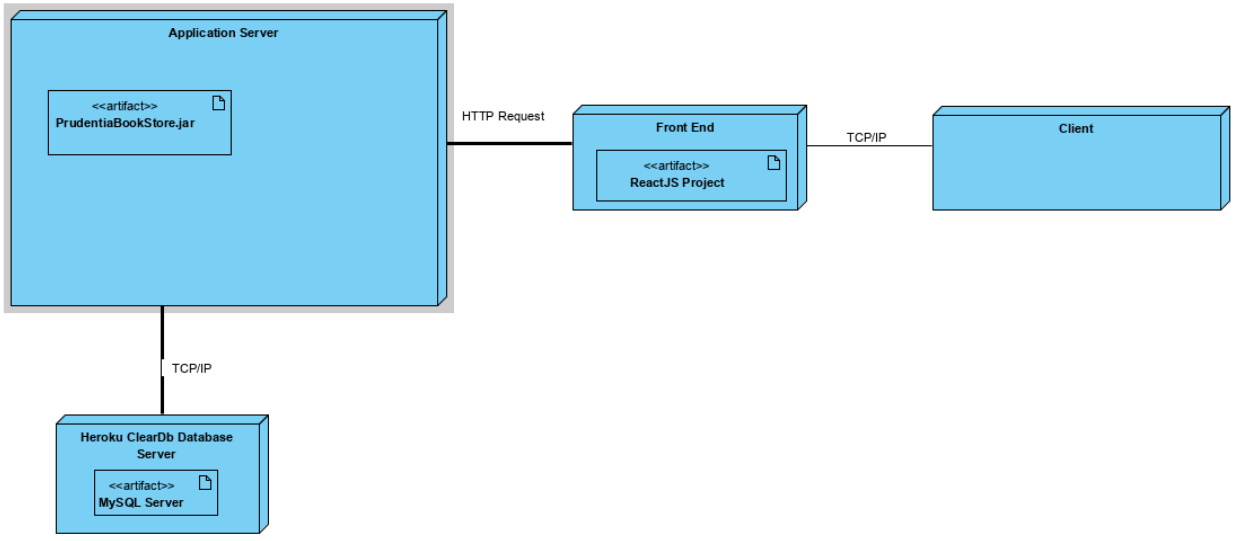


Package Diagram



Component Diagram

Physical view: Deployment diagram shows how the system will look in the physical layer with software components and their hardware parts. It shows relations between them. Deployment Diagram is as below:



Deployment Diagram