

**<<website review movies>>**

**Software Design Document**

**Table of Contents**

[**I. Overview 3**](#_Toc32534)

[1. Code Packages/Namespaces 3](#_Toc23029)

[2. Database Schema 4](#_Toc13337)

[**II. Code Designs 5**](#_Toc27992)

[1. <Feature/Function Name1> 5](#_Toc4807)

[2. <Feature/Function Name2> 7](#_Toc30313)

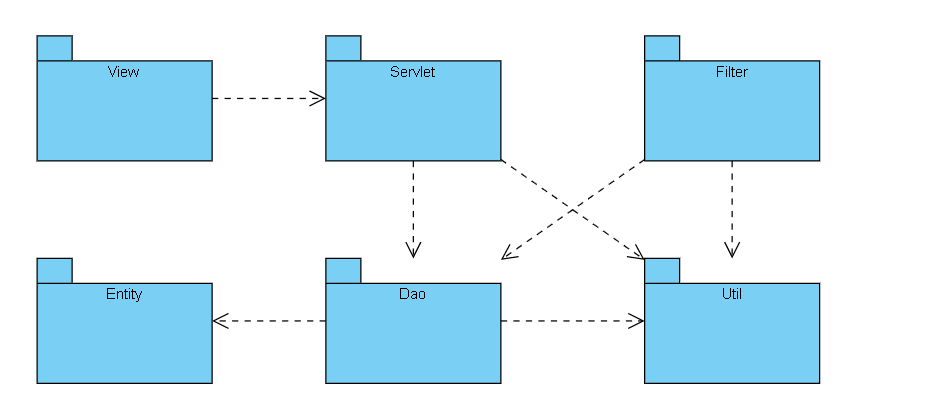
[**III. Database Tables 7**](#_Toc14695)

[1. <Table name 1> 7](#_Toc10672)

[2. <Table name 2…> 7](#_Toc25658)

