**I.T.U.**

**FACULTY OF COMPUTER AND INFORMATICS**

**COMPUTER ENGINEERING**



**2013-2014 FALL**

**BLG 361E DATABASE MANAGAMENT SYSTEMS**

**CRN: 10795**

**GROUP 2 TERM PROJECT INDIVIDUAL REPORT**

**E-LIBRARY**

**Tuğrul YATAĞAN**

**040100117**

# **TABLE OF CONTENT**

[TABLE OF CONTENT 2](#_Toc376145154)

[1 General Informations 3](#_Toc376145155)

[1.1 Project Description 3](#_Toc376145156)

[1.2 Development and Execution Environment 3](#_Toc376145157)

[1.3 Connection to MySQL Database Server 5](#_Toc376145158)

[1.4 My Tasks 5](#_Toc376145159)

[2 User Manual 6](#_Toc376145160)

[2.1 User Interface 6](#_Toc376145161)

[**2.1.1** **User Edit Profile Page** 6](#_Toc376145162)

[2.2 Administrator Interface 6](#_Toc376145163)

[**2.2.1** **Admin Edit Profile Page** 6](#_Toc376145164)

[**2.2.2** **Add Source Page** 7](#_Toc376145165)

[**2.2.3** **Source Placed Page** 7](#_Toc376145166)

[**2.2.4** **Edit Source Placed Page** 9](#_Toc376145167)

[3 Technical Manual 10](#_Toc376145168)

[3.1 Database Design 10](#_Toc376145169)

[**3.1.1** **Tables** 10](#_Toc376145170)

[**3.1.2** **Views** 12](#_Toc376145171)

[**3.1.3** **Entity Relationship Diagram** 14](#_Toc376145172)

[3.2 Database Server Installation and Setup 15](#_Toc376145173)

[3.3 Software Design 16](#_Toc376145174)

[**3.3.1** **AddSourcePage.java** 16](#_Toc376145175)

[**3.3.2** **Common.java** 16](#_Toc376145176)

[**3.3.3** **DatabaseConnection.java** 17](#_Toc376145177)

[**3.3.4** **EditAdminProfilePage.java** 21](#_Toc376145178)

[**3.3.5** **EditProfilePage.java** 21](#_Toc376145179)

[**3.3.6** **UserEditProfileForm.java** 22](#_Toc376145180)

[**3.3.7** **SourcePlacedPage.java** 23](#_Toc376145181)

[**3.3.8** **SourcePlacedForm.java** 24](#_Toc376145182)

[**3.3.9** **SourceEditPage.java** 28](#_Toc376145183)

[**3.3.10** **SourceEditForm.java** 28](#_Toc376145184)

[**3.3.11** **Start.java** 29](#_Toc376145185)

[**3.3.12** **WicketApplication.java** 29](#_Toc376145186)

# **General Informations**

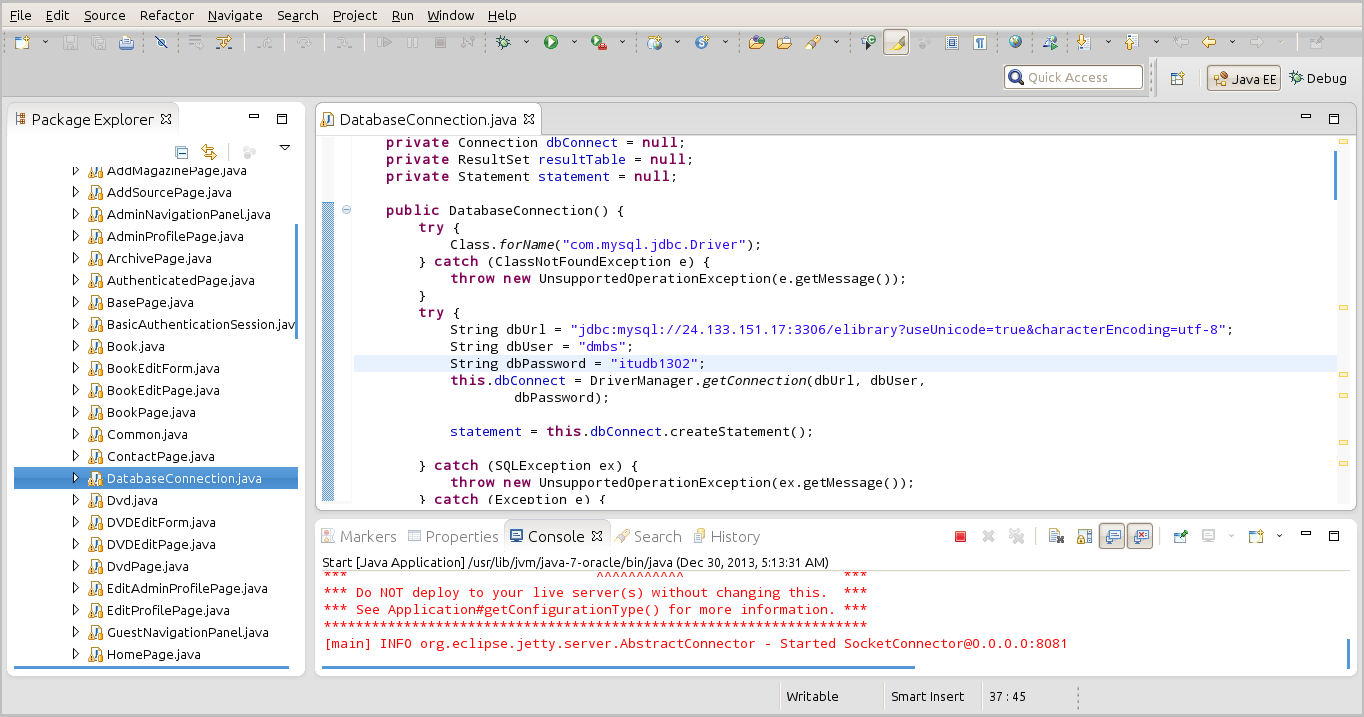
## **Project Description**

The aim of this project is providing a whole library system which is accessible via web. Users can view sources (book, e-book, DVD, magazine etc.) and if users are member of the system, they can simply rent sources. Users can find the location of the sources from website and they can take it actual material from library. If a user wants to use electronic sources from website, he/she can simply access information about electronic source and a link of the electronic source. When a user takes a source from library, user will have 30 days to bring it back. The system have public, regular and administrator users. Administrator users should maintain and control the system (adding source, deleting source, updating source). The system will have different search types (type of source, author of source, category of source etc.). In addition, 10 most popular source listed on the website according to popular search topics and popular rented sources. If a user does not enter the system for a year, account of the user will be deleted by the system. Admin users are the main responsible for continuity of the system, only admins can add source and records, edit source and records. Admin users can see renting process and requests&suggestions. Regular user and public user can give feedback via requests&suggestions page. Public users can only view and search sources but hey cannot rent sources. Everybody can register and be a regularly member. Regular member can view, search and rent sources. If source is not available regular member can add source to their waiting lists. Admin users are responsible for placing materials to library. For these reason admin should enter source information like name, year, category, etc. and then admin should place this source to library. There can be multiple materials of one source in library. If admins wants to place material of existing source they can place source directly to library without entering source information like name, year, category, etc. They only enter place information like bookcase, bookshelf, floor, library, etc. If place is taken a warning message will display.

