PIICHKARI-Online Paint Editor

PIICHKARI (ONLINE PAINT EDITOR)

Software Design and Analysis

Course: SE 606

Submitted to,

Md. Nurul Ahad Tawhid
Assistant Professor
Institute of Information Technology
University of Dhaka

Submitted By,

Shahla Shaan Ahmed (BSSE 0810)
Taslima Binte Kamal (BSSE 0835)
BSSE Session: 2015-2016

Institute of Information Technology
University of Dhaka

Conducting Component Level design

Step 1: Identify all design classes that correspond to the problem domain.

Using the requirements and architectural model, each analysis class and architectural component is elaborated.

Identified Classes:

- ❏ User
  - ❏ Admin
  - ❏ Member
- ❏ Image
- ❏ Authentication
- ❏ Validation

Step 2: Identify all design classes that correspond to the infrastructure

These classes are not described in the requirements model and are often missing from the architecture model, but they must be described at this point.

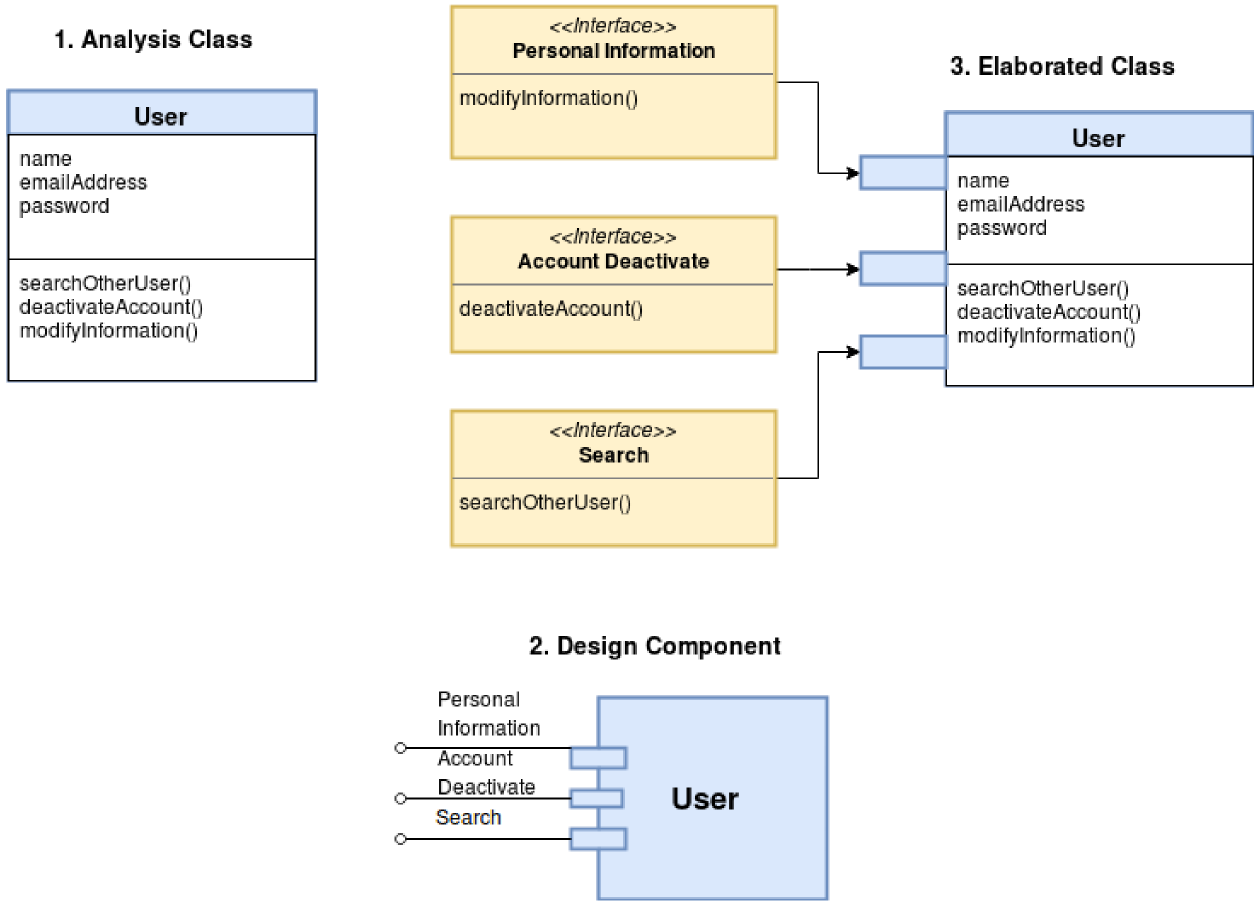
Identified Classes:

- ❏ DatabaseOperator

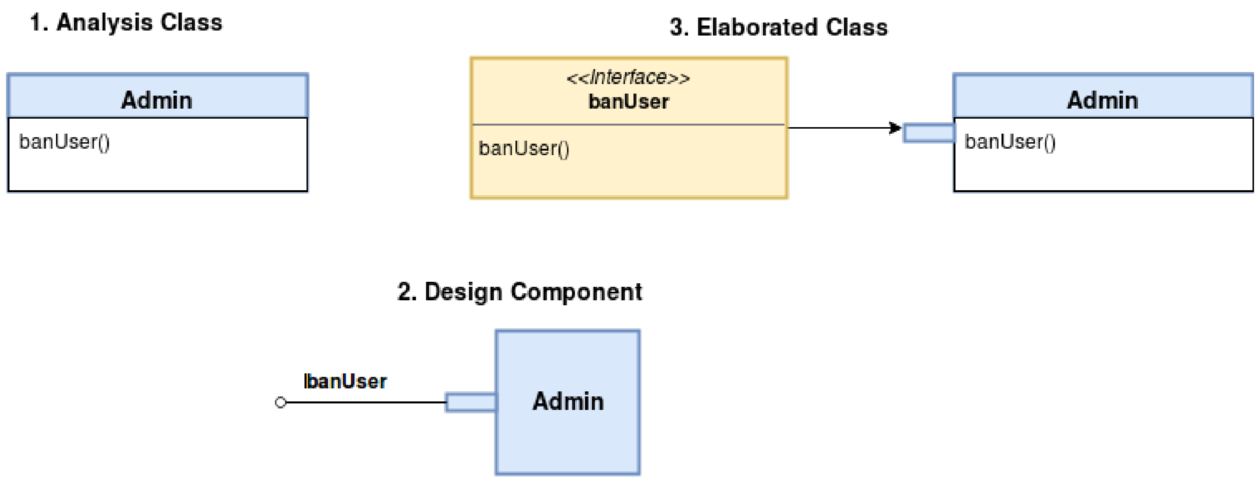
Step 3. Elaborate all design classes that are not acquired as reusable components.

Elaboration requires that all interfaces, attributes, and operations necessary to implement the class be described in detail.

User Class:



Admin Class:



Member Class:

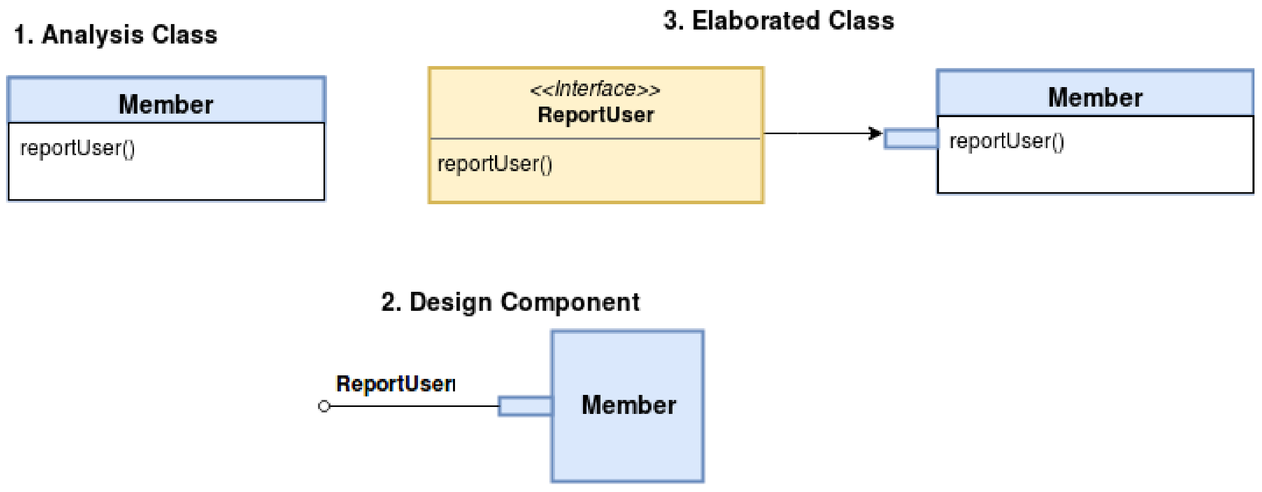
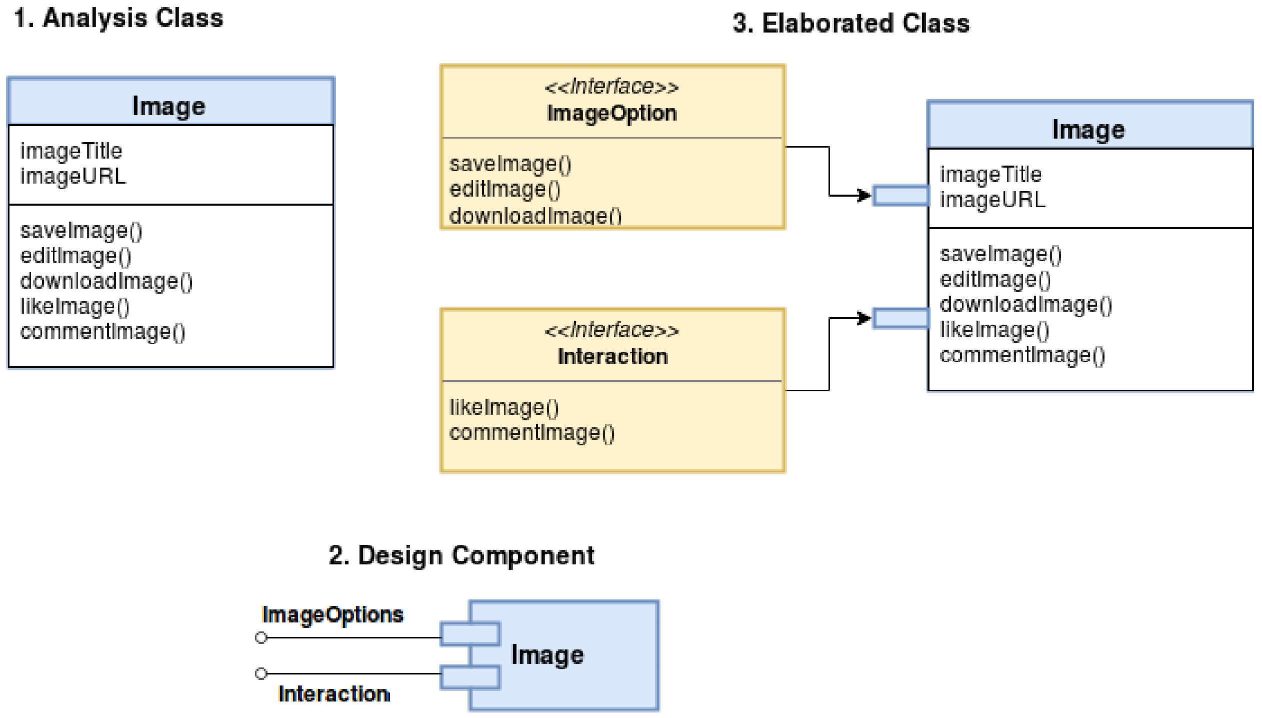
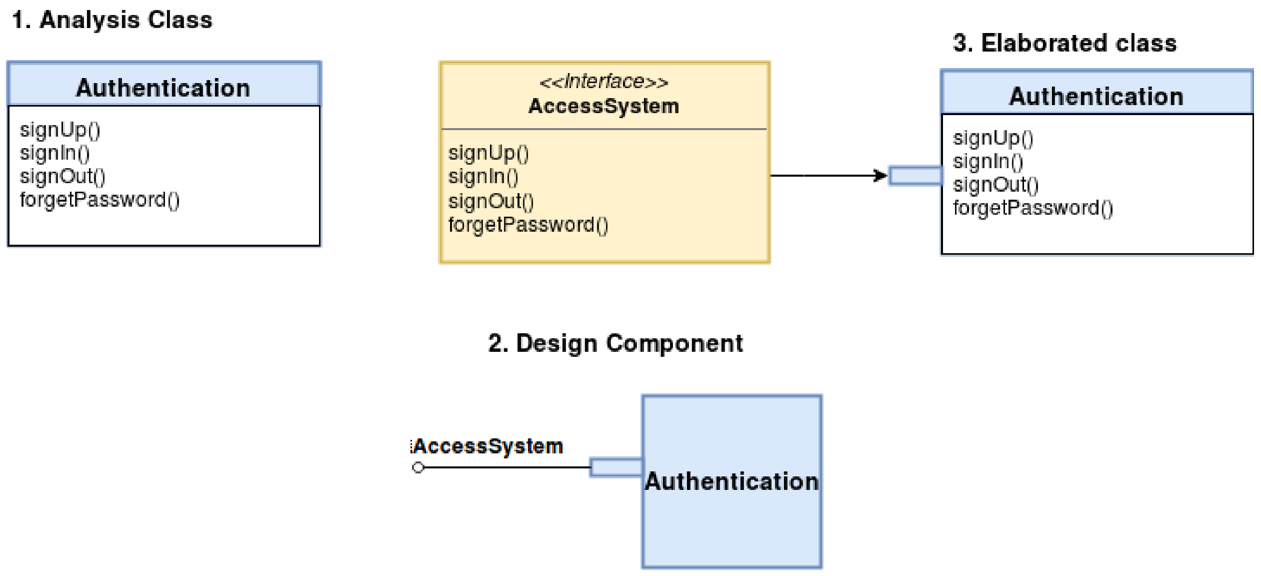