# I. Overview

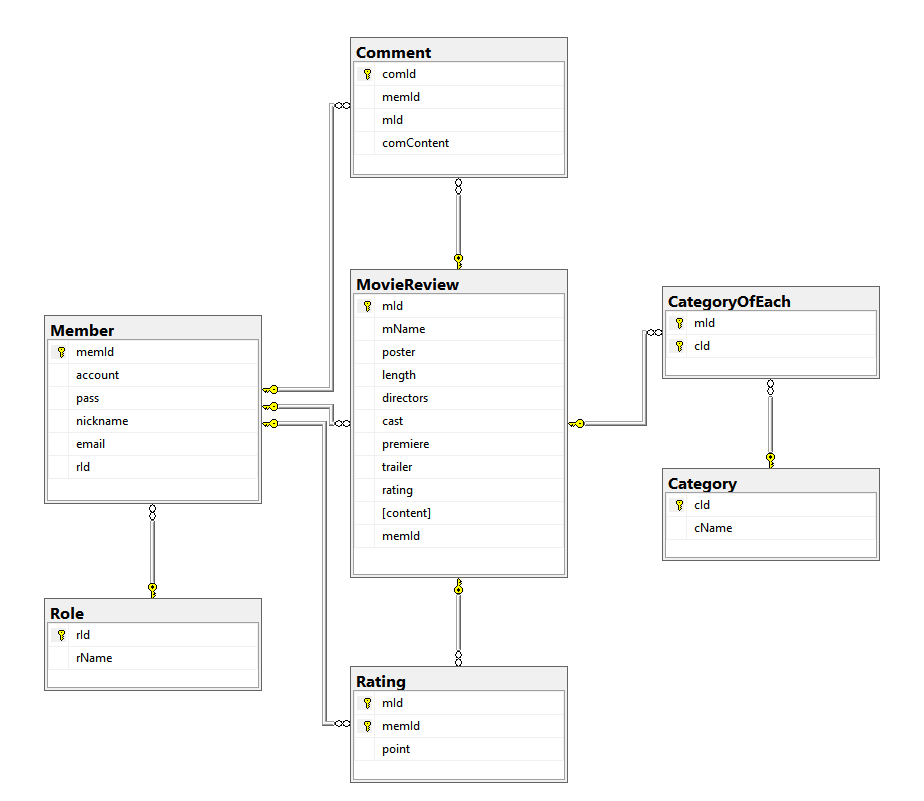
## 1. Code Packages/Namespaces



***Package descriptions & package class naming conventions***

|  |  |  |
| --- | --- | --- |
| No | Package | Description |
| 1 | util | Connect database |
| 2 | servlet | Contains servlet classes to receive the user's request and return |
| 3 | dao | Contains the code of the functions |
| 4 | entity | Contain entities in program |
| 5 | filter | Contains Filter classes to assign permissions to admin and user |
| 6 | view | Contain html file, css file, js file, … |

## 2. Database Schema



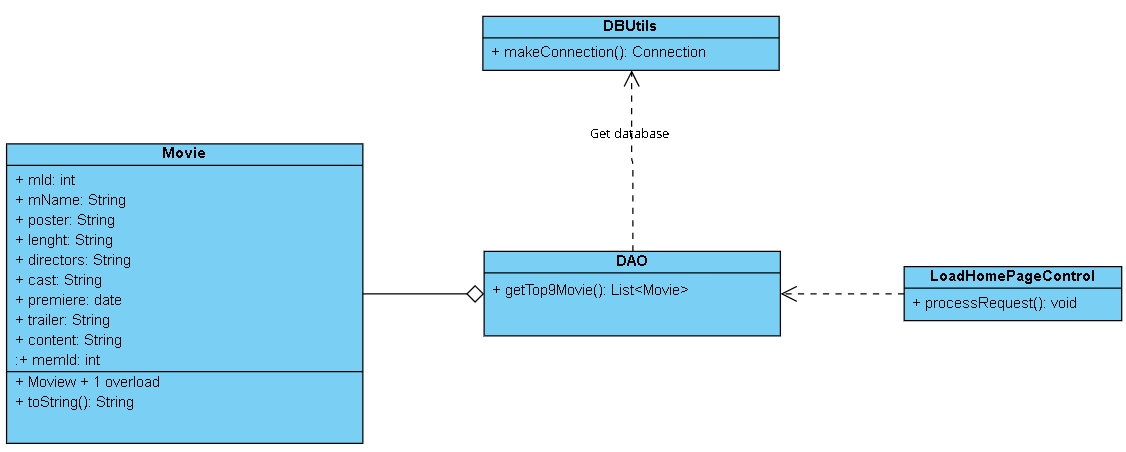
***Table descriptions & package class naming conventions are as below***

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| 01 | Role | - Contain role  - Primary keys: [rId] |
| 02 | Member | - Contain username,password and information of member, be distinguish by [Role]  - Primary keys: [memId]  - Foreign keys: [rId] = [Role.rId] on delete cascade |
| 03 | MovieReview | - Contain information of movie and reviewer’s review  - Primary keys: [mId]  - Foreign keys: [memId] = [Member.memId] on delete cascade |
| 04 | Category | - Contain all of categories of all movies  - Primary keys: [cId] |
| 05 | CategoryOfEach | - Contain all of categories of each movie  - Primary keys: ([mId],[cId])  - Foreign keys: [mId] = [MovieReview.memId] on delete cascade |
| 06 | Comment | - Contain comments of members  - Primary keys: [comId]  - Foreign keys: [memId] = [Member.memId],  [mId]=[MovieReview.mId] |
| 07 | Rating | - Contain each person's article rating for each movie  - Primary keys: ([mId],[memId])  - Foreign keys: [memId] = [Member.memId],  [mId]=[MovieReview.mId] |

# II. Code Designs

## 1. Home / Display Latest Movies

### a. Class Diagram



### b. Class Specifications

*[Provide the description for each class and the methods in each class, following the table format as below]*