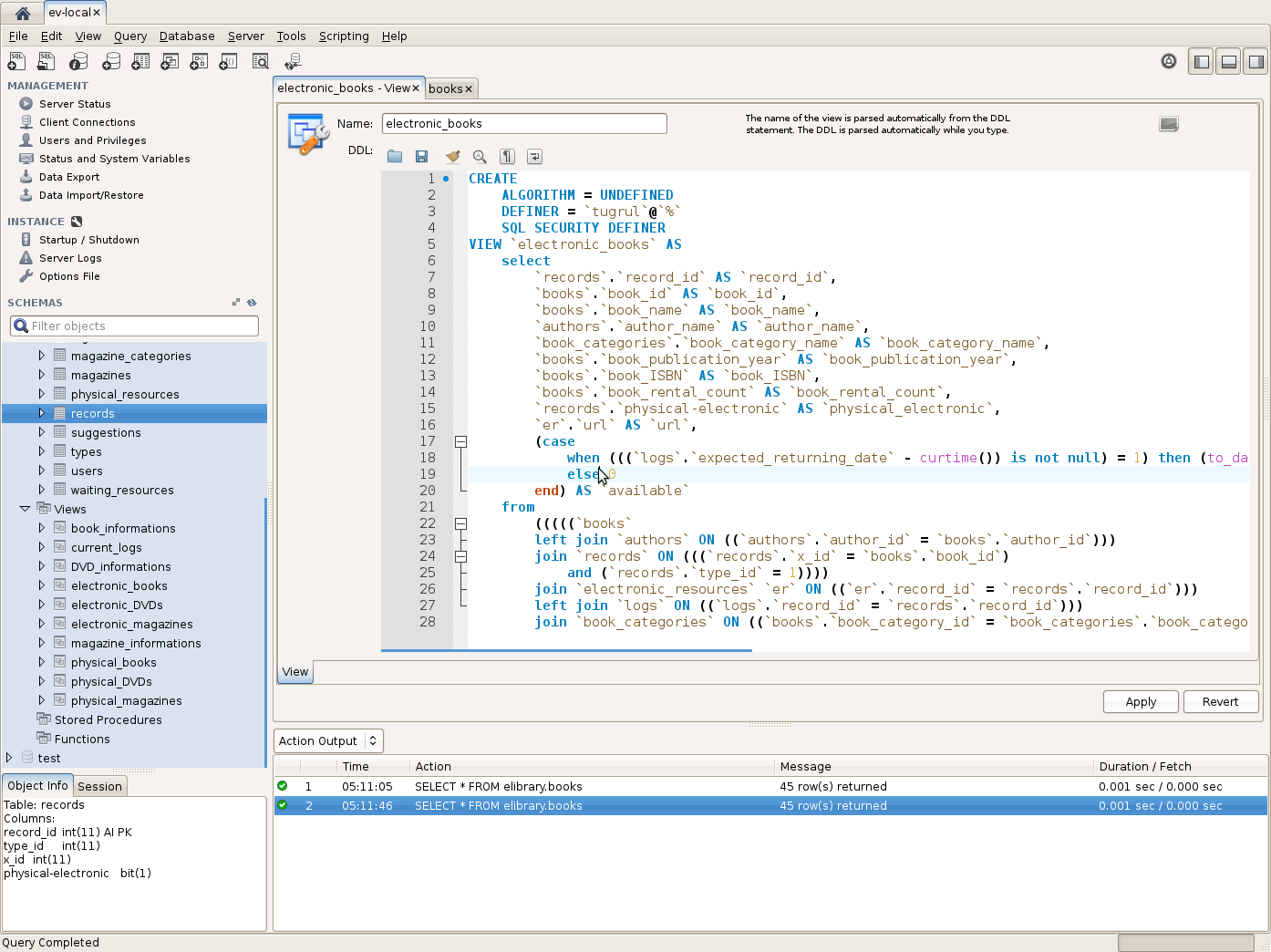
## **Development and Execution Environment**

Project is developed using Java programing language with Apache Wicket web framework and MySQL database server. Project is executed on Apache Tomcat web server. Apache Wicket framework provides connection between Java and HTML. In addition JDBC libraries’ MySQL driver is used for connection MySQL server with Java application. Over JDBC SQL queries can be executed directly.

Project is developed on Eclipse under Linux Ubuntu 12.04 operating system. Eclipse is the integrated development environment and it is also used for developing, debugging and starting Tomcat web server.



Also graphical Interface tool MySQL Workbench has facilitated the database operations during development.



## **Connection to MySQL Database Server**

Project uses one main dedicated MySQL database server. All nodes connects this common server via MySQL JDBC driver. This structure gives us data union while developing the project. Main database connection is established in *DatabaseConnection* class. In *DatabaseConnection* database *host*, *port*, *name*, *user name*, *user* *password* and *encoding* *type* is specified.



