DEPARTMENT OF COMPUTER ENGINEERING
BBM384 Online Book Management

BBM384 Software Engineering Laboratory
ONLINE BOOK MANAGEMENT SYSTEM
Architectural Notebook

BOOKIFY

Ismet Okatar 21727542
Atakan Ayyıldız 21526681
Göktuğ CANDEMİR 21627064
Ege ÇINAR 21627136
Muhammed Aydoğan 21726952

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. It serves as a communication medium between the software architect and other project team members regarding architecturally significant decisions which have been made on the project.

2. Architectural goals and philosophy

The main architectural goal is to provide a well – defined architecture that facilitate maintenance and future development. Future upgrades on underlying tools should be considered in the design process. The architecture also must provide ease of use to shop staff and customers easily so that there must not be any problem, during their interaction with each other.While the architecture is being developed, all requirements in former documents will be taken into consideration in order to provide a coherent and good user experience.

The online book shopping system will be a web application and will not be dependent on any hardware. It will be the responsive and cross-browser application, so that will work in any modern browsers. It will be a Java Web application and will use PostgreSQL as database system management.

3. Assumptions and dependencies

Assumptions,

- The backend service will be implemented with Java by using MVC Architecture.
- The frontend will be implemented with Javascript by using React.js Framework
- Backend service has to support CRUD operations which is needed for the frontend service.
- Users might have easily searched for their wanted products and bought them in a secure and reliable method.
- The UI has to be simple enough to be understandable by the users.

Dependencies

- Backend has to have enough processing power and Network bandwidth to communicate with frontend service
- Backend has to save every relevant information to the Database without data loss.

- Java Spring Framework and PostgreSQL have to be installed properly.

4. Architecturally significant requirements

There are some requirements for developing this system listed below:

1. The system must be executing on the Java Spring Boot platform.

2. The system will be a Web application.

3. The system's maximum response time to request must be less than 5 seconds.

4. The system provides maintenance as easy as possible.

5. The system provides the user to purchase safely

6. Customers should log in to the system for order products.

7. The system must be work synchronize with the external system

8. Admin should log in the system for management and control the entire system.

9. Salespeople also login system for add/delete/update products.

5. Decisions, constraints, and justifications

- GitHub is used to develop the project in order to make development easier.
- Branches are used in GitHub. Every developer uses his own branch to add changes to the project.
- The same IDE is used across all team members to be more compatible.
- Open source technologies are used such as PostgreSQL.

6. Architectural Mechanisms

Architectural mechanisms represent key technical concepts that will be standardized across the solution. They are refined during the project through three states, represented by the three categories of Architectural Mechanisms:

Architectural Mechanism 1: Analysis

Purpose: Analysis mechanisms are the initial state for an architectural mechanism. They allow the team to focus on understanding the requirements without getting distracted by the specifics of a complex implementation. Security, persistence and legacy interface are some examples of these.

Attributes:

1. Persistence,reliability is to survive a crash of the process, update frequency, how long does the object typically need to be kept in the system is analysed.
2. Communication, flow control, buffering, synchronicity, latency.

Once the list of analysis mechanisms has been defined it can be prioritized and the mechanisms refined in line with iteration objectives. It is not necessary to develop the entire set of architecture mechanisms into working software in a single pass. It is often more sensible to develop only those mechanisms required to support the functionality to be delivered in the current iteration.

Architectural Mechanism 2: Design

Purpose: Refining an analysis mechanism into a concrete technology (for example, an RDBMS can be used for persistence). The purpose of this category is to guide precise product or technology selection. The attributes captured against the corresponding analysis mechanisms should be used as criteria to prove the validity of the decisions.

PostgreSQL is used as a design mechanism (RDBMS). Also each work package in the Bookify application involved will independently provide the facilities to recover systems.

Architectural Mechanism 3: Implementation

The Implementation mechanism is the part where the actual implementation of the architectural mechanism is realized. In this part our software Architect and other Team members will be involved as Software developers and each team member will contribute to the implementation part of our Bookify application by working on various parts as frontend, backend or database. The use case diagram and entity diagram which has been realized before will be used in our development process.

