Analysis and Design Document

Student: Grigor Sonia Eufrosina Maria
Group: 30233

Revision History

Date	Version	Description	Author
24/May/19	1.0	Write first two capitols of this document	Grigor Sonia

Table of Contents

I. Project Specification

Software Design Project is a web application named "Tell your friend".

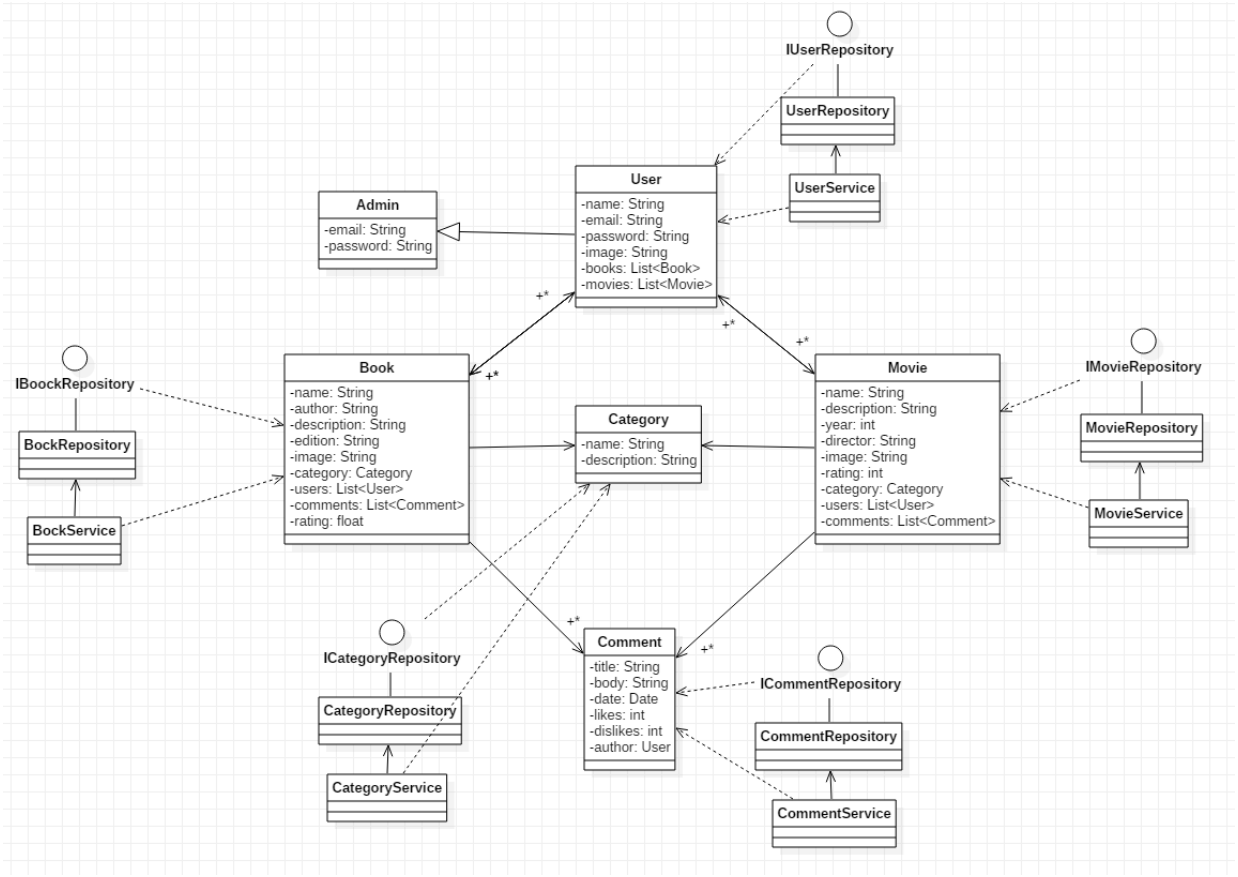
This will be a web application which can be used by everyone who wants to read or add a good review about a book or a movie. A user can register in the system, log in the system, add a new review, view all the reviews and add other comments on exiting reviews.

This application is implemented in **.Net** using **Entity Framework 6** for the back-end, and the front-end application is implemented using **Angular 7** and using the **Angular Material** Theme.

II. Elaboration – Iteration 1.1

1. Domain Model

A domain model is a conceptual model of the domain that incorporates both behavior and data.



