HACETTEPE UNIVERSITY DEPARTMENT OF COMPUTER ENGINEERING
BBM487: SOFTWARE ENGINEERING LABORATORY

ARCHITECTURE NOTEBOOK

GROUP - 6

CEMAL ÜNAL **21328538**

İREM KOCABAŞ **21328188**

METİN ARSLANTÜRK **21426625**

Library Borrowing System
Architectural Notebook

1. Purpose

The main purpose of this document is the design of a general perspective of the system architectural structures, to understand component interactions and modeling these connections make it easier for third persons, who will want to use or review our system.

In favor of this document are any chart and paragraphs more clear to architectural specifications of the system. This document describes the philosophy, the decisions, the constraints of the system, justifications, essential elements and other cross-cutting aspects of the system, the forms of design and implementation.

Also we are explaining the interaction with other systems, maintenance level and skills of the development team who generated the system.

2. Architectural goals and philosophy

This system generate purpose meets the library usage requirements, same as requirements and complexity of this system in general all over the world. This program philosophy developed a program in the easiest, fastest and most reliable way. This system will not have complex relationships with other systems or this system do not have to be so strong efficient skills.

Maintenance system requirements are so restricted, because we provided all of the system requirements and designed architecture of this system in accordance with these requirements, so major changes may force easy-to-change flexibility but generally our architecture, our architecture if required for maintenance.

This system creates a usable program to all systems which have internet connection because our program is a web-based program. Our system has many numbers unexpected condition handle style, all handles for generated using reliable and secure use of the system and for the security of the system.

3. Assumptions and dependencies

In our Web-based library loaning system; all the processes are created depending on the user requirements from user interfaces, so that the system will correct and consistent requests of the user, if it not system may not be enough useful than expected and generated.

This system requires only a web-browser in a internet connected system and enough connection quality if they are not be met the system work might not be done or be wrong.

We are trying the using abilities of members on correct parts,someone has better PHP knowledge and she doing the back end works.The other one making the basic architectural managements and these two works continuing concurrently,also all members efforting about the reporting and visualization works.

4. Architecturally significant requirements

We determined the architecturally significant requirements for developing this system listed below:

1) The system must work in a web-browser which in a system with internet connection
2) The system should be able to handle CRUD operations.
3) This system's maximum response time not should be long.
4) While an error handling,program should not crash,Should log and continue work.
5) The system should always display an interface to the user for correct usage.
6) All the processes must be choosen with mouse clicks to the interface.
7) The system should be able to easy to maintanance an flexible as much as possible.

5. Decisions, constraints, and justifications

- The system will be implemented using the Model-View-Controller pattern. Used to separate the view from the application logic and to facilitate maintainability.
- For easy manage and usage, system has one Controller which works on both items(books) and users(librarian,user).One point management made all controles easy for developer.
- This system can handle the concurrent usages.
- The system should be designed as much flexible as possible; because after publishing some maintainance needs could be appear ,so the system should be flexible much as it can.
- The system has doesn't require any OS specs,it is web-based so independent from OS.

6. Architectural Mechanisms

We have designed an Architectural Mechanism because of reducing the workload, getting help for planning, and making the system more easier to maintain.

We have used each 3 states of Architectural Mechanism which are called analysis, design and implementation. These categories are summarized in the table that fallows :

State	Description
Analysis	**1)** We have used Inheritance which is one of the important member of Object Oriented Design, because of getting control for Student and Librarian classes in one class as called Person. So, that design helps us to control more easier these two classes. **2)** We have used a graphic user interface design to make an understandable application for users
Design	**1)** We used **CSS + HTML5** to make an graphic user interface **2)** We worked in basic text editor.
Implementation	We have implemented a personal class which is ancestor of student and librarian classes. And we have made a book class to hold the id, name, and author attributes of book objects. Also, a Main class to initialize all operations, and a Manager class to control all functions.

7. Key abstractions

Librarians: Librarians can add, delete, and update books. They also can add a student to the system and remove him/her from system.

User: Users can ask for getting a book from system or return it. They have password for authentication.

Book: Is the abstraction of book class that holds information about the book.

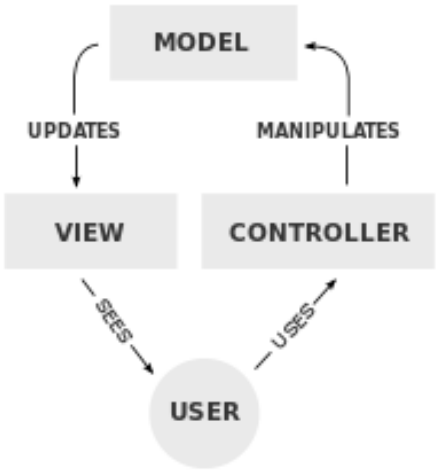
8. Layers or architectural framework

We have used Model-View-Controller architectural pattern while implementing our pro. Since, we have a lot data to hold as book, user, and librarian information. And we need to separate the data (view) from the functions of the application.