7. Key abstractions

The different components shall be loosely coupled - all communication shall be done asynchronously and no requests or data shall be lost whenever one or more of these components fails. We are encouraged to maximize functional cohesion. Each module is strongly interconnected and mostly does not get details of other modules.

User interaction and business process diagrams are expressed in the Architectural Views section.

There are 3 key abstractions in this project:

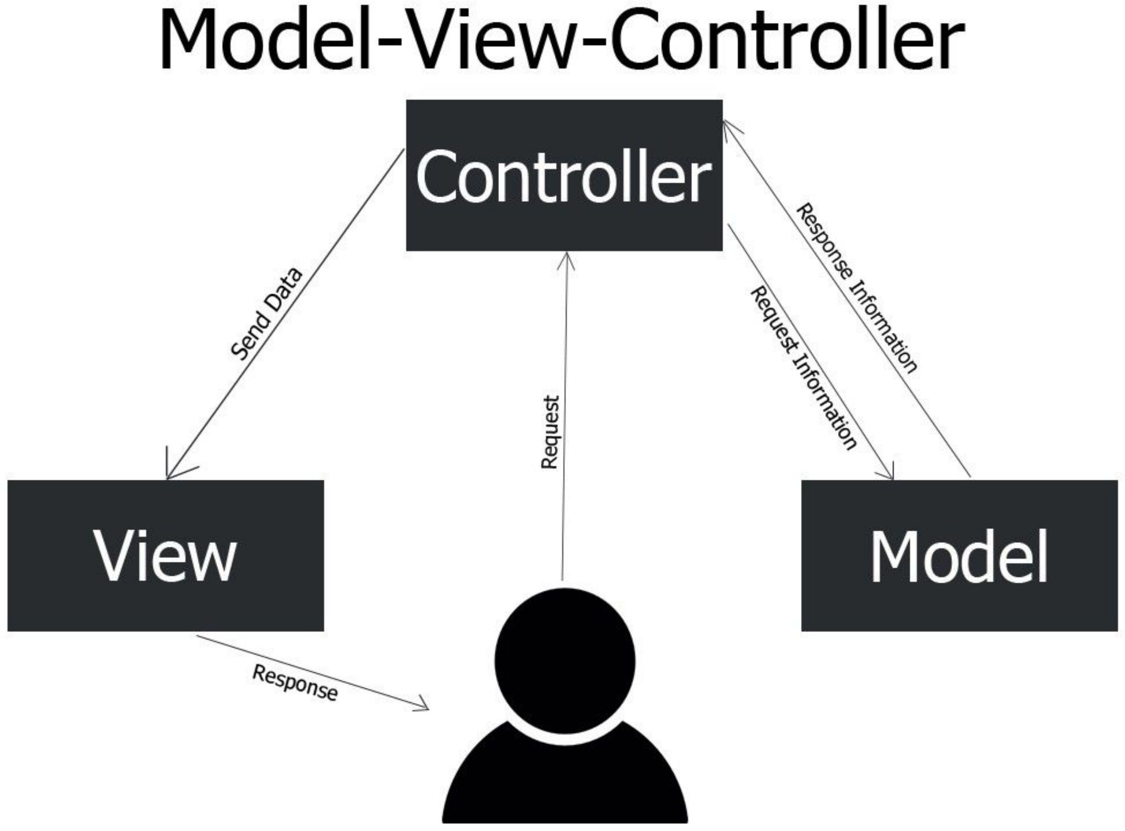
- Customer: Each customer has, id, username, mail, orders history and encrypted password
- Admin: Admin has, id, username, mail, and encrypted password
- Books: Each book has, id, name, price, stock, categories, page_count, author, sale_count and an image.

8. Layers or architectural framework

In this project we are using the Model-View-Controller pattern.

Model-View-Controller is a modern and most frequently used industry-standard architecture pattern that divides application into three main parts. A big benefit of this pattern is preventing the system from getting a little complicated. Here is how it works:

Users can request information only from the controller. Then the controller communicates with the model if it’s needed. Then sends data to view. Only view responses to users directly. The most important case about this pattern is that the model and view cannot communicate with each other directly.