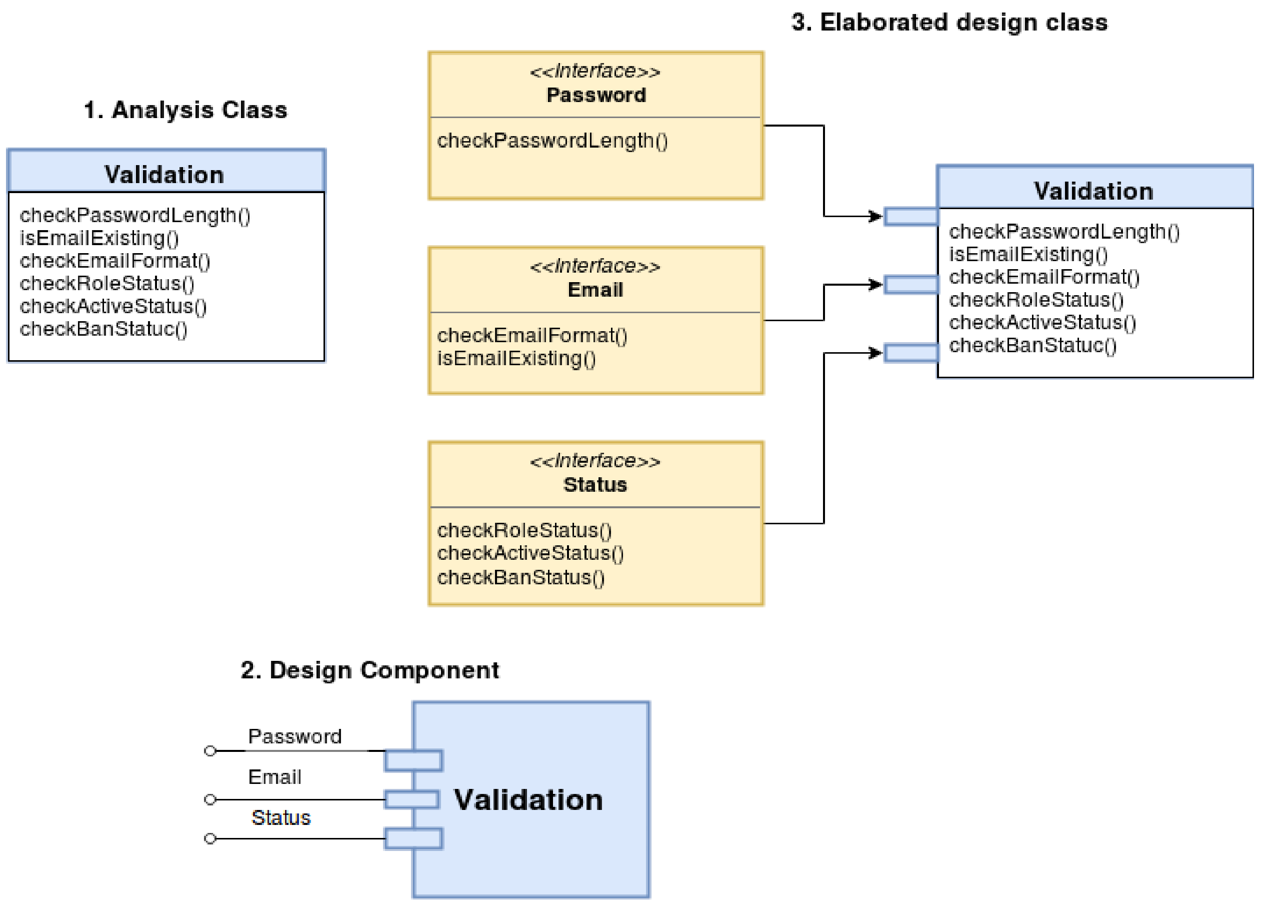
Image Class:



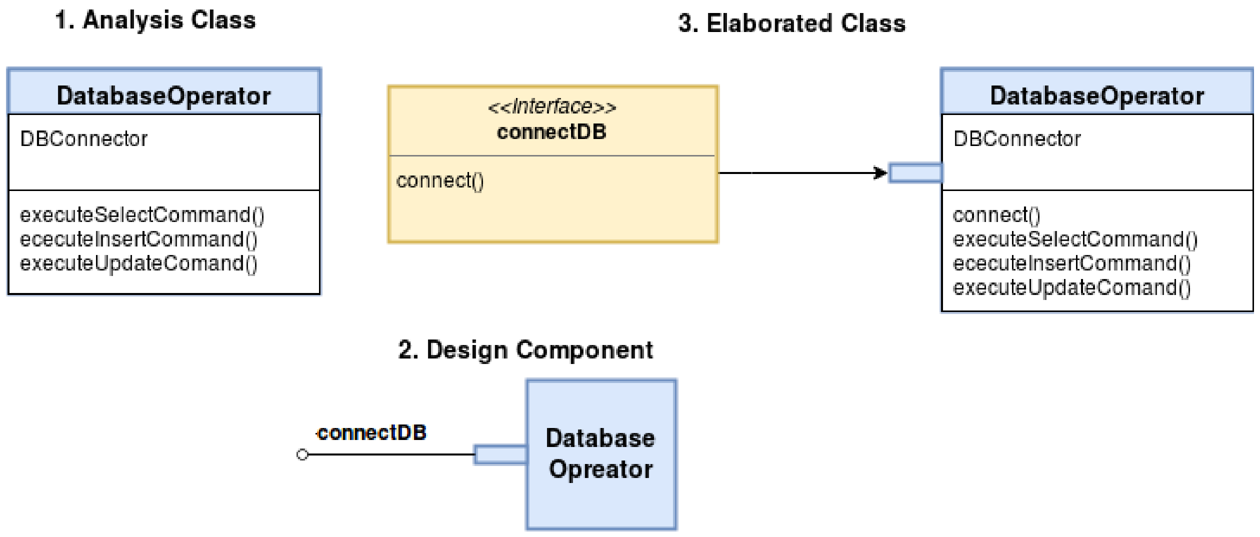
Authentication:



Validation:



DatabaseOperator Class:



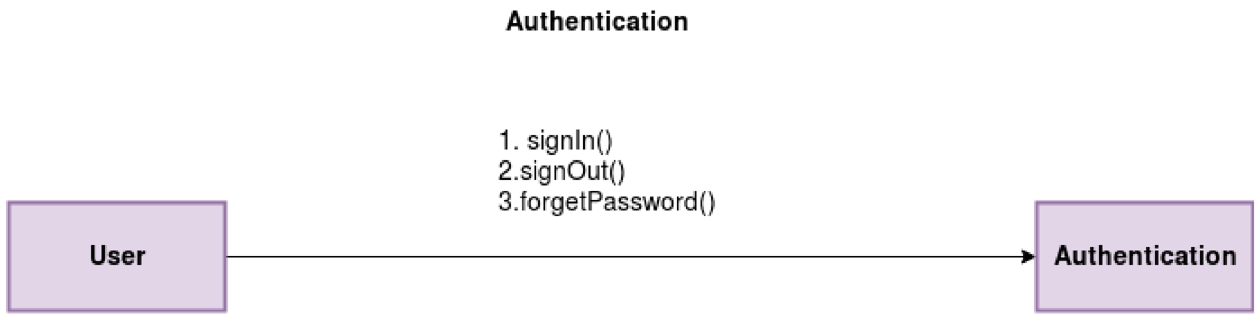
Step 3a. Specify message details when classes or components collaborate.