Model View Controller or MVC, is a software design pattern for developing web applications. MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. The pattern runs in that way :

1) Controller receives all requests for the application
2) Controller works with the Model to prepare any data needed by the View
3) The View then uses the data prepared by the Controller to generate a final presentable response

The MVC abstraction can be graphically represented as follows:



MODEL	The system uses two model for the person, including librarian and library member; and one for the book. Each model is responsible for the interface between the entity and the stand-alone application, providing functions to create, update and manipulate the different items. Thus, the model is able to provide information to the view
VIEW	The view is the representation of the current information in the stand-alone application. Each view will be shown to the librarian, supplying information to him/her and enabling him/her to use the system and interact with it. Each view will be supported by the controller.
CONTROLLER	The controller is responsible for responding to user (librarian) input and perform interactions on the data (library member, book) model objects. And, it also need to be the interface between the view and the system. The controller receives the input, it validates the input and then performs the business operation that modifies the state of the data model.

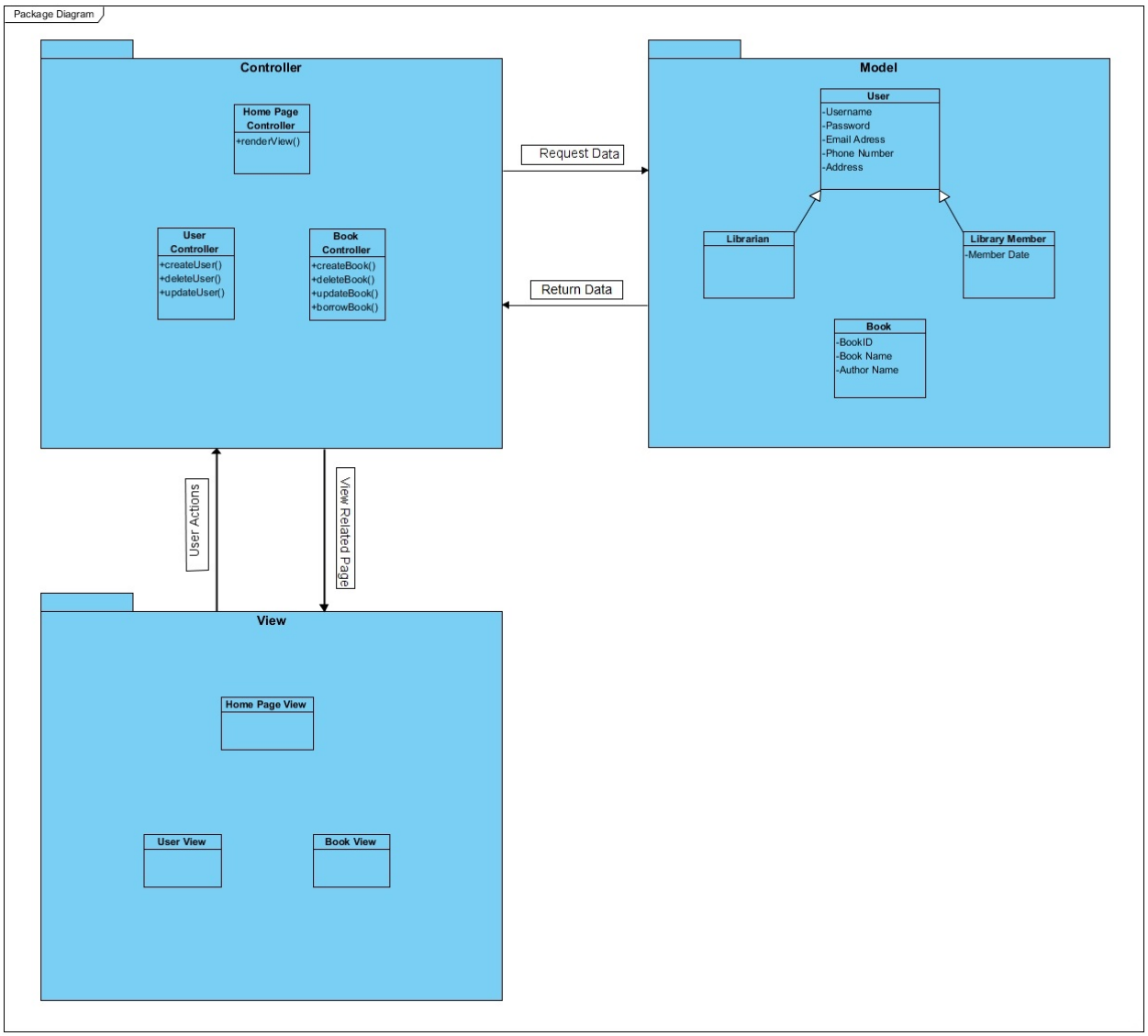
9. Architectural views

To choose the appropriate views, identify the stakeholders who depend on the software architecture documentation and the information they need.