8.1 Model Architecture

The Model has representations of data structures of the key abstractions and logical components of the software. The model object acts as a template in the database. In our project the main abstractions such as the Book have a model object. Also secondary abstractions such as the Writer have also a model in order to make changes easier in such objects.

8.2 View Architecture

The view part is the UI related part of the Model View Controller Architecture. View is the part which sends to the user the relevant information which will be shown by displaying. After the Controller and Model the wanted Information from the database will be selected and it will be displayed by using the View part.

We have two main Views in our Bookify project. One is for the Customer module which includes the Registered Customer and Unregistered Customer. Those two

Customer modules use our Customer View in our application. We also have an Admin View which is implemented for realizing the actions which will be made by the admin.

Each action made in the View will be sent to the Controller and the relevant information returned from the Controller is shown in the view again to the user which makes the action.

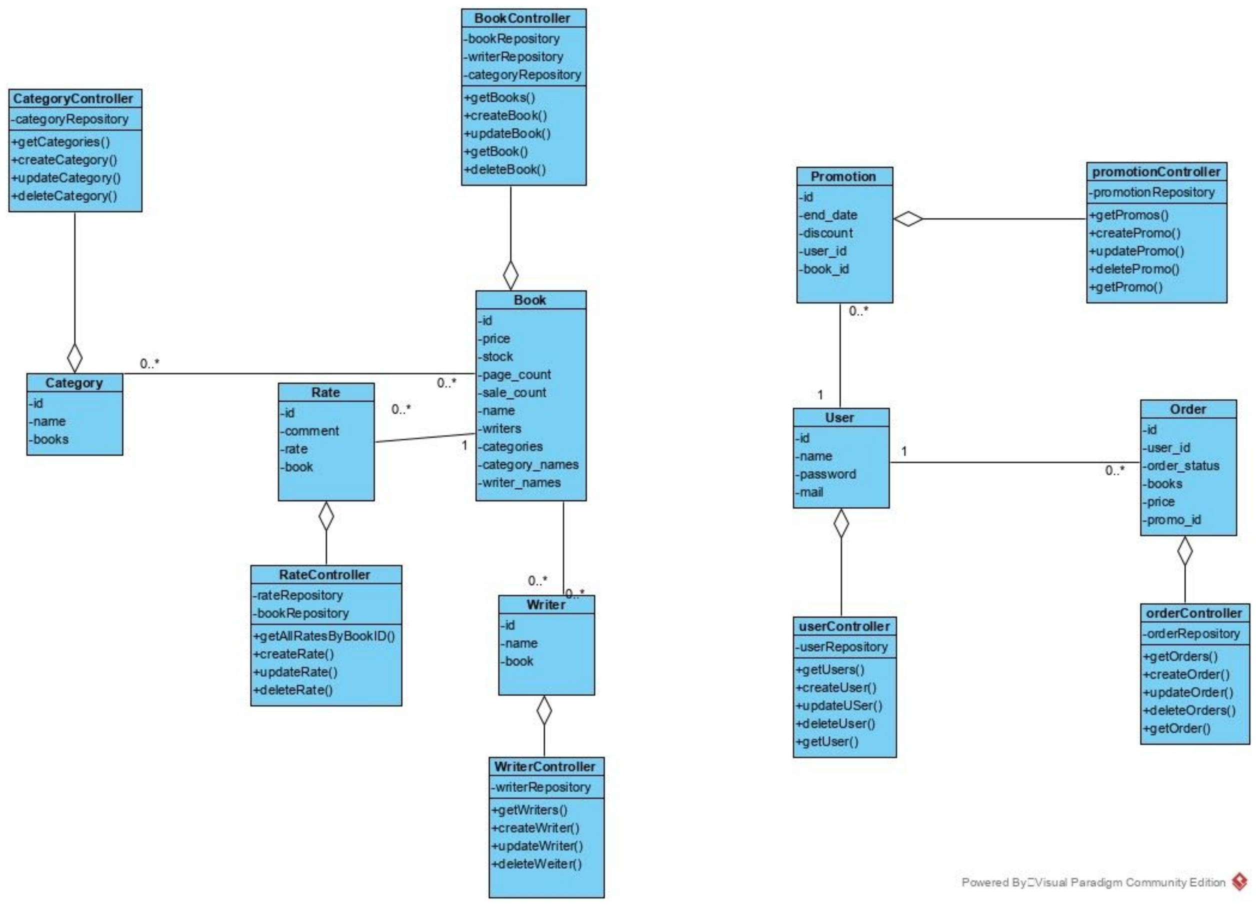
8.3 Controller Architecture

The Controller part of the Model View Architecture is the part which the Controller listens to the requests which can be sent by the user. When a user sends a request to the Controller, the controller asks for the Model part and if it is necessary then sends the relevant information to the View part. This project's controller consists of Book Controller, Rate Controller, User Controller, Writer Controller, Category Controller, Promotion Controller and Order Controller.

9. Architectural views

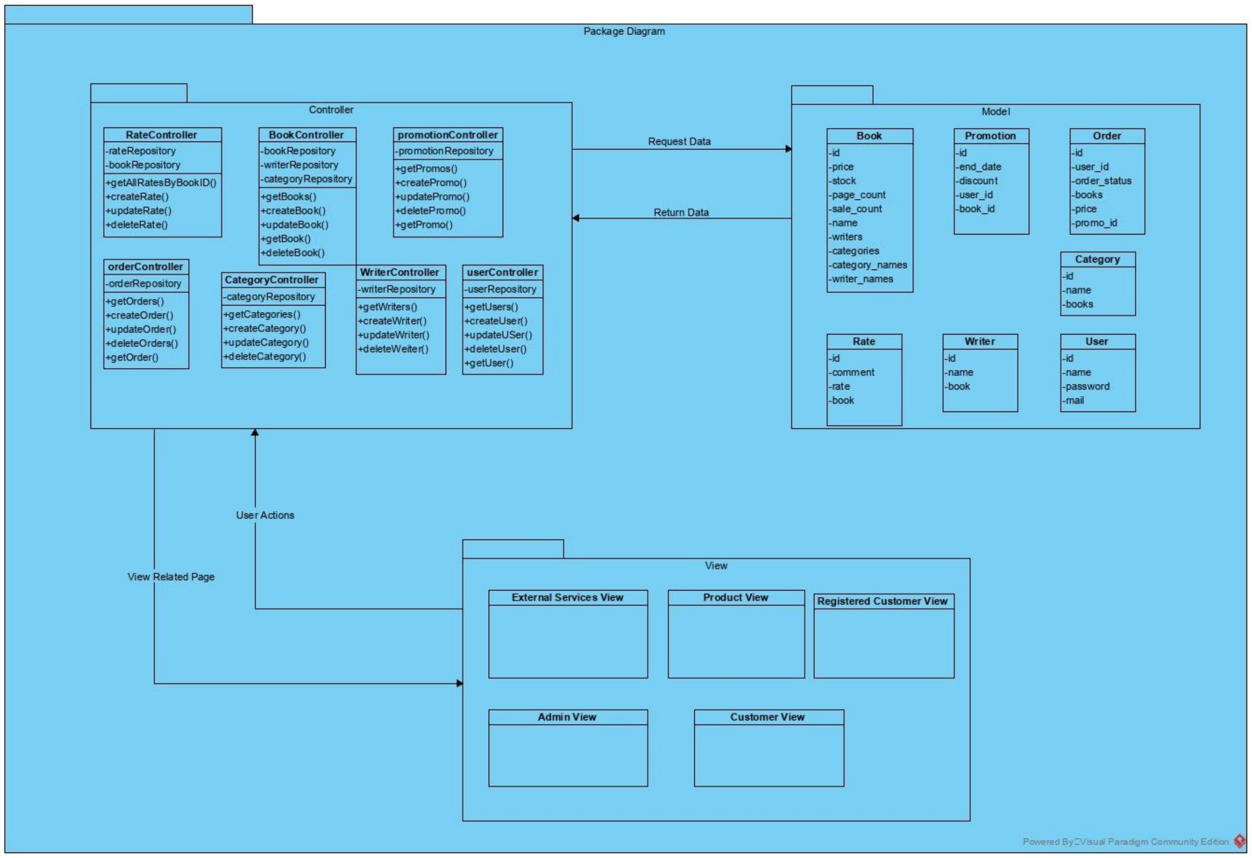
Recommended views

- **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.
  - **Class Diagram:**