## **My Tasks**

* Database server setup and database initialization.
* Creating project repository on Pikacode and setup project’s Mercurial version control
* Classes which are implemented by me:
  + AddSourcePage.java
  + Common.java
  + DatabaseConnection.java
  + EditAdminProfilePage.java
  + ditProfilePage.java
  + UserEditProfileForm.java
  + SourceEditForm.java
  + SourceEditPage.java
  + SourcePlacedForm.java
  + SourcePlacedPage.java
  + Start.java
  + WicketApplication.java

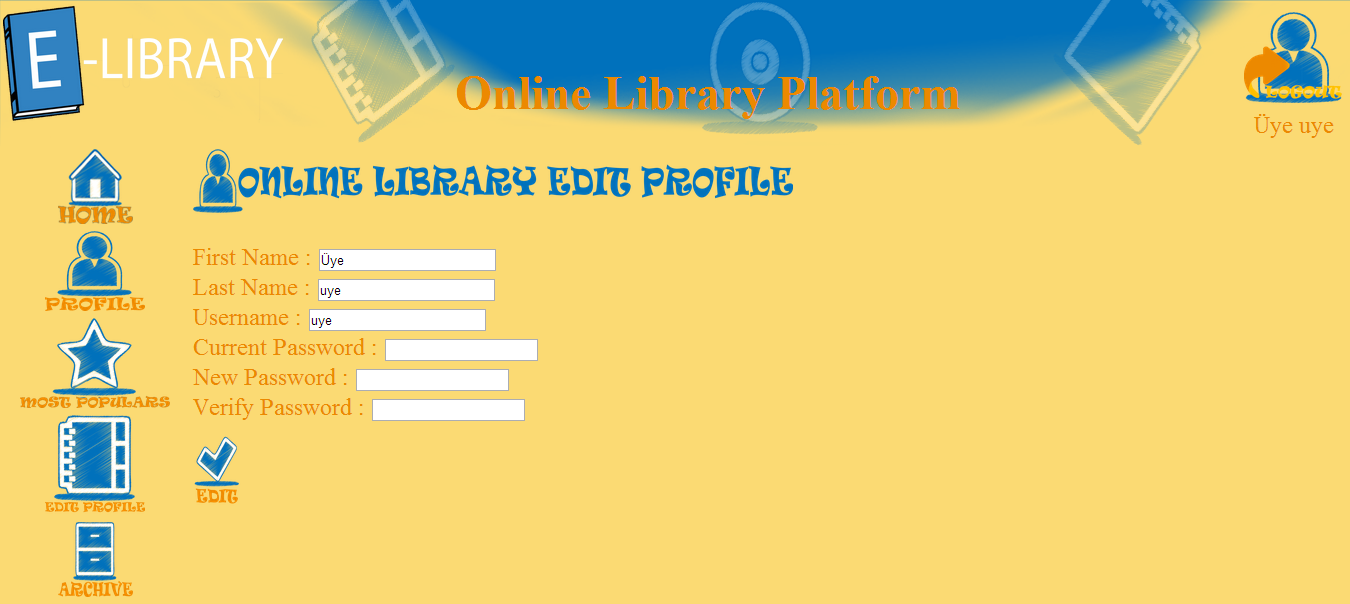
# **User Manual**

## **User Interface**

This Interface is accessible for regular registered users. Everyone can register and can be a member of the system.

### **User Edit Profile Page**

In this page regular user can update his/her informations like name, surname, username and password. User have to reenter his/her current password to change his/her information. When user clicks the edit button user’s informations will be updated on database.

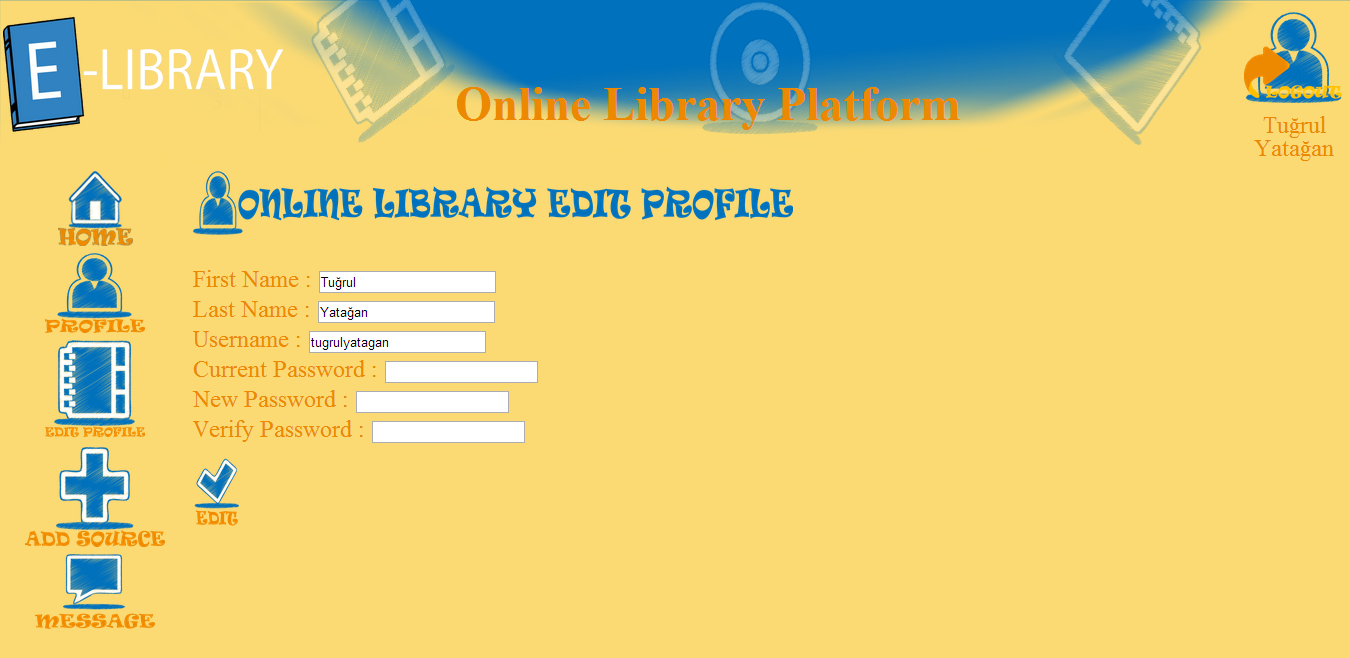


## **Administrator Interface**

This Interface is accessible for only administrator users. Administrative operations are located in this module. Admins are responsible for using this module correct way. Sources and resources informations are taken from admin in this Interface.

### **Admin Edit Profile Page**

In this page regular admins can update his/her informations like name, surname, username and password like regular users. Admin have to reenter his/her current password to change his/her informations. When admin clicks the edit button admin’s informations will be updated on database.



