<Pharmacy>

Student: Pop David

**Group: 30235**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 4

3. System Architectural Design 6

4. UML Sequence Diagrams 7

5. Class Design 8

6. Data Model 12

7. System Testing 13

8. Bibliography 13

1. Requirements Analysis

# Assignment Specification

The application aims to implement a management system for the employees of a pharmacy. The application should also differentiate administrator from chemist by means of possible actions.

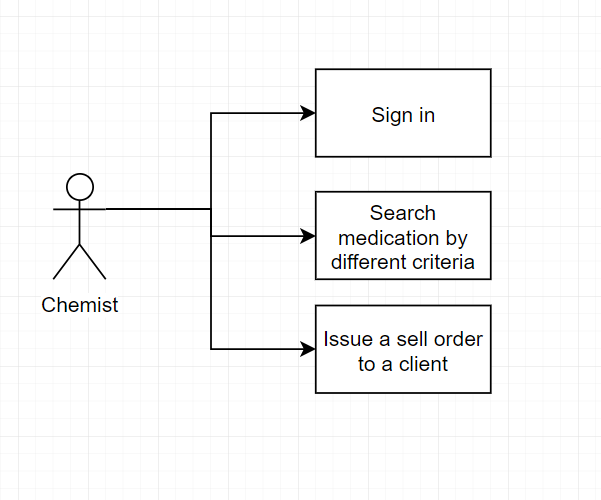
# Functional Requirements

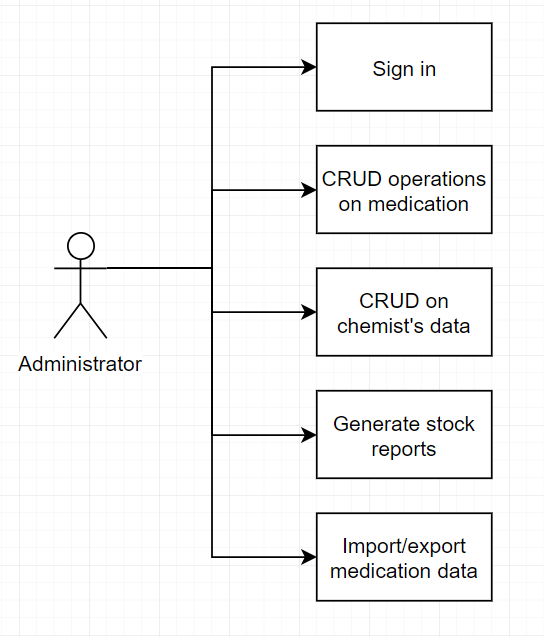
* All input data types are constrained by their respective container, and, optionally, additional logic
* Medication ingredient list is an array of ingredients, separated by a comma, and stored as a string
* After sign-in, the application will redirect the user to the corresponding interface (chemist/administrator)
* All objects stored in the database have been mapped to platform-specific objects using an ORM
* The application must provide means to import and export medication related data to and from XML files
* The application must not be accessible by an unauthorized user

# Non-functional Requirements

* The application should be intuitive enough to be properly used with the knowledge contained in the manual
* The application should not take longer than 1 second to respond to a request

2. Use-Case Model





**Use case:** Export medication data

**Level:** user goal

**Primary actor:** administrator

**Main success scenario:** The agent navigates to the related tab and initiates an export operation. The administrator then chooses a file location and presses “OK”