2. Architectural Design

2.1 Conceptual Architecture

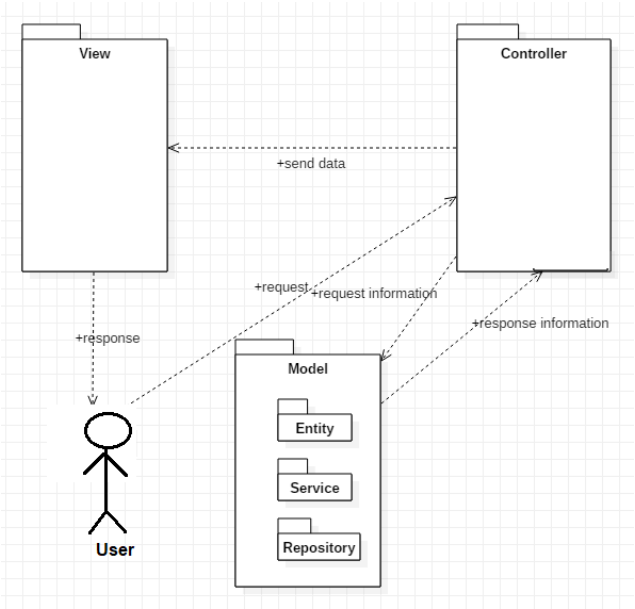
Layered pattern can be used to structure programs that can be decomposed into group of sub-tasks, each of which is a particular level of abstraction. Each layer provides services to the next high layer. The most commonly found four layers of a general application systems are as follows:

1. Presentation Layer
2. Business Logic Layer
3. Data Access Layer

Model-View-Controller architectural pattern follows an elementary idea – we must separate the responsibilities in any application on the following basis:

- **Model:** Handles data and business logic. Model represents an object carrying data. It can also have logic to update controller if its data changes.
- **View:** Presents the data to the user whenever asked for. View represents the visualization of the data that model contains.
- **Controller:** Entertains user requests and fetch necessary resources. Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

Client/server architecture is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server over a network or internet connection. This system shares computing resources. Client/server architecture is also known as a networking computing model or client/server network because all the requests and services are delivered over a network.



The motivation using those two architectural patterns start from the bottom because considering is a web application, delimitation between view, logic and database is truly needed. The MVC design has an organizational structure that better supports scalability. Also organization in layers is a good practice regarding to software engineering. First of all, it is known the layered pattern offers greater structure to the project and also speeds up the developing process once done, because services like the database connection and operations (CRUD) are going to have their own package(**Persistence**) and classes(**UserRepository, BookRepository, Connection, etc**). Another layer, with it's

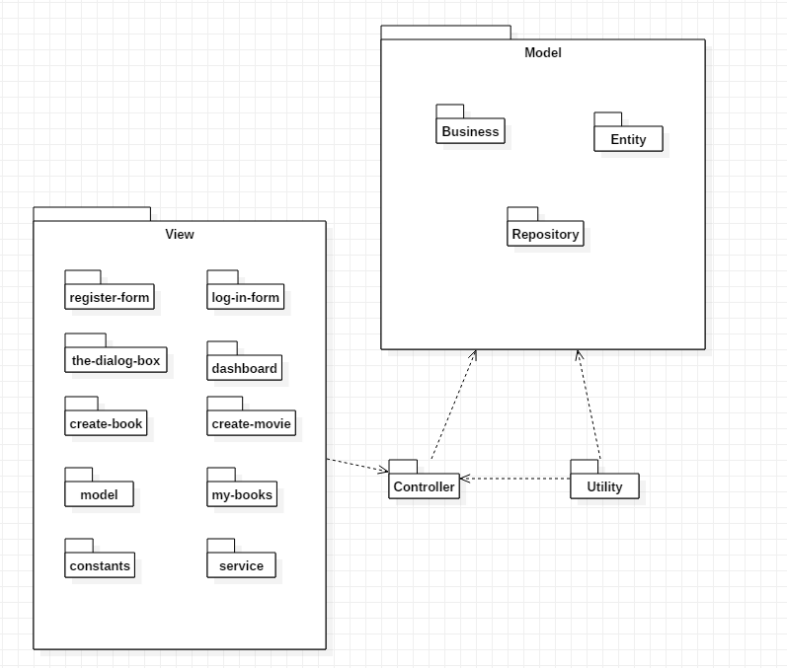
own package(**Business**) will be the business layer. The package will contain classes like: **UserService, BookService etc**, classes which will implement operations on the stored data, accessed through the Persistence layer. Finally, we'll have the Presentation layer, which will encapsulate everything related to the user interface and all that's related to the visual part of the project.

2.2 Package Design

"Tell your friends" application is a web application which is develop in two separated IDEs so it has two different solutions. As it is specified above, application is design using layered architecture pattern, so it is divided in three different layers.