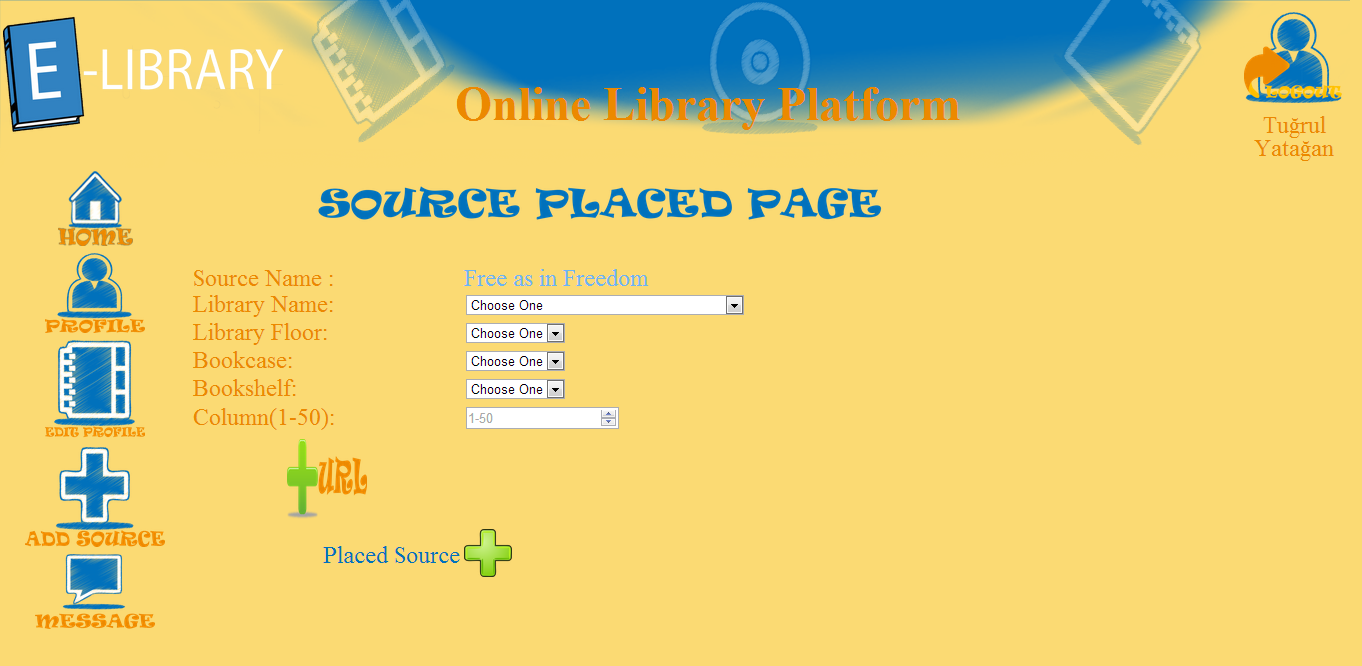
### **Add Source Page**

In this page admin can add source like book, DVD and magazine. Only the admin user can view this page. When admin clicks one of the add source icons, page will redirect to relative source add page. For example if admin clicks add book icon, page will redirect to *AddBookPage.*

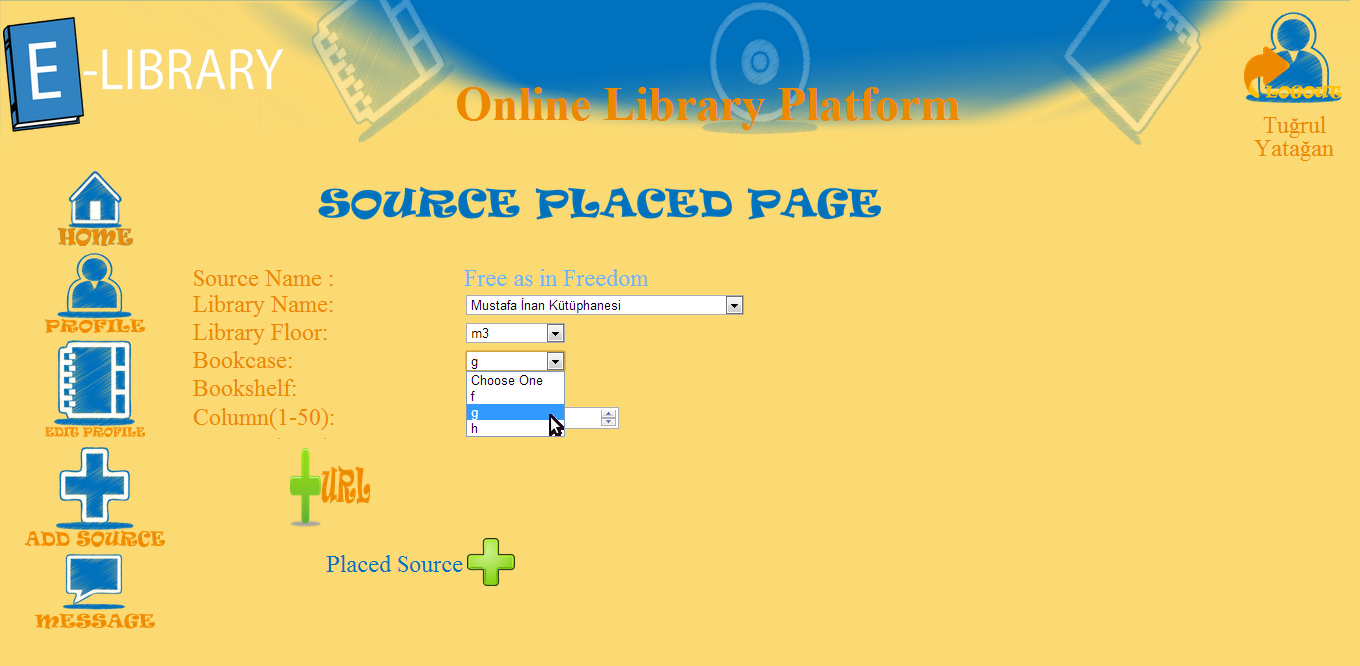


### **Source Placed Page**

In this page admin can place source to library. Admin gives place informations like library, floor, bookcase, bookshelf and column.



When user selects library dropdown choice, floor dropdown choice is restricted according to user’s selection and it continuous to the bookshelf dropdown choices. Every selection restricts next dropdown choice’s selection. When admin clicks placed source icon, source is added to library.