Logical View:

The logical view is concerned with the functionality that the system provides to end-users. It is considerably simplified to take into account only the items that are architecturally significant. UML Diagrams used to represent the logical view include **Package diagram.** We will analyze just Class diagram, and Package diagram in this assignment.

- Package Diagram:

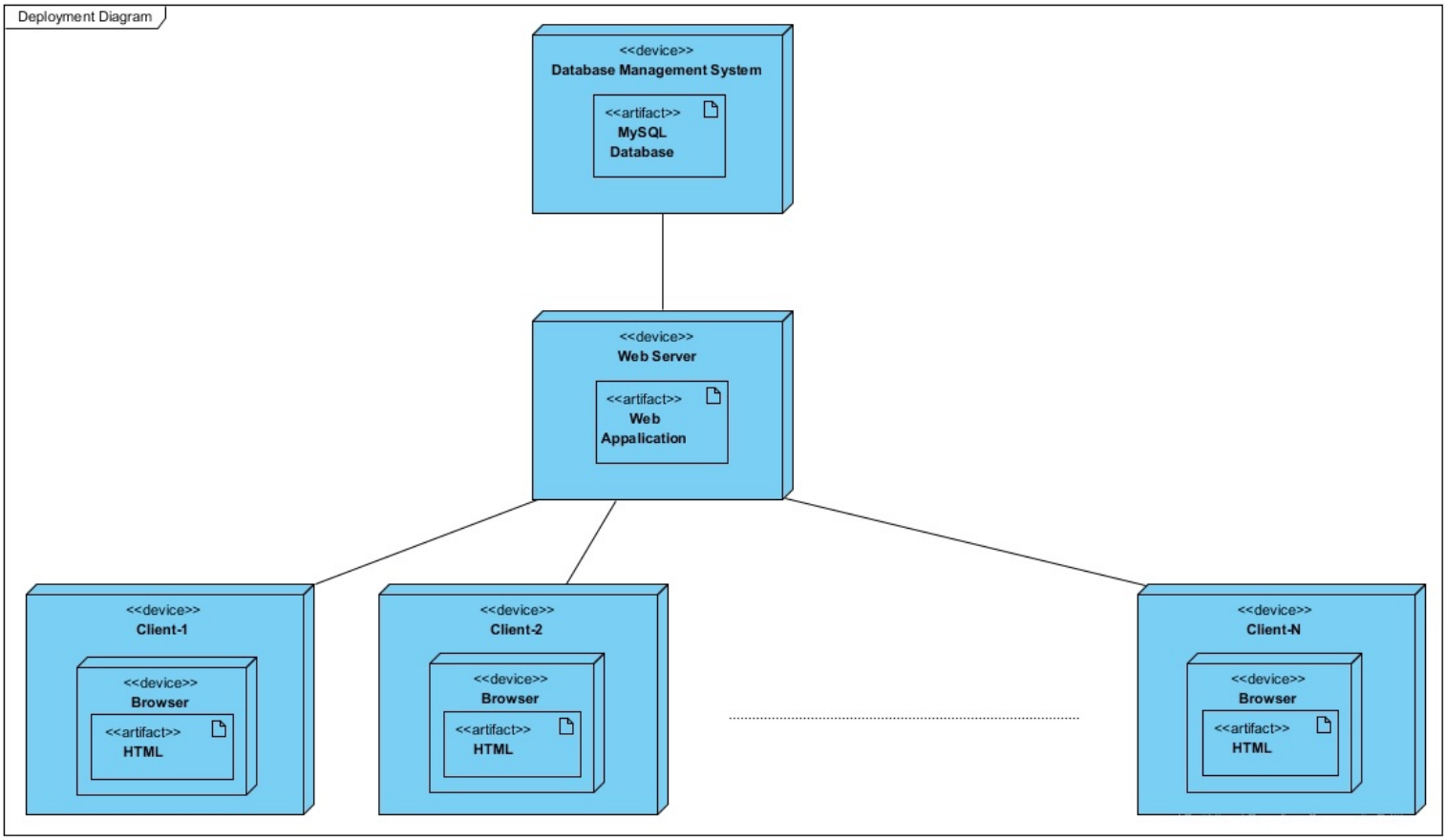


- Operational View:

Describes how development artifacts are organized in the file system. And also, it focuses on how the system is built. UML Diagrams used to represent the logical view include Deployment diagram, and Component Diagram.

1)Deployment Diagram:

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.



2) Component Diagram:

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Component diagrams represent a set of components such as classes, interfaces and collaborations. They also show us the relationships between these components. In other words; component diagrams explain us the implementation view of a system. So, component diagrams are used to visualize the implementation.