#### DBUtils:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *makeConnection()* | *Connect to database* |

#### Members:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *ToString()* | *Print attribute of Member in console* |

#### LoadHomePageControl:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *Get top 9 latest Movies* |

### c. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*

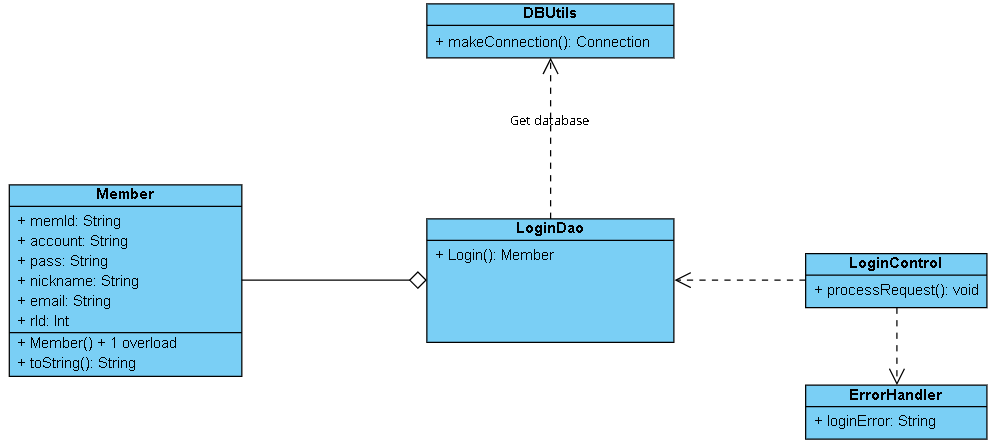


### d. Database queries :



## 2. Authorization / Login

### a. Class Diagram



### b. Class Specifications

#### DBUtils:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *makeConnection()* | Connect to database |

#### Members:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *ToString ()* | Print attribute of Member in console |

#### LoginDao:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Login()* | - Check if Member existed in database. If existed, return that Member.  - Input: Username : String,  Password : String.  - Output: new Member(id, username, password, name, avatar, email, role); |

### c. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*

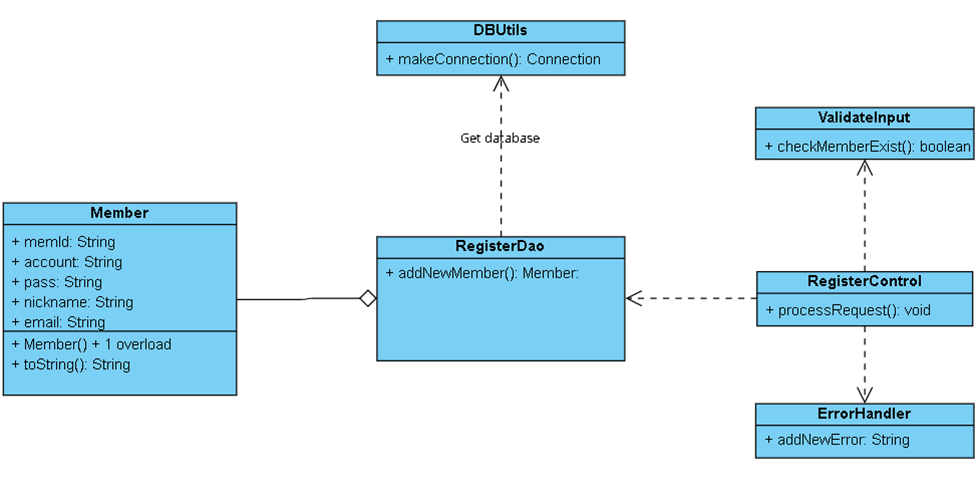


### d. Database queries

*[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]*

## 3. Authorization / Register

### a. Class Diagram



### b. Class Specifications

*[Provide the description for each class and the methods in each class, following the table format as below]*