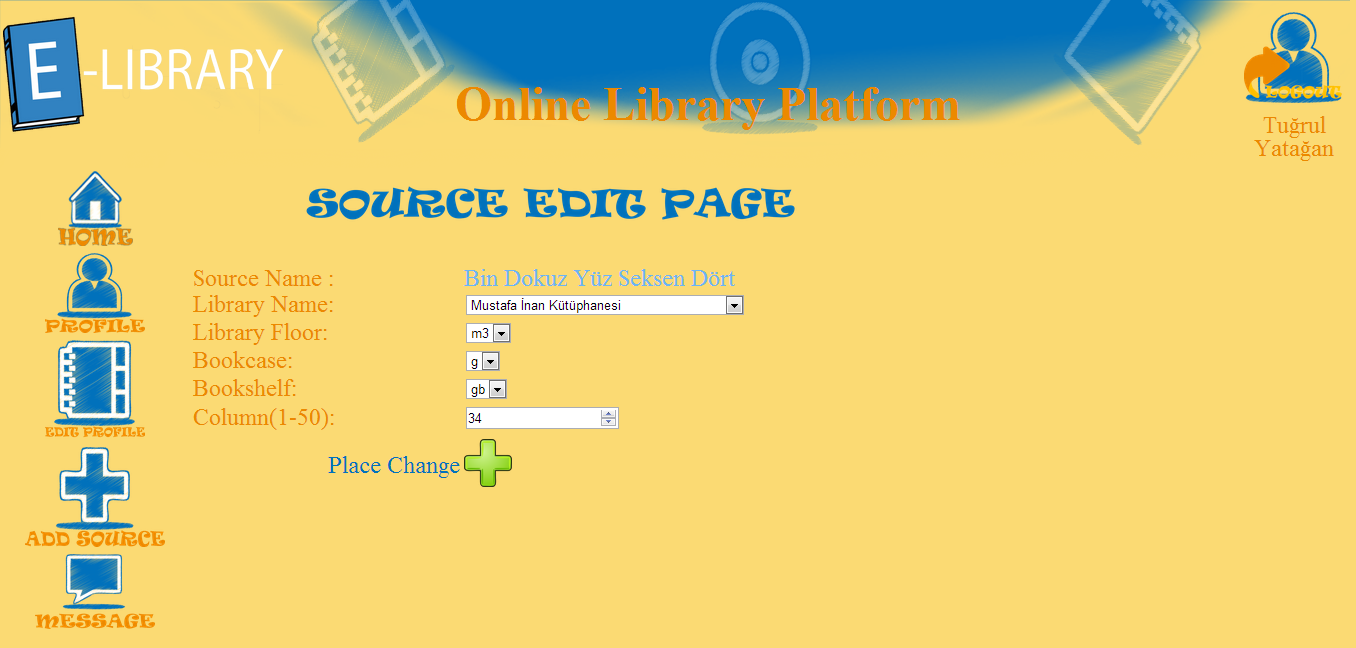


One source can have only one URL field. When admin clicks URL icon, a page with URL field is displayed then admin can enter URL information. When a URL information is entered to a source, add URL icon and field is disappear for the source.



### **Edit Source Placed Page**

In this page admin can edit source’s place information of record. Admin updates place informations like library, floor, bookcase, bookshelf and column.



If record is electronic resource, a page with URL field is displayed then admin can update URL information. When admin clicks place change icon, URL information of electronic record is updated.

# **Technical Manual**

## **Database Design**

### **Tables**

#### **“types” Table**

This table consist of type informations like book, DVD, magazine. *books*, *DVDs* and *magazines* tables connects to *records* table via *type\_id* coloumn.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| type\_id | Int(11) | 1 | 1 |
| type\_name | Varchar(45) | 1 | 0 |

#### **“books” Table**

This table consist of book informations like name, author, year, ISBN and category. Author\_id and book\_category\_id are foreign key of *authors* and *book\_categories* table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| book\_id | Int(11) | Auto increment | 1 |
| book\_name | Varchar(45) | Not Null | 0 |
| author\_id | Int(11) | Not Null | 0 |
| book\_publication\_year | Year(4) | Not Null | 0 |
| book\_ISBN | Varchar(45) | Not Null | 0 |
| book\_rental\_count | Int(11) | Default “0” | 0 |
| book\_category\_id | Int(11) | Not Null | 0 |

#### **“magazines” Table**

This table consist of magazine informations like name, issue number, year and category. *magazine*\_category\_id is foreign key of *magazine\_categories* table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| magazine\_id | Int(11) | Auto increment | 1 |
| magazine\_name | Varchar(45) | Not Null | 0 |
| magazine\_issue\_number | Int(11) | Not Null | 0 |
| magazine\_publication\_date | Date | Not Null | 0 |
| magazine\_rental\_count | Int(11) | Default “0” | 0 |
| magazine\_category\_id | Int(11) | Not Null | 0 |

#### **“DVDs” Table**

This table consist of DVD informations like name, duration, year, and category. *DVD*\_category\_id is foreign key of *DVD\_categories* table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| DVD\_id | Int(11) | Auto increment | 1 |
| DVD\_name | Varchar(45) | Not Null | 0 |
| DVD\_duration | Int(11) | Not Null | 0 |
| DVD\_publication\_date | Date | Not Null | 0 |
| DVD\_rental\_count | Int(11) | Default “0” | 0 |
| DVD\_category\_id | Int(11) | Not Null | 0 |

#### **“bookshelves” Table**

This table consist of list of all bookshelves. One bookshelf keeps name, limit and *bookcase\_id* informations. *bookcase\_id* is foreign key of *bookcases* table. One bookcases may contain multiple bookshelves. Every bookshelf has their limits.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| bookshelf\_id | Int(11) | Auto increment | 1 |
| bookcase\_id | Int(11) | Not Null | 0 |
| bookshelf\_name | Varchar(10) | Not Null | 0 |
| book\_limit | Int(11) | Not Null | 0 |

#### **“bookcases” Table**

This table consist of list of all bookcases. One bookcase keeps name and *floor\_id* informations. *floor\_id* is foreign key of *floors* table. One floor may contain multiple bookcases.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| bookcase\_id | Int(11) | Auto increment | 1 |
| floor\_id | Int(11) | Not Null | 0 |
| bookcase\_name | Varchar(10) | Not Null | 0 |

#### **“floors” Table**

This table consist of list of all floors. One floor keeps name and *library\_id* informations. *library\_id* is foreign key of *libraries* table. One library may contain multiple floors.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| floor\_id | Int(11) | Auto increment | 1 |
| library\_id | Int(11) | Not Null | 1 |
| floor\_name | Varchar(45) | Not Null | 1 |

#### **“libraries” Table**

This table consist of list of all libaries. One library keeps name and addressinformations. *library\_id* is root of all record place tables.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| library\_id | Int(11) | Auto increment | 1 |
| library\_name | Varchar(45) | Not Null | 0 |
| library\_address | Varchar(45) | Not Null | 0 |