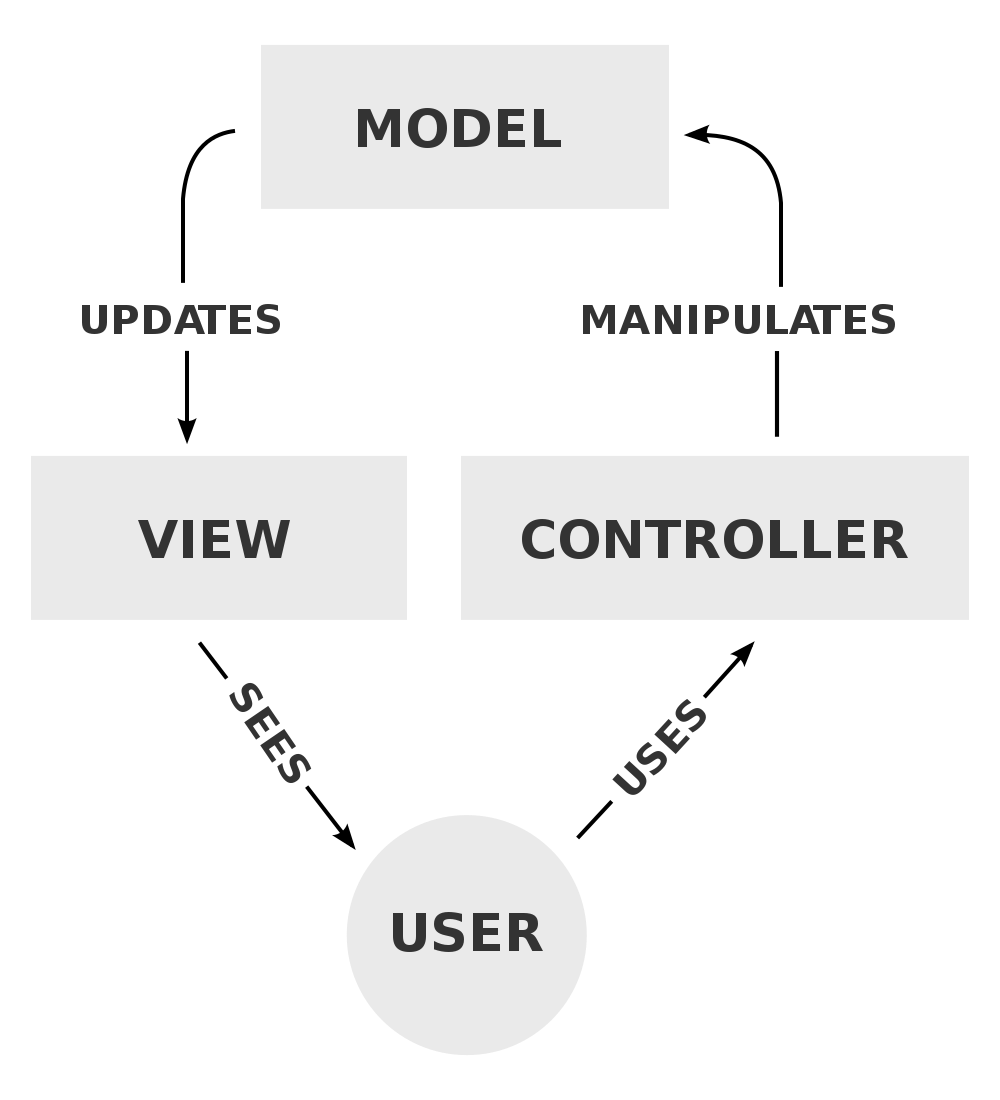
**Extensions:** none

3. System Architectural Design

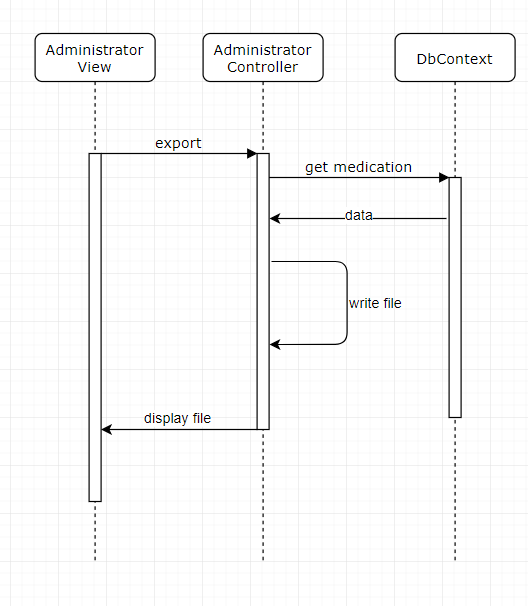
**3.1 Architectural Pattern Description**

The main architectural pattern used in this application is the Model-View-Controller pattern. This pattern states that the application should be divided into three main types of components: controllers (which perform the main logic and IO operations), models (which structurally represent the data being handled in the system) and views (which provide a visual interface with accompanying logic). The difference between MVC and other such patterns is that in the former case, any layer has dependencies on both the other layers.