#### DBUtils:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *makeConnection()* | *Connect to database* |

#### Members:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *ToString* | *Print attribute of Member in console* |

#### RegisterDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | void AddNewMember() | - Create new Member in database.  - Input: name: String, username : String, password : String, email: String |

### c. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*

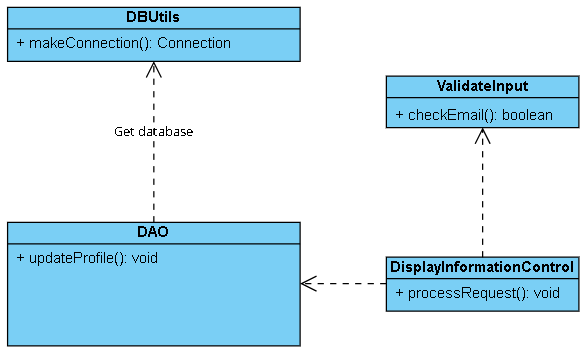


### d. Database queries

*[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]*

## 4. Member/ Update Profile

### a. Class Diagram



### b. Class Specifications

*[Provide the description for each class and the methods in each class, following the table format as below]*

#### XYZ Class

*[Provide the detailed description for the class methods]*

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *<method name>* | *<Description of the method, including the inputs, outputs & internal method processing>* |
|  |  |  |

#### ABC Class

***Class Methods***

*[Provide the detailed description for the class methods]*

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *<method name>* | *<Description of the method, including the inputs, outputs & internal method processing>* |
|  |  |  |

### c. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*



### d. Database queries

*[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]*

# III. Database Tables

## 1. Role

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | rId | int |  |  | ✓ | PK |  |
| 2 | rName | varchar | 30 |  | ✓ |  |  |

## Member

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | memId | int |  |  | ✓ | PK |  |
| 2 | account | varchar | 20 |  | ✓ |  |  |
| 3 | pass | varchar | 20 |  | ✓ |  |  |
| 4 | nickname | nvarchar | 30 |  | ✓ |  |  |
| 5 | email | varchar | 50 |  | ✓ |  |  |
| 6 | rId | int |  |  | ✓ | FK | [rId] = [Role.rId] on delete cascade |

## MovieReview

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | mId | int |  |  | ✓ | PK |  |
| 2 | mName | varchar | 30 |  | ✓ |  |  |
| 3 | poster | varchar | 100 |  | ✓ |  |  |
| 4 | length | varchar | 20 |  | ✓ |  |  |
| 5 | directors | nvarchar | 100 |  | ✓ |  |  |
| 6 | cast | nvarchar | 100 |  | ✓ |  |  |
| 7 | premiere | date |  |  | ✓ |  |  |
| 8 | trailer | varchar | 100 |  | ✓ |  |  |
| 9 | rating | int |  |  | ✓ |  |  |
| 10 | content | nvarchar | MAX |  | ✓ |  |  |
| 11 | memId | int |  |  | ✓ | FK | [memId] = [Member.memId] on delete cascade |

## Category:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | cId | int |  |  | ✓ | PK |  |
| 2 | cName | varchar | 30 |  | ✓ |  |  |

## CategoryOfEach:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | mId | int |  |  | ✓ | PK,FK | [mId] = [MovieReview.memId] |
| 2 | cId | int |  |  | ✓ | FK |  |

## Comment:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | comId | int |  |  | ✓ | PK |  |
| 2 | memId | int |  |  | ✓ | FK | [memId] = [Member.memId], |
| 3 | mId | int |  |  | ✓ | FK | [mId]=[MovieReview.mId] |
| 4 | comContent | nvarchar | 500 |  | ✓ |  |  |

## Rating:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | mId | int |  |  | ✓ | PK,FK | [memId] = [Member.memId] |
| 2 | memId | int |  |  | ✓ | PK,FK | [mId]=[MovieReview.mId] |
| 3 | point | float |  |  |  |  |  |