#### **“records” Table**

This table consist of records of sources. Every source at least has one record. On record keeps *type\_id*, *x\_id* and *physical-electronic* information. Type id defines type of source Exp: book, DVD, magazine. *x\_id* connect sources table(Exp: *books, DVDs, magazines*) to *records* table. x\_id indicates that primary key of source’s id. Exp: for books*, x\_id* is *book\_id. physical-electronic* coloumn indicates that the source is physical or electronic. Every record mut be inserted *physical\_resources* or *physical\_resources* table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| record\_id | Int(11) | Auto increment | 1 |
| type\_id | Int(11) | Not Null | 0 |
| x\_id | Int(11) | Not Null | 0 |
| physical-electronic | bit(1) | Not Null | 0 |

#### **“physical\_resources” Table**

This table consist of physical informations of records. *records* and placing informations like *bookshelves* connects via this table. *booksehlf\_id* is foreign key of *bookshelves* table. This table also keeps coloumn of the metarial and *current\_log\_id*. *current\_log\_id* is forign key of *logs* table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| record\_id | Int(11) | Auto increment | 1 |
| bookshelf\_id | Int(11) | Not Null | 0 |
| book\_column | Int(11) | Not Null | 0 |
| current\_log\_id | Int(11) | Null | 0 |

#### **“electronic\_resources” Table**

This table consist of URL informations of records. *records* and URL informations connects via this table. Every source can have only one URL information so *url* has unique constraint.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Not Null** | **Primary Key** |
| record\_id | Int(11) | Auto increment | 1 |
| url | Varchar(45) | Not Null | 0 |

### **Views**

#### **“book\_information” View**

This view combines one book’s all informations like name, author, category, year, ISBN, rental count, library name, floor name, bookcase name, bookshelf name, physical status, availability and URL. This view joins *books, book\_categories, authors, libraries, floors, bookcases, bookshelves, records, physical\_records, electronic\_resources* and *logs tables*. This view is union of *electronic\_books* and *physical\_books* views.

|  |  |
| --- | --- |
| **Name** | **Type** |
| record\_id | Int(11) |
| book\_id | Int(11) |
| book\_name | Varchar(45) |
| author\_name | Varchar(45) |
| book\_category\_name | Varchar(45) |
| book\_publication\_year | Year(4) |
| book\_ISBN | Varchar(45) |
| book\_rental\_count | Int(11) |
| library\_name | Varchar(45) |
| floor\_name | Varchar(10) |
| bookcase\_id | Int(11) |
| bookshelf\_id | Int(11) |
| bookcase\_name | Varchar(45) |
| bookshelf\_name | Varchar(45) |
| book\_column | Int(11) |
| physical\_electronic | bit(1) |
| available | bit(1) |
| url | Varchar(45) |

#### **“DVD\_information” View**

This view combines one book’s all informations like name, category, year, duration, rental count, library name, floor name, bookcase name, bookshelf name, physical status, availability and URL. This view joins *DVDs, DVD\_categories, libraries, floors, bookcases, bookshelves, records, physical\_records, electronic\_resources* and *logs tables*. This view is union of *electronic\_DVDs* and *physical\_DVDs* views.

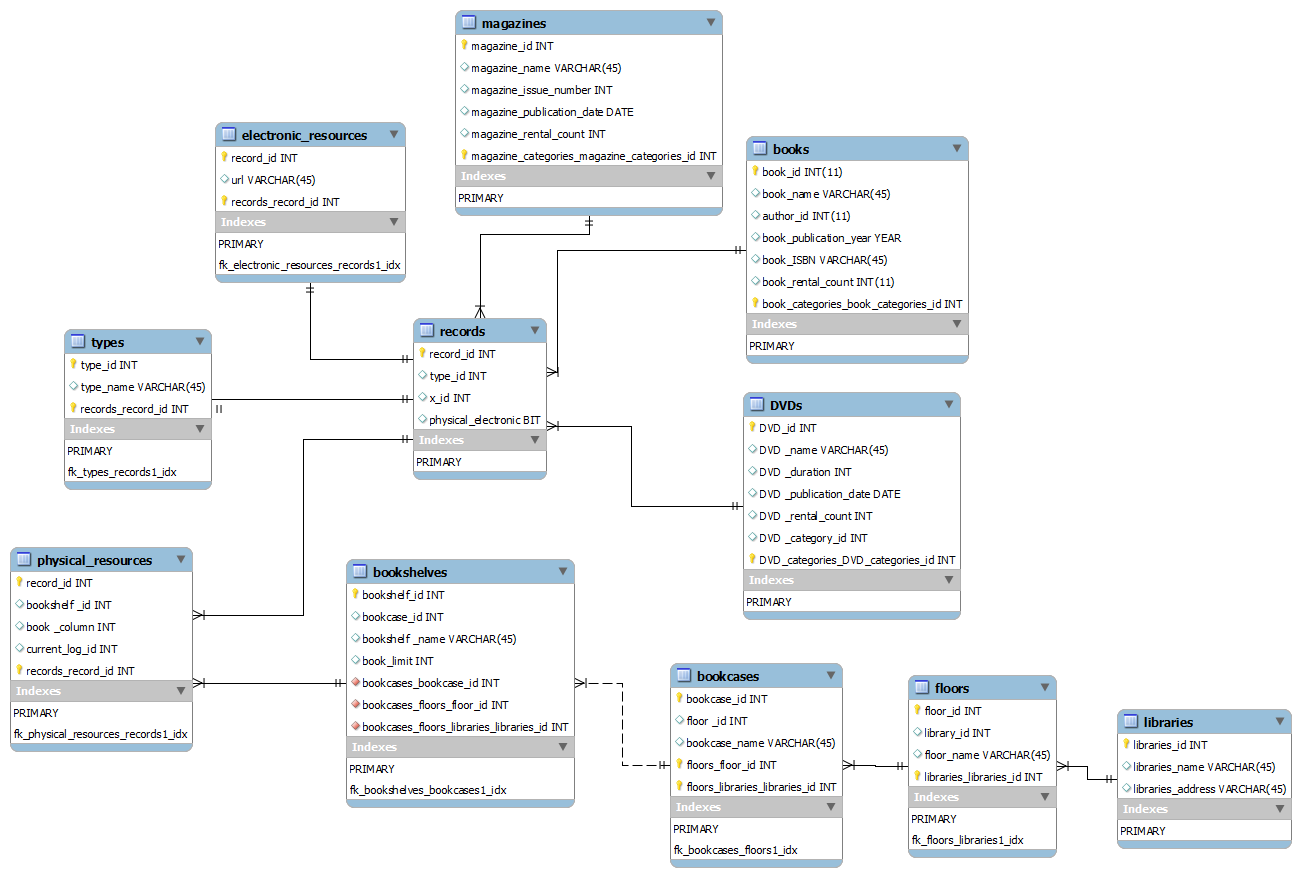
|  |  |
| --- | --- |
| **Name** | **Type** |
| record\_id | Int(11) |
| DVD\_id | Int(11) |
| DVD\_name | Varchar(45) |
| DVD\_category\_name | Varchar(45) |
| DVD\_publication\_year | Year(4) |
| DVD\_duration | Varchar(45) |
| DVD\_rental\_count | Int(11) |
| library\_name | Varchar(45) |
| floor\_name | Varchar(10) |
| bookcase\_id | Int(11) |
| bookshelf\_id | Int(11) |
| bookcase\_name | Varchar(45) |
| bookshelf\_name | Varchar(45) |
| DVD\_column | Int(11) |
| physical\_electronic | bit(1) |
| available | bit(1) |
| url | Varchar(45) |