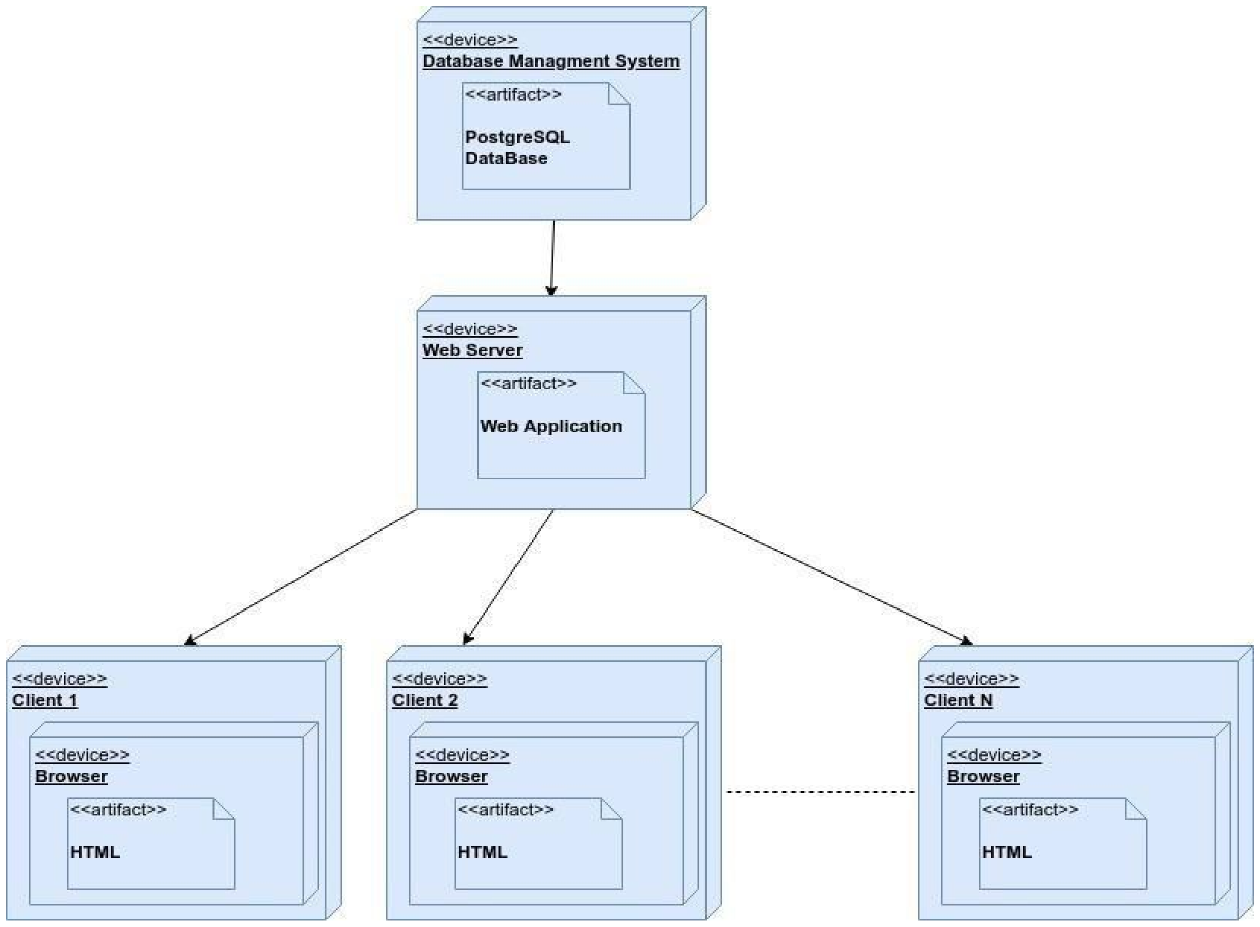


- **Operational:** Describes the physical nodes of the system and the processes, threads, and components that run on those physical nodes. This view isn't necessary if the system runs in a single process and thread.