The requirements model makes use of a collaboration diagram to show how analysis classes collaborate with one another. As component-level design proceeds, it is sometimes useful to show the details of these collaborations by specifying the structure of messages that are passed between objects within a system. Although this design activity is optional, it can be used as a precursor to the specification of interfaces that show how components within the system communicate and collaborate.

Syntax:

[guard condition] sequence expression (return value) : message name (argument list)

Authentication:



Registration:

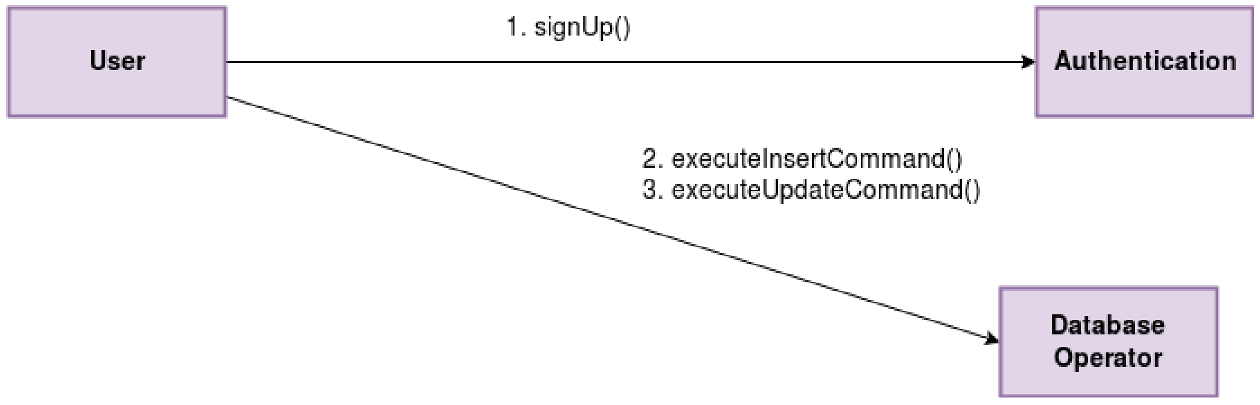
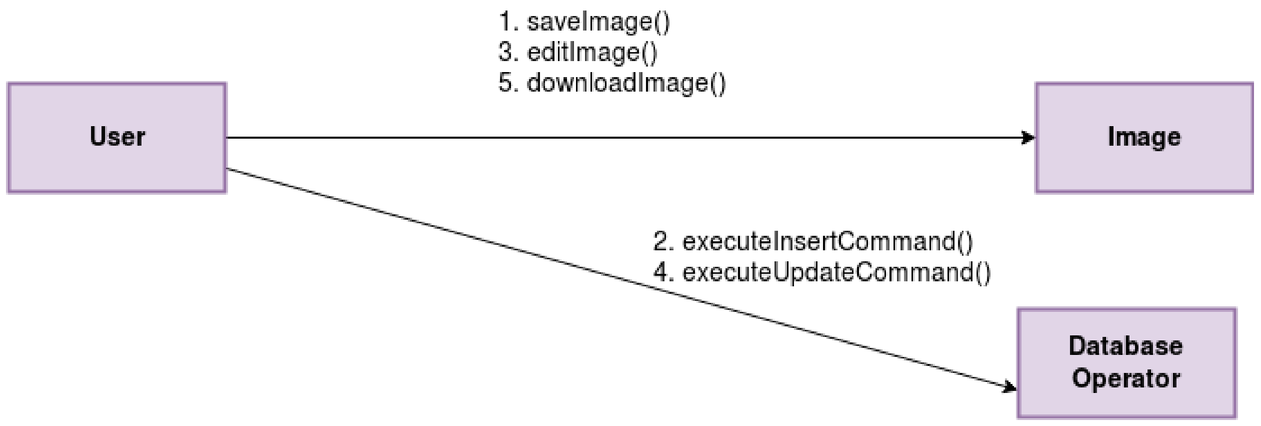
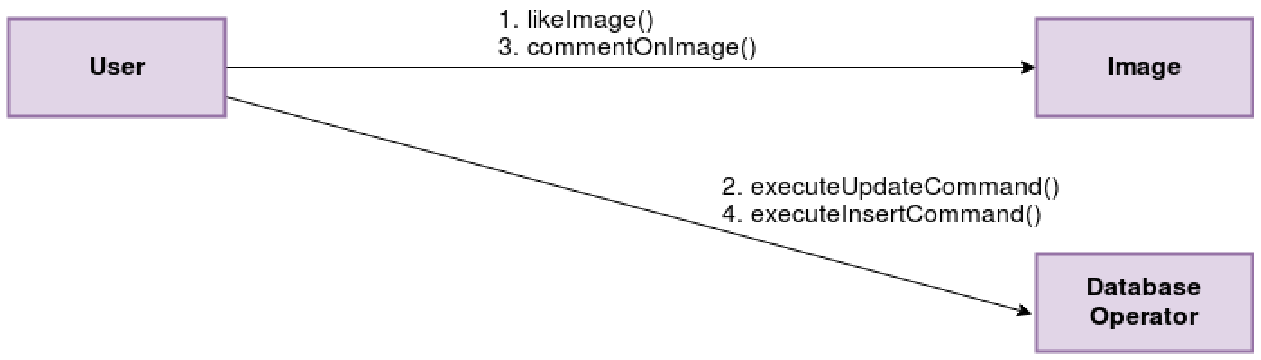


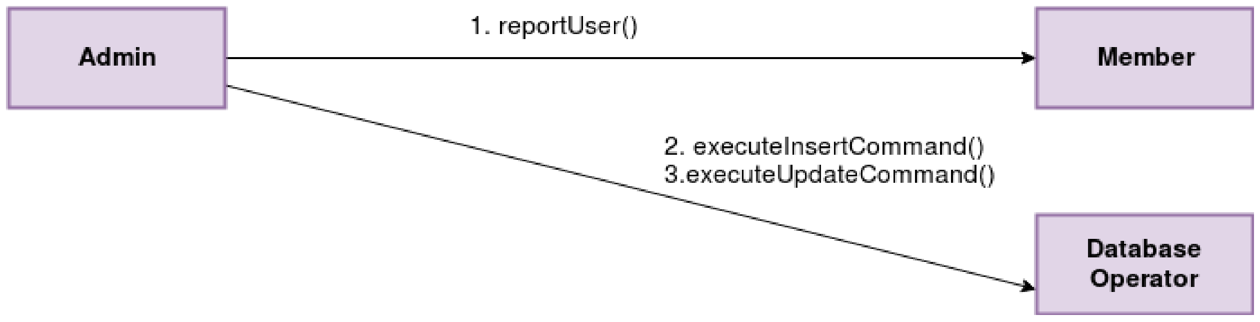
Image Options:



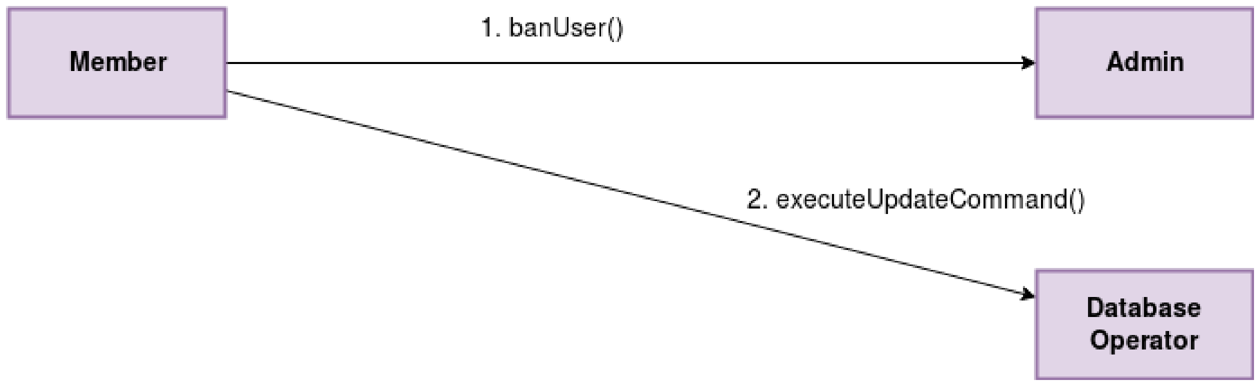
Communication:



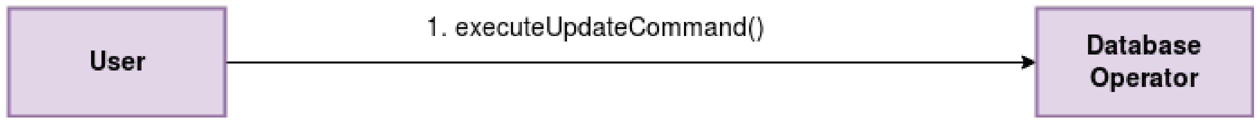
Report User:



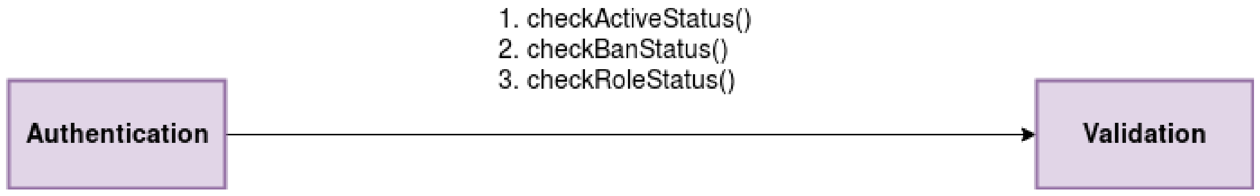
Ban User:



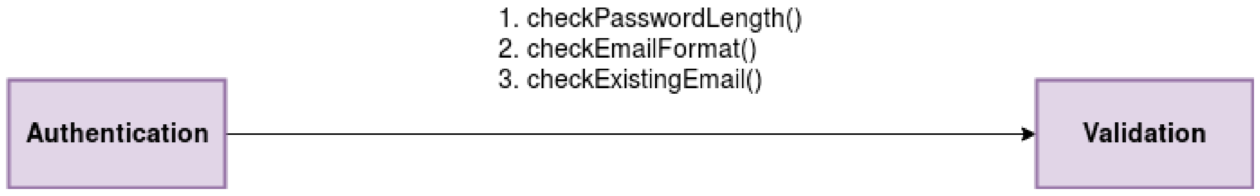
Information Modification:



Information Validation:



Validate Credentials:



Step 3b. Identify appropriate interfaces for each component.