#### **“magazine\_information” View**

This view combines one magazine’s all informations like name, category, year, issue number, rental count, library name, floor name, bookcase name, bookshelf name, physical status, availability and URL. This view joins *magazines, magazine\_categories, libraries, floors, bookcases, bookshelves, records, physical\_records, electronic\_resources* and *logs tables*. This view is union of *electronic\_magazines* and *physical\_magazines* views.

|  |  |
| --- | --- |
| **Name** | **Type** |
| record\_id | Int(11) |
| magazine\_id | Int(11) |
| magazine\_name | Varchar(45) |
| magazine\_category\_name | Varchar(45) |
| magazine\_publication\_year | Year(4) |
| magazine\_issue\_number | Varchar(45) |
| magazine\_rental\_count | Int(11) |
| library\_name | Varchar(45) |
| floor\_name | Varchar(10) |
| bookcase\_id | Int(11) |
| bookshelf\_id | Int(11) |
| bookcase\_name | Varchar(45) |
| bookshelf\_name | Varchar(45) |
| book\_column | Int(11) |
| physical\_electronic | bit(1) |
| available | bit(1) |
| url | Varchar(45) |

### **Entity Relationship Diagram**



## **Database Server Installation and Setup**

A Linux Debian Wheezy server is used for dedicated database server. MySQL database server installed on Debian Wheezy from standard package manager with following commands:



During installation user name, user password and root password are entered. When installation ends MySQL configuration file is opened with following command:



And the following line is edited for changing server’s listening address (listen all address):



Server is reloaded for changing to take effect



And than we can access to database server on command line by giving user name, host name, host port with following command:During development of this project host name was: 24.133.151.17

Adding new users to database server is done by following code:



After that new database is created on database server with UTF-8 encoding via following command:



## **Software Design**

### **AddSourcePage.java**

This page is used for adding new sources to databases. These sources can be book, DVD or magazine. Only the admin can access this page so admin navigation panel is embedded this page. In this page admin must choose the source type via their links for adding new source. Like all page classes this page also inherits *BasePage*.

This page calls *AddBookPage*, *AddDVDPage* and *AddMagazinePage* via

These links respectively when user clicks one of these iconic buttons. After that these sources pages calls relative forms for them.

### **Common.java**

This class is mainly used for operations which are commonly used for all types for example adding source library. Taking physical library informations are same for all types like *bookcase*, *bookshelf*, *floor* and *library* informations. Also all records has type, log, *date*, *record name* and *record\_id* informations. All these common attributes combines in this class. *x\_id* keeps sources real id’s. For example for book *x\_id* is equal to *book\_id*.

Declarations and data types of this class’s variables are:



All these variables has their own public getter and setter methods. Common class inherits Serializable class for serial operations so we can use these class as list of objects.

### **DatabaseConnection.java**

This class is bridge class for all database JDBC operations in this project. All other classes which uses database connections, must take connection object from this class and use them for database operations. This class handles constant database connection procedures like giving hostname, host port name, database name, database user name, database user password and database connection type parameters in its constructor so initial database connection can be provided. Insert, update and select operations have their own methods. These methods takes string type queries and returns database *ResulSet* objects. Class itself has three variables:

*dbConnect* object supplies database connections and is closed in *close*() method. *resultTable* object is returned from *GetResult* and *Insert* methods to caller. *Statement* object executes string queries in *executeQuery* method of JDBC.

JDBC MySQL driver selection is done with:

JDBC MySQL database connection is provided. Database *host*, *port*, *name*, *user name*, *user* *password* and *encoding* *type* is specified in this block: Constructor creates *statement* object for other methods query executions.

*GetResult* method takes select queries, executes them and returns *Resultset* objects. This method is commonly used in whole projects, all select queries is done by this method. String type queries executed in *Statement* objects which is supplied by class’s constructor. One *Statement* objects can be used multiple times until object’s *close* method’s execution.

 *Insert* method takes insert queries and executes them. This method is commonly used in whole projects, all insert queries is done by this method. String type queries executed in *Statement* objects which is supplied by class’s constructor. One *Statement* objects can be used multiple times until object’s *close* method’s execution.

*close* method closed database connection object, database statement object, ant *resultset* object. After execution this method, object of this class and return value of class *resultset* object cannot be used.



*get\_books* method returns list of book object which is retrieved from *book\_informations* view in database. Getting book information is an operation which is commonly used in project so a function is written for getting all books information at one time. All book object’s variables retrieved from database and setted to objects by setter methods as possible as. Common variables like *bookcase*, *bookshelf*, *floor*, *library*, *rental count*, *year*, *type* and specific variables to books like *book name*, *author*, *category*, *ISBN* are retrieved from database:

Example select query for *get\_books* method in book page is:



*get\_dvds* method returns list of DVD object which is retrieved from *dvd\_informations* view in database. Getting dvd information is an operation which is commonly used in project so a function is written for getting all DVDs information at one time. All DVD object’s variables retrieved from database and setted to objects by setter methods as possible as. Common variables like *bookcase*, *bookshelf*, *floor*, *library*, *rental count*, *year*, *type* and specific variables to DVDs like *DVD name*, *duration*, *category* are retrieved from database:

Example select query for *get\_dvds* method in DVD page is:



*get\_magazines* method returns list of magazine object which is retrieved from *magazines\_informations* view in database. Getting magazine information is an operation which is commonly used in project so a function is written for getting all magazines information at one time. All magazine object’s variables retrieved from database and setted to objects by setter methods as possible as. Common variables like *bookcase*, *bookshelf*, *floor*, *library*, *rental count*, *year*, *type* and specific variables to books like *magazines name*, *issue number*, *category* are retrieved from database:

Example select query for *get\_magazines* method in magazine page is:



### **EditAdminProfilePage.java**

This page is used for updating admin’s profile informations like nickname, name, surname, and password. For updating information, admin has to enter his/her current password. Obviously only the admin can access this page so admin navigation panel is embedded to this page. After opening this page, it is redirected to common *EditProfilePage*. Like all page classes this page also inherits *BasePage*.