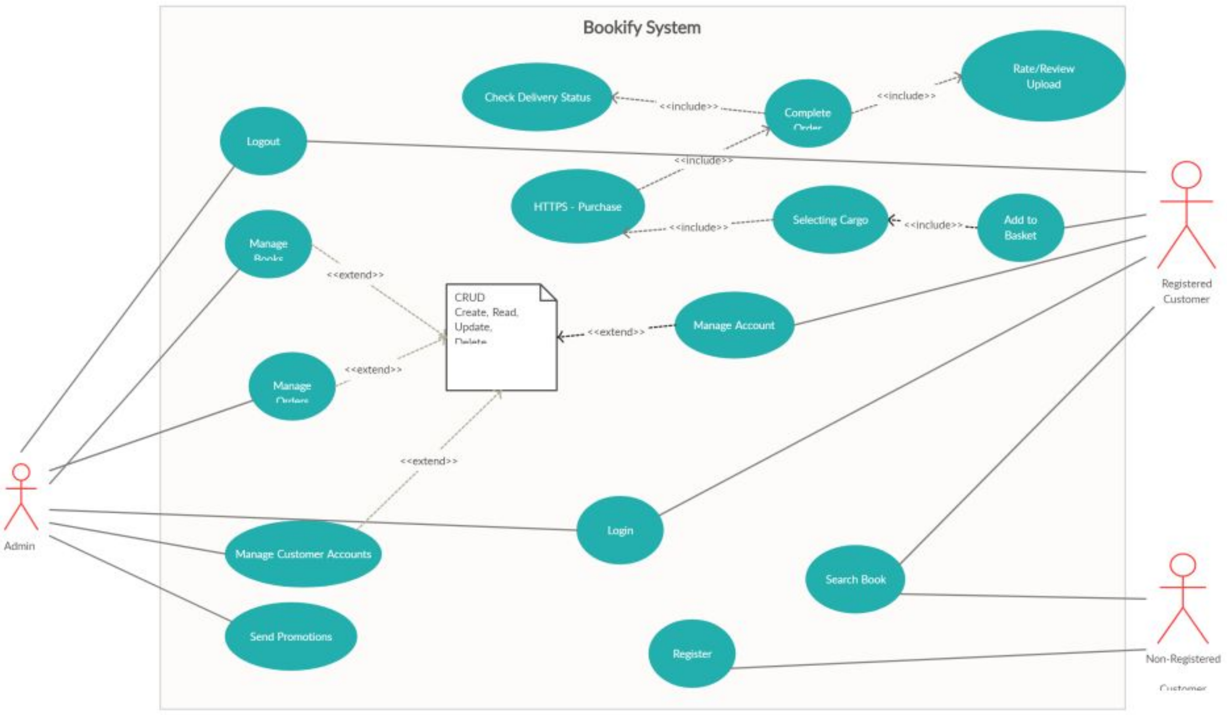
- **Package Diagram:**



- **Deployment Diagram:**



- **Use case:** A list or diagram of the use cases that contain architecturally significant requirements.


Bookify System
Check Delivery Status
<<include>>
Complete Order
<<include>>
Rate/Review Upload
Logout
<<include>>
HTTPS - Purchase
<<include>>
Selecting Cargo
<<include>>
Add to Basket
Manage Books
<<extend>>
CRUD Create, Read, Update, Delete
<<extend>>
Manage Account
Manage Orders
<<extend>>
<<extend>>
Registered Customer
Admin
Login
Manage Customer Accounts
Search Book
Send Promotions
Register
Non-Registered Customer