Within the context of component-level design, a UML interface is "a group of externally visible (i.e., public) operations. Every operation within the abstract class (the interface) should be cohesive; that is, it should exhibit processing that focuses on one limited function or subfunction. In our case, there is no necessity to divide the classes in subclasses as they exhibit sufficient cohesion. So there is no need to define appropriate interfaces.

Step 3c. Elaborate attributes and define data types and data structures required to implement them.

In general, data structures and types used to define attributes are defined within the context of the programming language that is to be used for implementation. UML defines an attribute's data type using the following syntax:

name : type-expression initial-value {property string}

Identified Attributes:

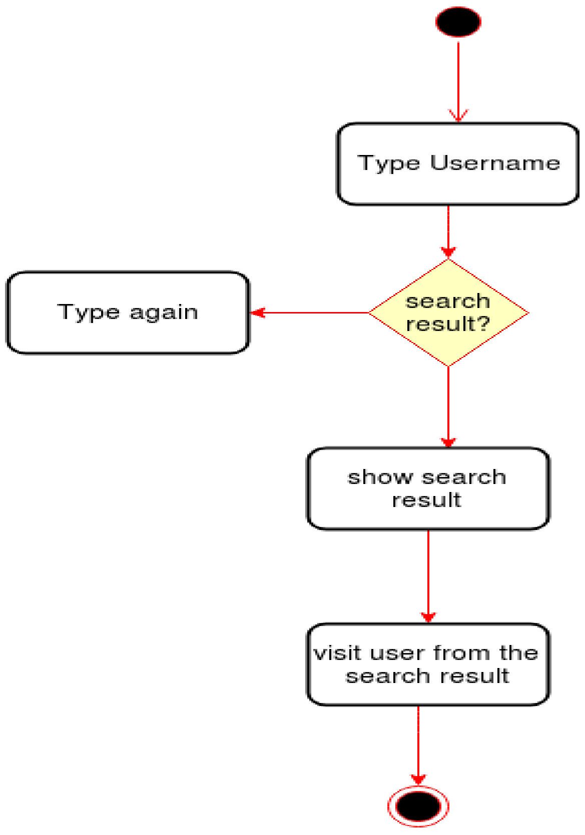
- Name : String = NULL
- Email : String = NULL
- Password : String = NULL { 8<= NumberOfCharacters <= 16 }
- ImageTitle : String = NULL
- ImageURL : String = NULL

Step 3d. Describe processing flow within each operation in detail.

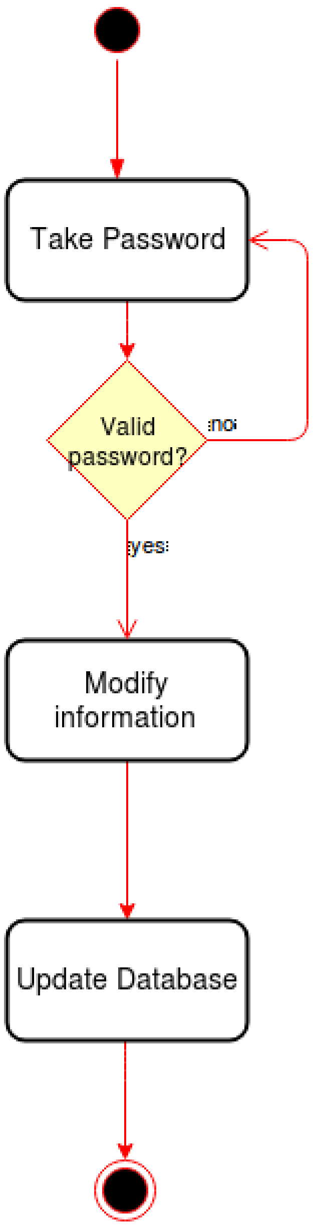
This may be accomplished using a programming language-based pseudocode or with a UML activity diagram.

Methods of User class:

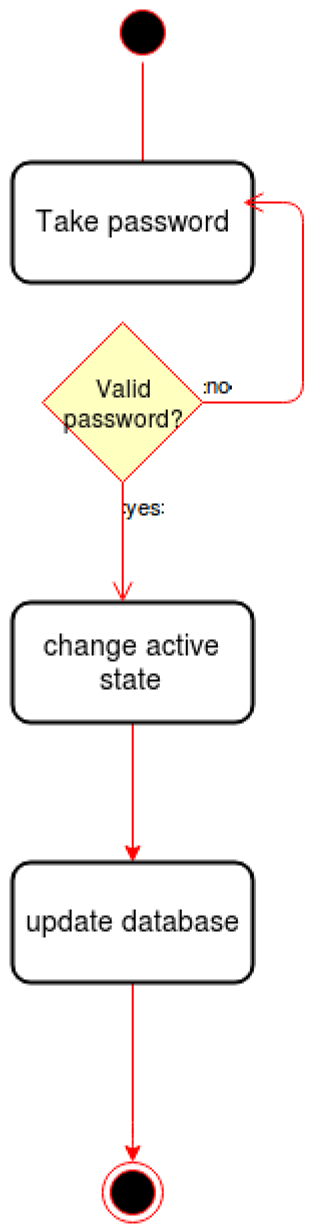
Process flow of searchOtherUser():



Process flow of modifyInformation():

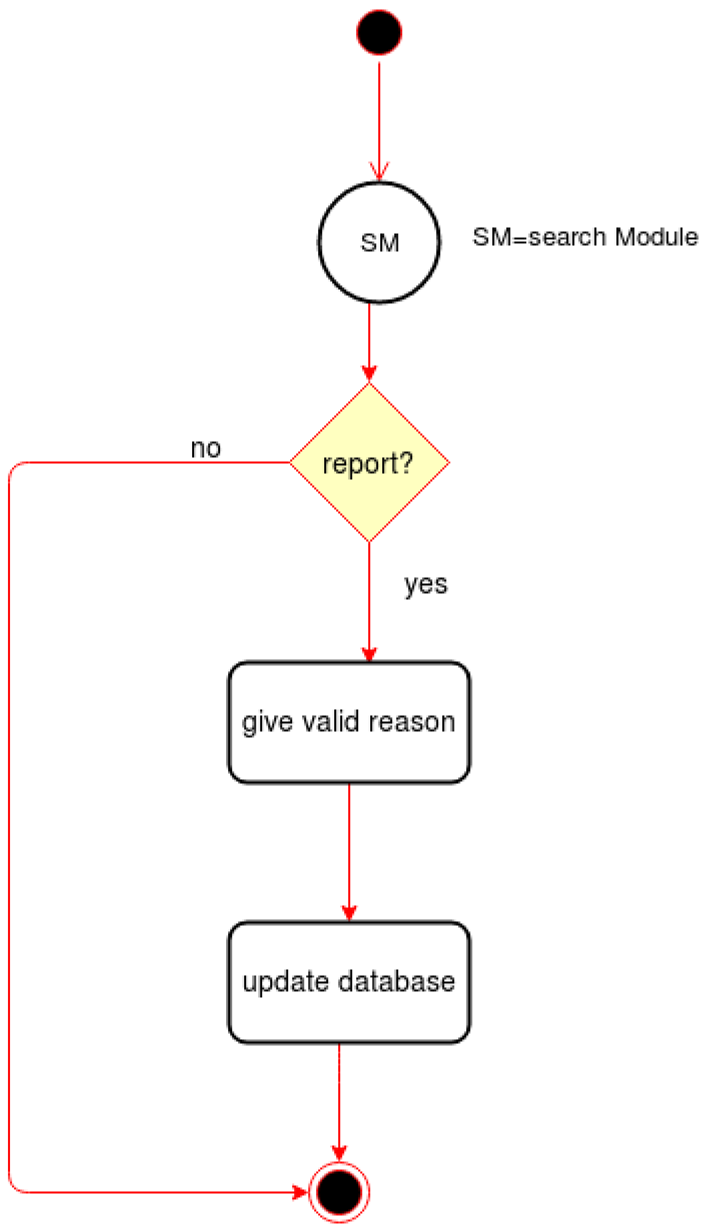


Process flow of deactivateAccount():



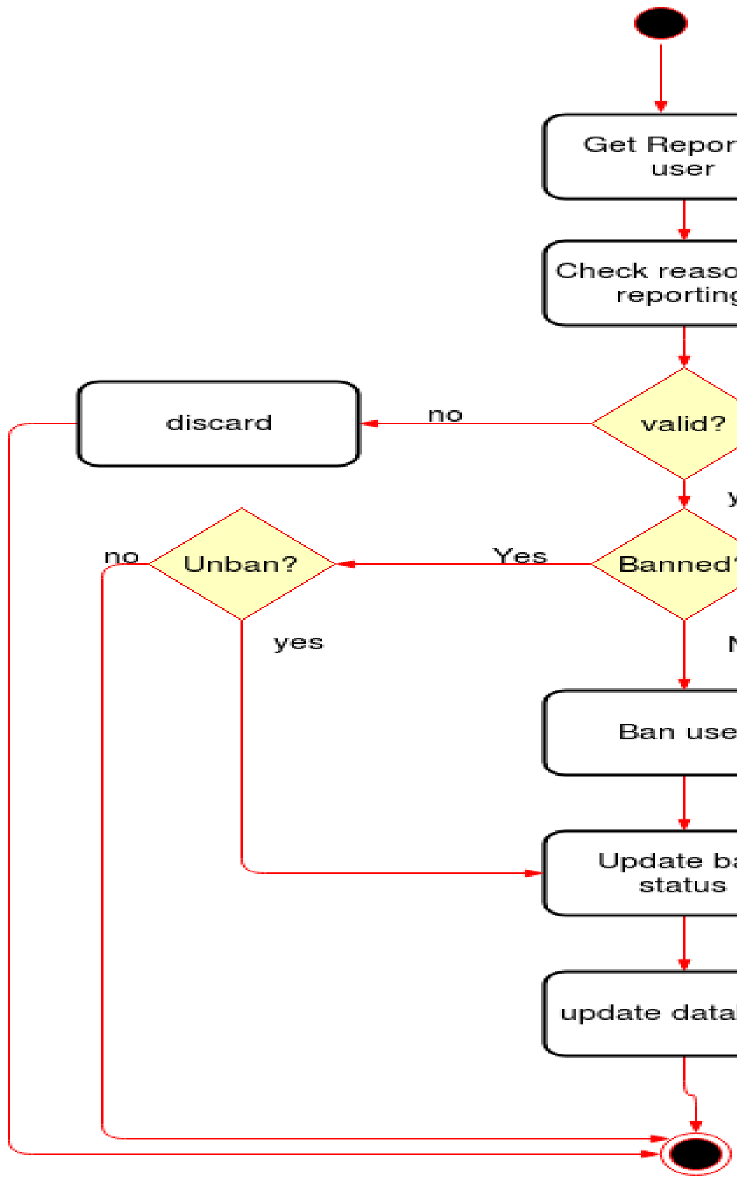
Methods of Member class:

Process flow of reportUser():



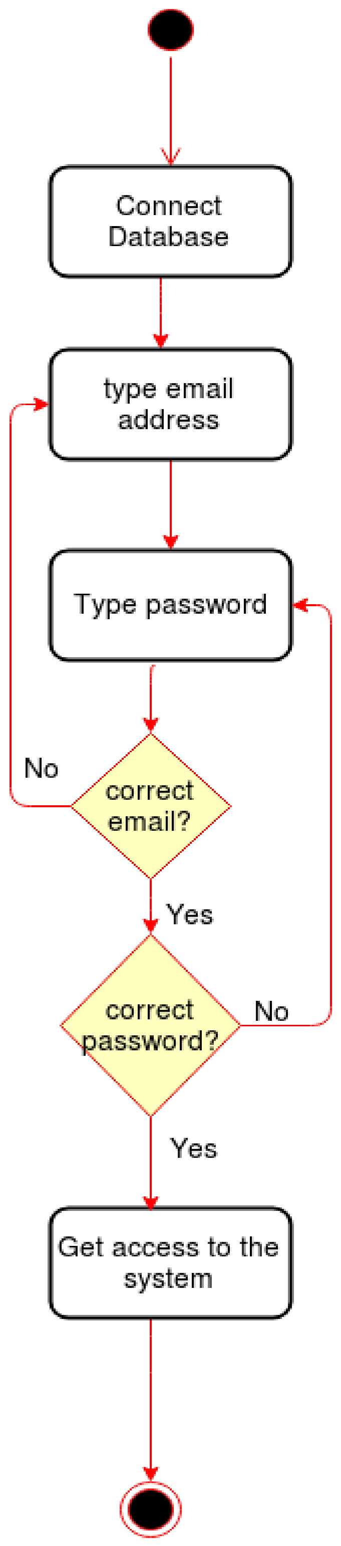
Methods of Admin class:

Process flow of banUser():

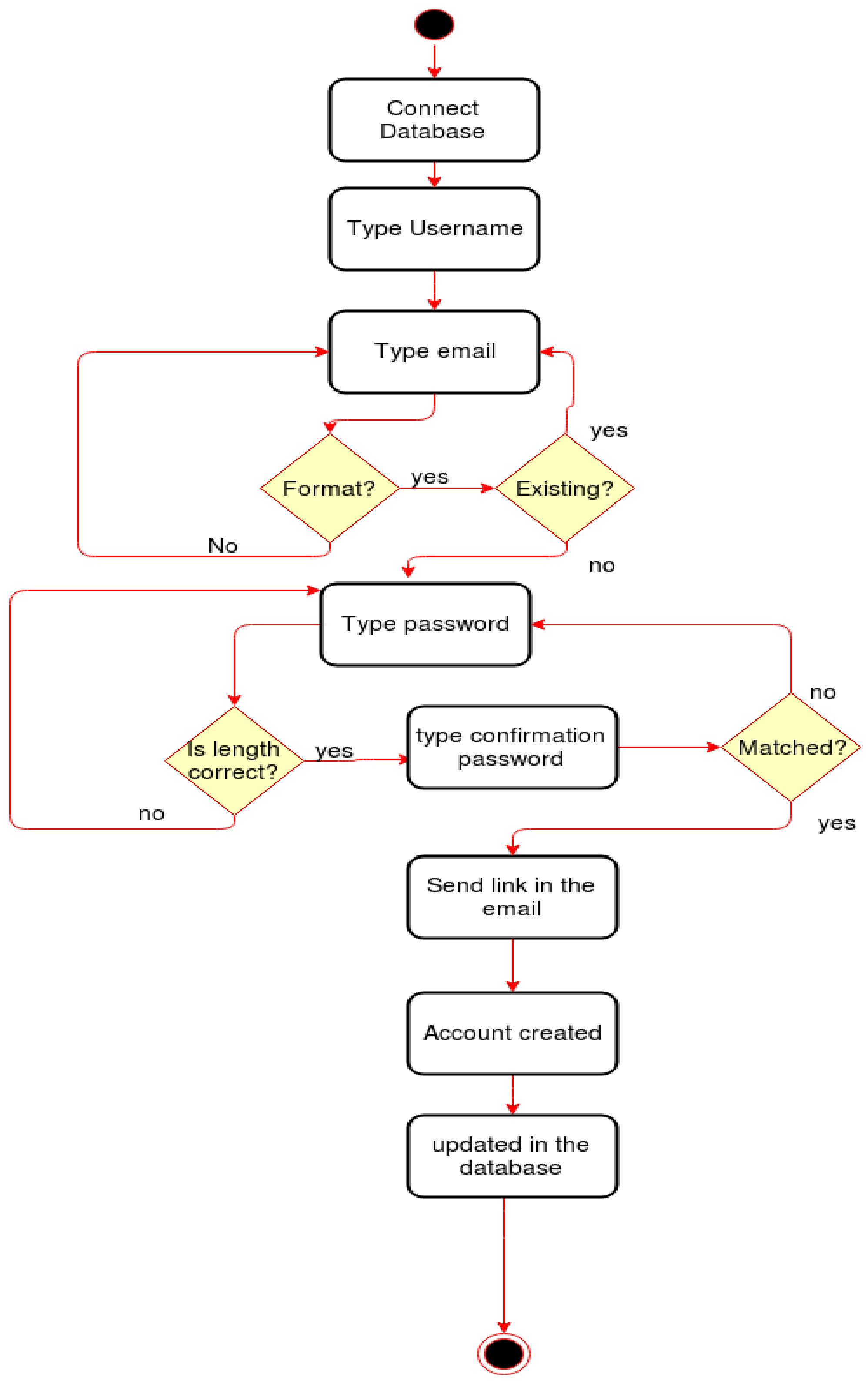


Methods of Authentication Class:

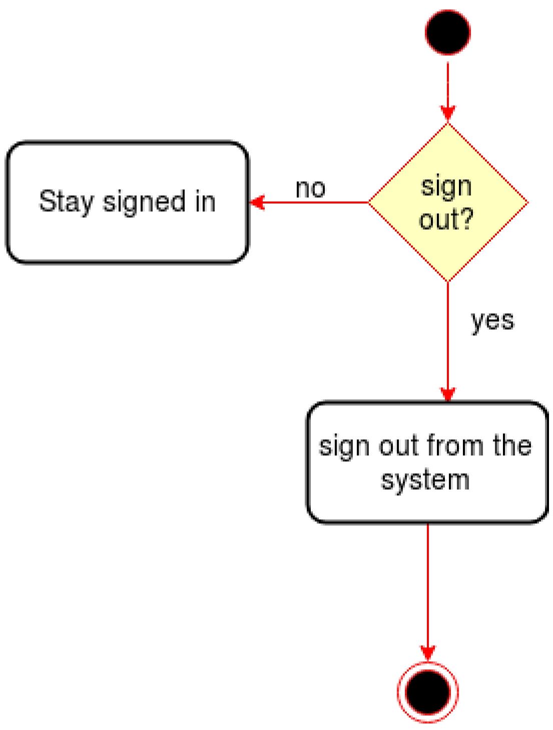
Process flow of signIn():



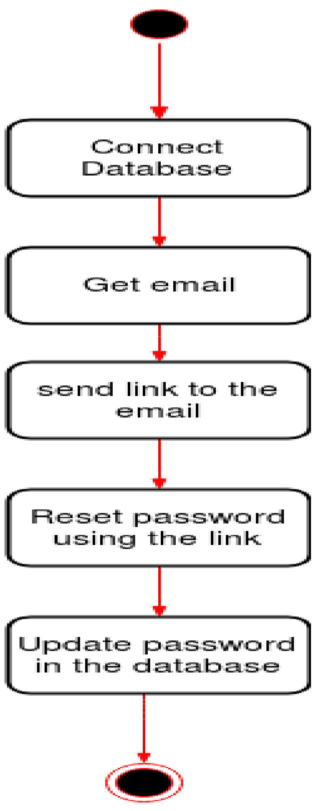
Process flow of signUp():



Process flow of signOut():

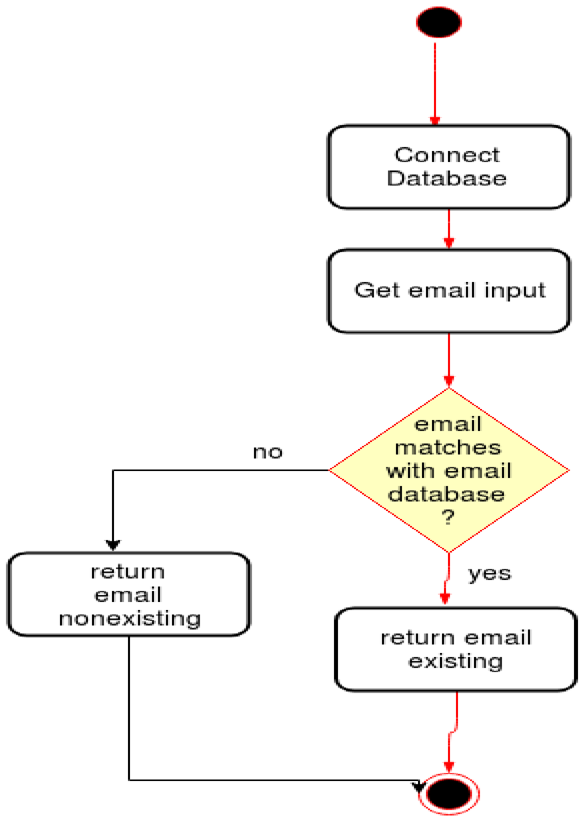


Process flow of forgetPassword():