### **EditProfilePage.java**

This page is used for updating users profile informations like nickname, name, surname, and password. For updating information, user has to enter his/her current password. Users and admins can access this page so admin navigation panel or user navigation panel is embedded to the page according to person’s *authorityState.* After user/admin distinguish, a form object *UserEditPrfileForm* is called. Like all page classes this page also inherits *BasePage*.



### **UserEditProfileForm.java**

This form is used for updating users profile informations like nickname, name, surname, and password in database. For updating information, user has to enter his/her current password. These variables keeps in class as string and taken from user as *TextField*, or *PasswordField* component.

Logged user’s information is taken from session class:



Firstly current user’s user\_id is founded via a select query. Than according to user’s input name or surname or nickname which was not null so only the valid inputs are updated on the database. To find a user’s row, *user\_id* is enough for WHERE clause. Finally if user’s actual current password is same as the input’s current password, password information is updated on the database.

And then new name and surname setted to session class for displaying them on main navigation bar in home page:



### **SourcePlacedPage.java**

This page is used for adding source information to library like bookcase, bookshelf, floor and library for physical sources or URL for electronic sources. For adding library information user must be admin. Obviously only the admin can access this page so admin navigation panel is embedded to this page. After opening the page, it is redirected to *SourcePlacedForm*. Like all page classes this page also inherits *BasePage*. Class’s constructor takes two arguments; first one is *Common* object for source which is placed in library and second is *checkUrl* bit for taking URL information for electronic sources or taking place information for physical sources.



### **SourcePlacedForm.java**

This form is used for taking source information and adding to library like bookcase, bookshelf, floor and library for physical sources or URL for electronic sources. Class’s constructor takes two arguments; first one is *Common* object for source which is used to place in library and second is *checkUrl* bit for taking URL information for electronic sources or taking place information for physical sources.

There is 4 dropdown choices in this page. AJAX is used for dropdown choices. *SelectedMake* string variables keeps choice of the user and *modelsMap* map variables connects one choice to another. When user selects top dropdown choice, second dropdown choice is restricted according to user’s selection and it continuous to the last dropdown choices. Every selection restricts next dropdown choice’s selection.



Labels and fields of this pages are URL, column, library, floor, bookcase and bookshelf. URL and column information is taken from user by manual in text or number fields.



During dropdown choice restriction all information about place is retrieved from database. All selections are mapped and listed next selections so user can be select dropdown choices by one by. Retrieving place information from database is done by following code block:



When user clicks Placed Source icon, according to type of the record (electronic or physical) source will be added to library. Following code block inserts physical record Into *records* table and retrieves record, library, floor, bookcase and bookshelf id’s from database then inserts this id informations Into *physical\_resources* tables. All these id’s connects on *physical\_resources* table like this:



When user clicks Placed Source icon, according to type of the record (electronic or physical) source will be added to library. Following code block inserts electronic record Into *records* table and retrieves record id’s from database then inserts this id information Into *electronic\_resources* tables. All these id’s connects on *electronic\_resources* table like this:



İf a conflict occurs while adding record to *records* table, a *DuplicateKeyException* exception is thrown *physical\_resources* tables has unique key constraInt on its *record\_id, bookshelf\_id, book\_column* coloumns and *electronic\_resources* table tables has unique key constraInt on its *record\_id, url* columns. If a conflict occurs *record\_id* column in *records* table must be deleted. Following catch statement handles this duplicate key exception:



After all adding source to place operations user is redirected to sources page according to source type. Redirection need source’s type and id. Following code makes this redirection:



### **SourceEditPage.java**

This page is used for updating source information in library like bookcase, bookshelf, floor and library for physical sources or URL for electronic sources. For updating library information user must be admin. Obviously only the admin can access this page so admin navigation panel is embedded to this page. After opening the page, it is redirected to *SourceEditForm*. Like all page classes this page also inherits *BasePage*. Class’s constructor takes an argument, *Common* object for source which is updated in library information like URL for electronic sources or taking place information for physical sources.



### **SourceEditForm.java**

This form is used for editing source information and updating in library like bookcase, bookshelf, floor and library for physical sources or URL for electronic sources. Class’s constructor takes an argument *Common* object for source which is used to place in library and it takes URL information for electronic sources or takes place information for physical sources.

There is 4 dropdown choices in this page like *SourcePlacedForm*. AJAX is used for dropdown choices. *SelectedMake* string variables keeps choice of the user and *modelsMap* map variables connects one choice to another. When user selects top dropdown choice, second dropdown choice is restricted according to user’s selection and it continuous to the last dropdown choices. Every selection restricts next dropdown choice’s selection.

This page is almost same as *SourcePlacedForm,* only differences between *SourcePlacedForm and SourceEditForm* is default variables. For editing source current source information has to be inserted textboxes and dropdown choices. Embedding initial default values into form is made by following code blocks all other codes are same as the *SourcePlacedForm* so same codes are not explained again, only the different code blocks are explained:



Giving default values to string *SelectedMake* variables is enough for embedding default values. Giving *Model* objects to field are also enough for embedding default values.

### **Start.java**

This class is main class of the project. Project starts from this class. In this class mainly server settings are done like port number, timeout and debug mode selection.

When application is start a starting message is prInted to console and then server is started. When developer enters something on console, a stopping message is prInted to console and then server is stopped. When server is stopped application should be stopped either.



### **WicketApplication.java**

This class provides entegration between wicket application and project. This class mostly remained untouched, changes on this class for session class adjustment. The class inherits *AuthenticatedWebApplication* class for session operation by this Session class can control whole project. Sign in page is overridden for session log in operations. Finally *WicketApplication* class’s *init* method is called.