**3.2 Diagrams**



4. UML Sequence Diagrams



5. Class Design

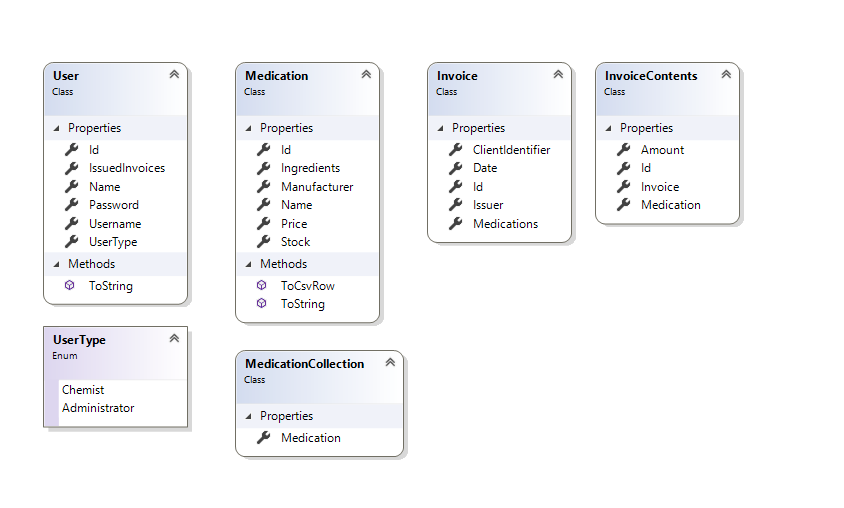
**5.1 Design Patterns Description**

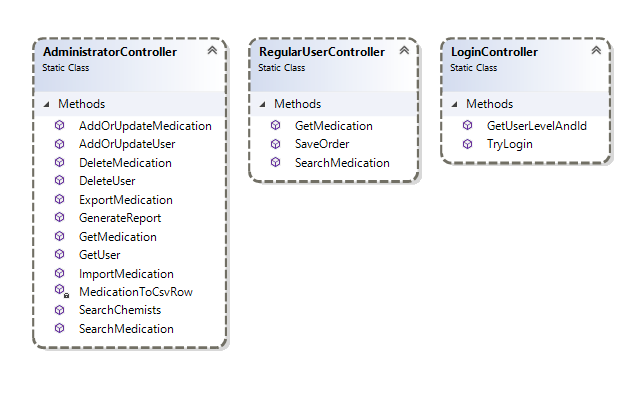
The ORM data pattern is used to bind database tables with objects. As such, any relevant data retrieval and saving is done transparently in the controller level.

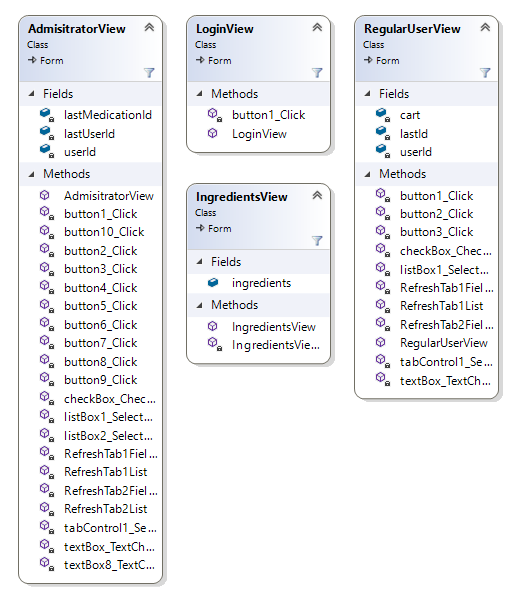
The factory method pattern is used to create a database connection to a databased related to a certain application.

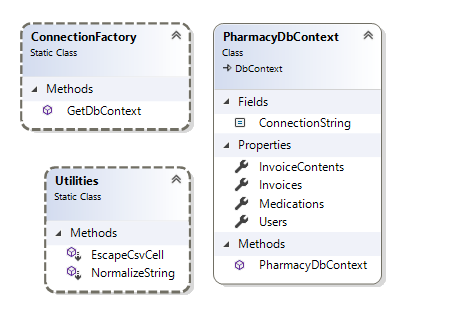
Filter design pattern has been used to search users and medication. The filter design pattern implies the dynamic creation of criteria which filter a collection of objects.

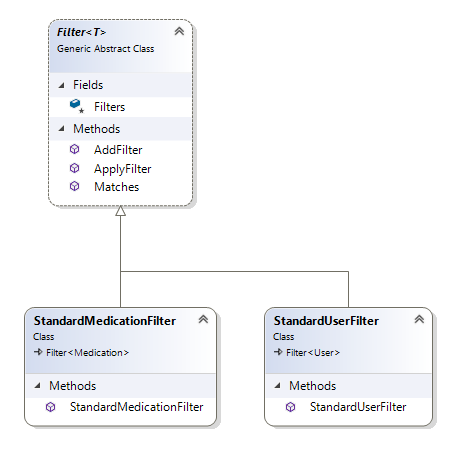
**5.2 UML Class Diagram**



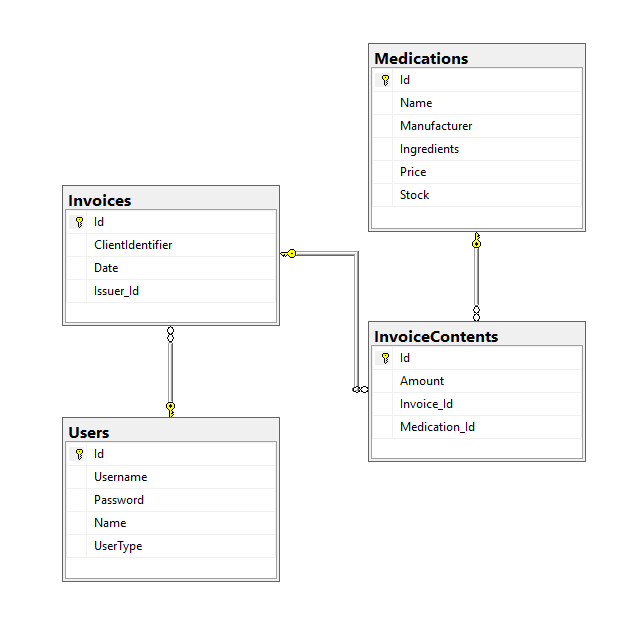








6. Data Model



7. System Testing

Unit tests have been created for testing controller functionality using NUnit 3.1 framework.

User interface has been tested manually.

8. Bibliography