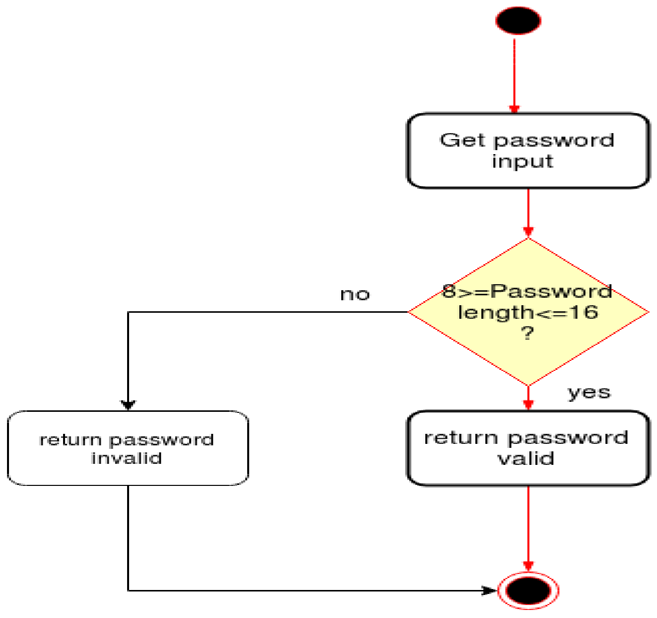

Connect Database
Get email
send link to the email
Reset password using the link
Update password in the database

Methods of Validation Class:

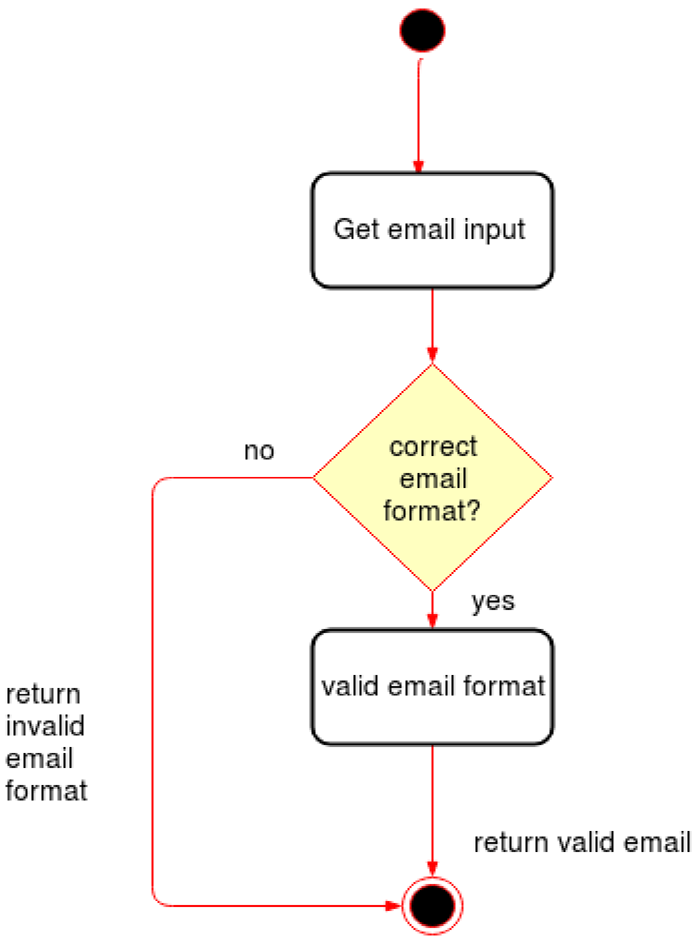
Process flow of isEmailExisting():



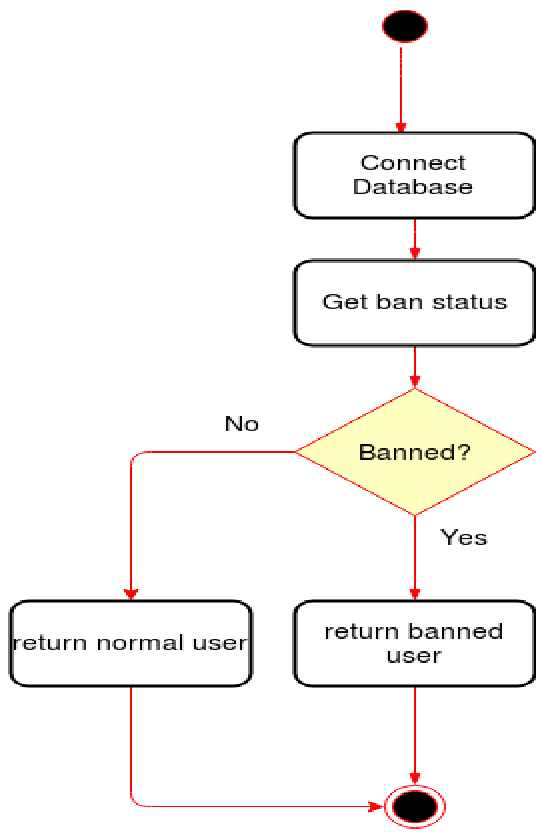
Process flow of checkPasswordLength():



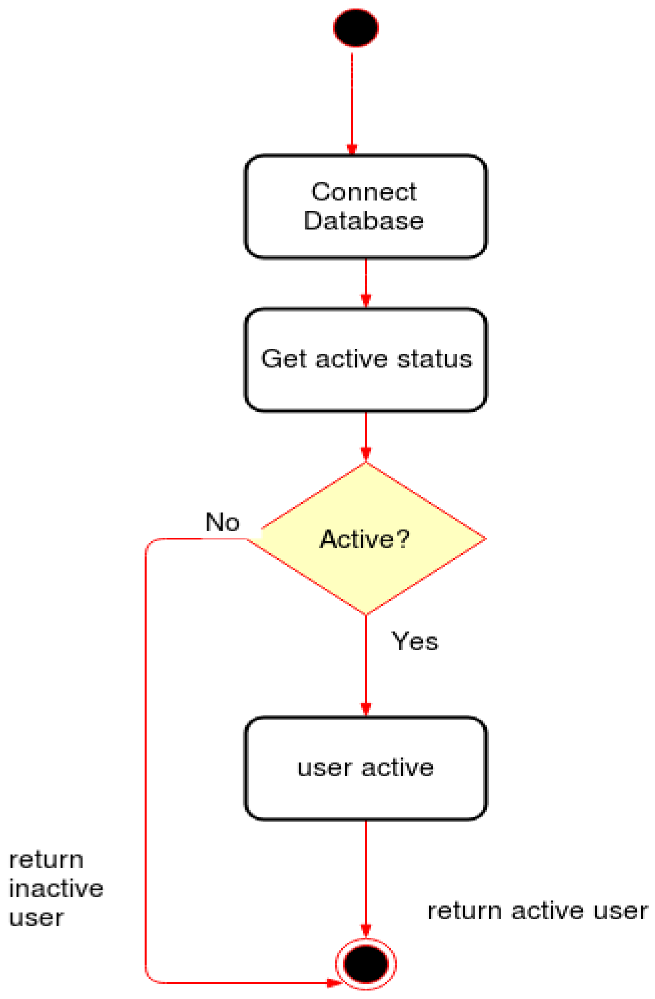
Process flow of checkEmailFormat():



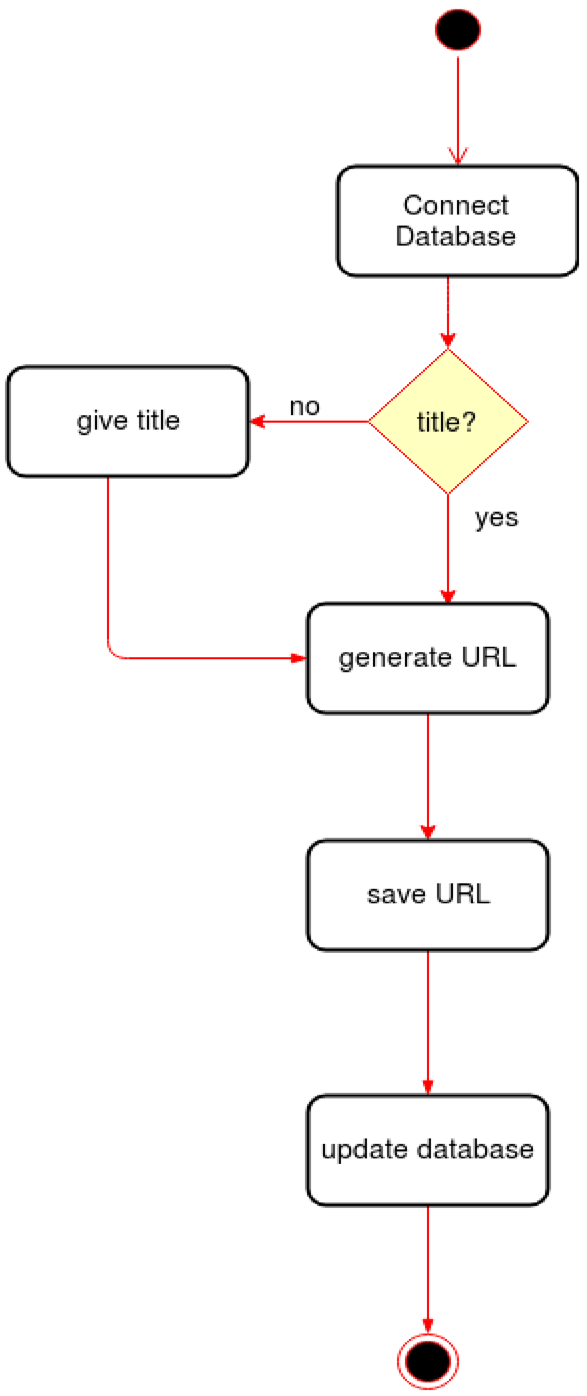
Process flow of checkBanStatus():



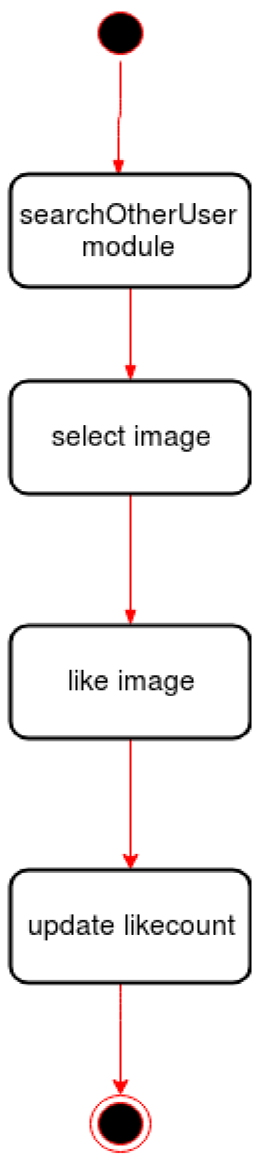
Process flow of checkActiveStatus():