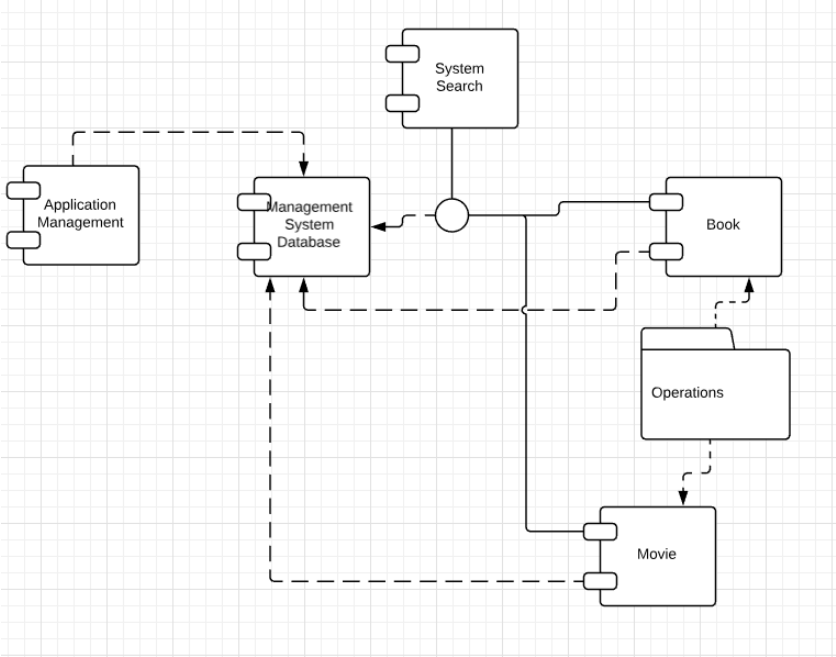
The back-end solution is separated in two packages: Model and Controller. Inside the Controller package there are other three packages: Entity, Repository and Business. Each of them has a specific role and it has a specific layer associated.

The front-end solution is associated with the View part of a MVC architecture project but also with presentation layer of a layered architecture project. Front-end solution is organized in more than 10 packages because each of them is the representation of a component. Each package has three types of files in order to customize, analysis, process data and organize each web-page.

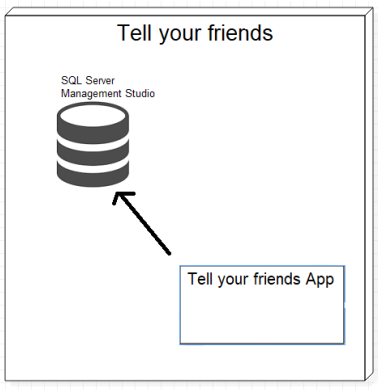


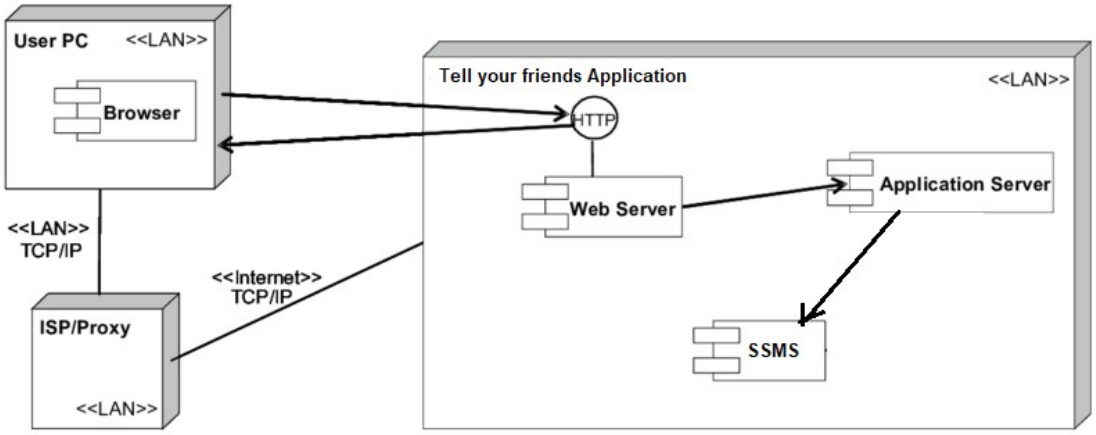
2.3 Component and Deployment Diagrams

A component is a code module. Component diagram are physically analogs of class diagram.
UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.



Deployment diagrams show their physical configurations of software and hardware. Deployment Diagram show the structure of the run-time system, capture the hardware that will be used to implement the system and the links between different items of hardware.





III. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

1.2 Class Design

[Create the UML class diagram; apply GoF patterns and motivate your choice]

2. Data Model

[Create the data model for the system.]

3. Unit Testing

[Present the used testing methods and the associated test case scenarios.]

IV. Elaboration – Iteration 2

1. Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2. Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

V. Construction and Transition

1. System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]

2. Future improvements

[Present future improvements for the system]

VI. Bibliography

1. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
2. https://www.codeproject.com/Articles/552846/Why-s-How-s-of-Asp-Net-MVC-Part-1