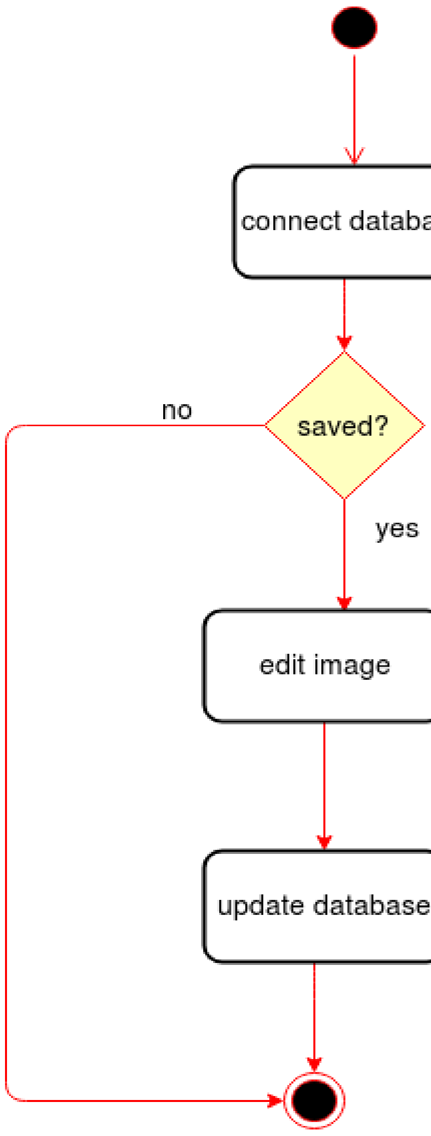
Process flow of saveImage():



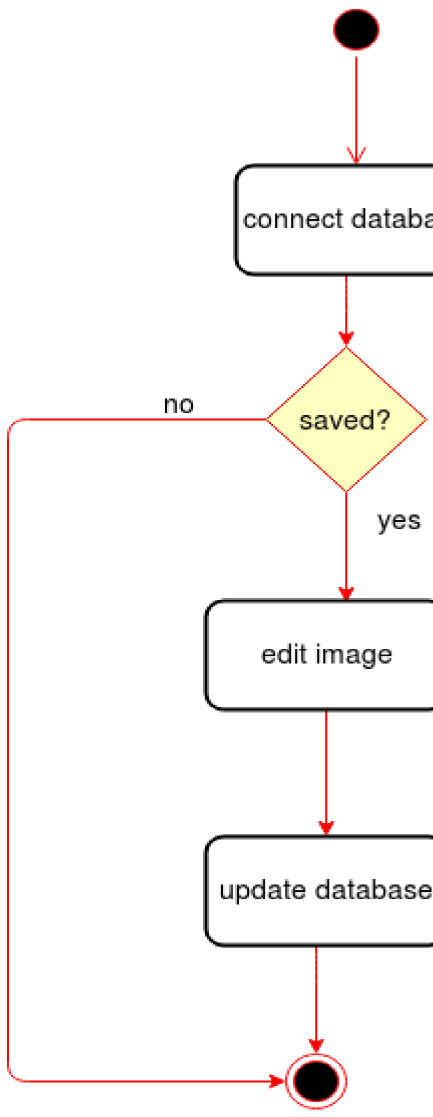
Process flow of likeImage():



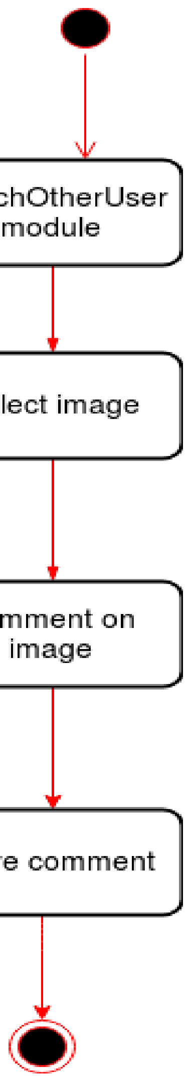
Process flow of editImage():



Process flow of downloadImage():

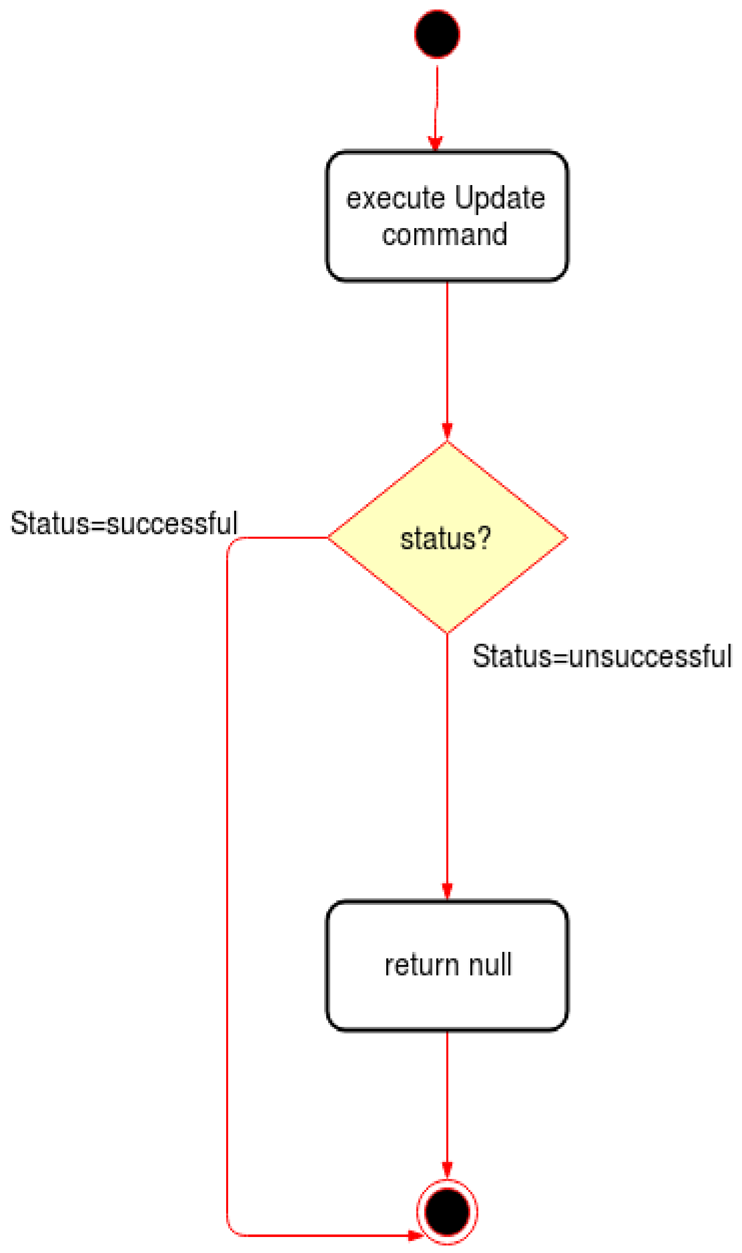


Process flow of commentOnImage():

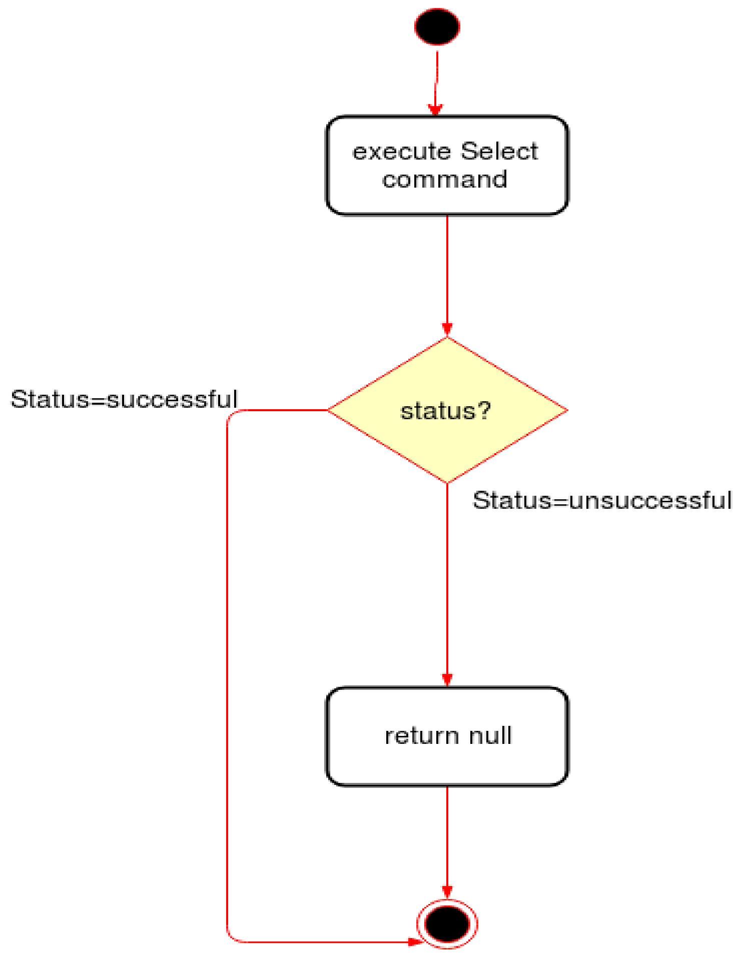


Methods of DatabaseOperator Class:

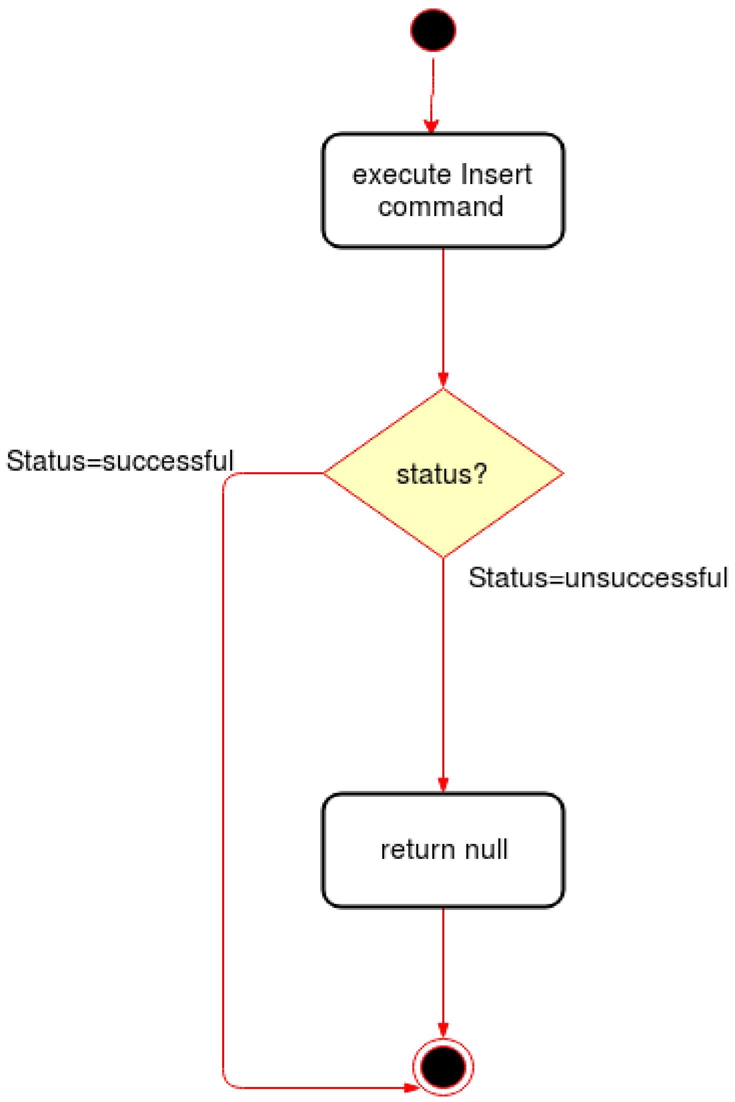
Process flow of executeUpdateCommand():



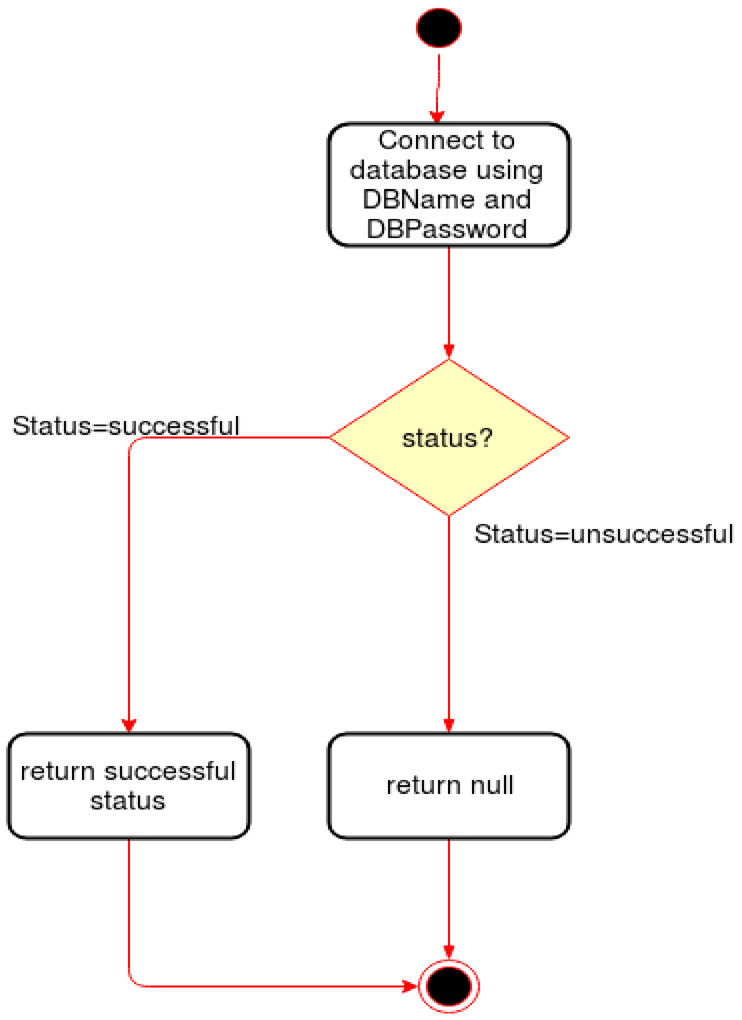
Process flow of executeSelectCommand():



Process flow of executeInsertCommand():



Process flow of connect():



Step 4. Describe persistent data sources (databases and files) and identify the classes required to manage them.

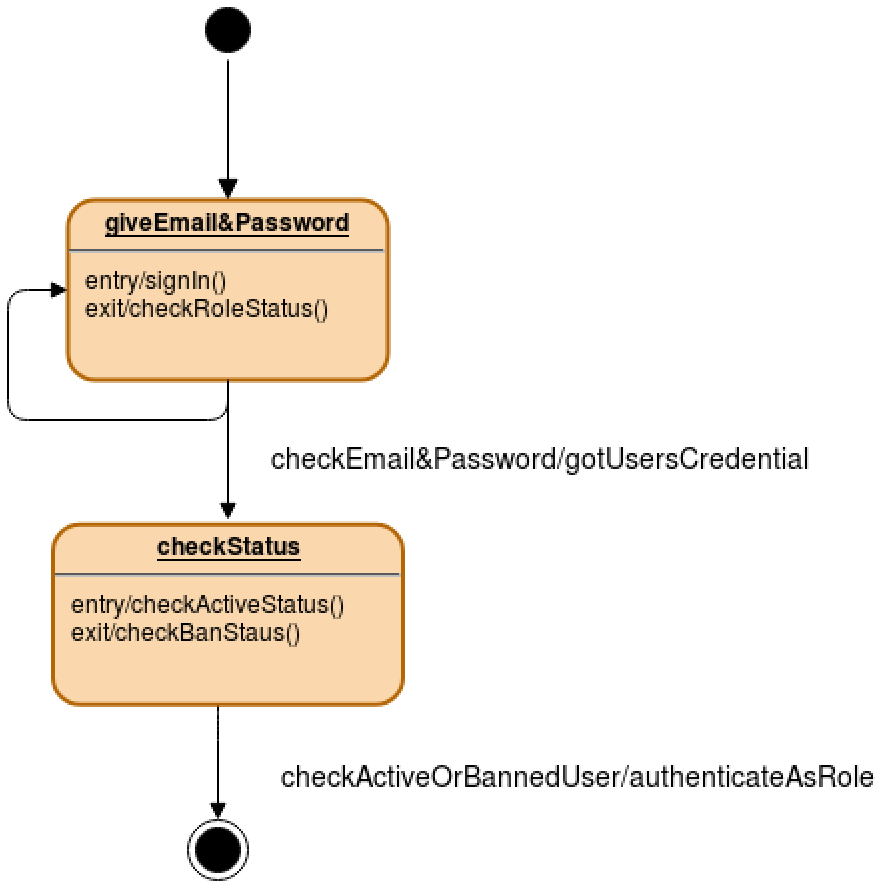
Databases and files normally transcend the design description of an individual component. In most cases, these persistent data stores are initially specified as part of architectural design. However, as design elaboration proceeds, it is often useful to provide additional detail about the structure and organization of these persistent data sources.

MySQL database: An open source relational database management system which is powerful and free that delivers a rich and reliable data store for lightweight Web Sites and desktop applications.

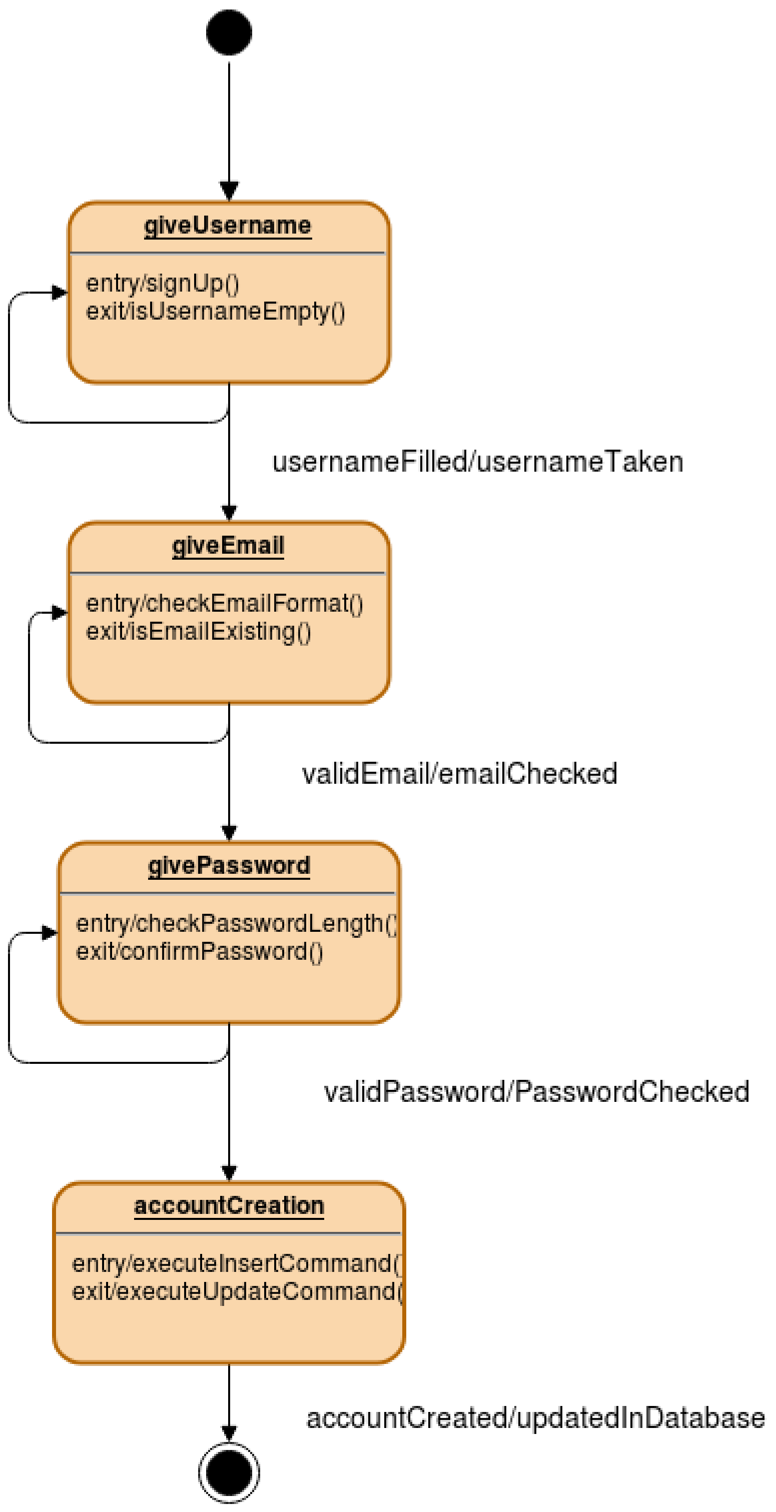
Class required to manage DB: DatabaseOperator

Step 5. Develop and elaborate behavioral representations for a class or component.

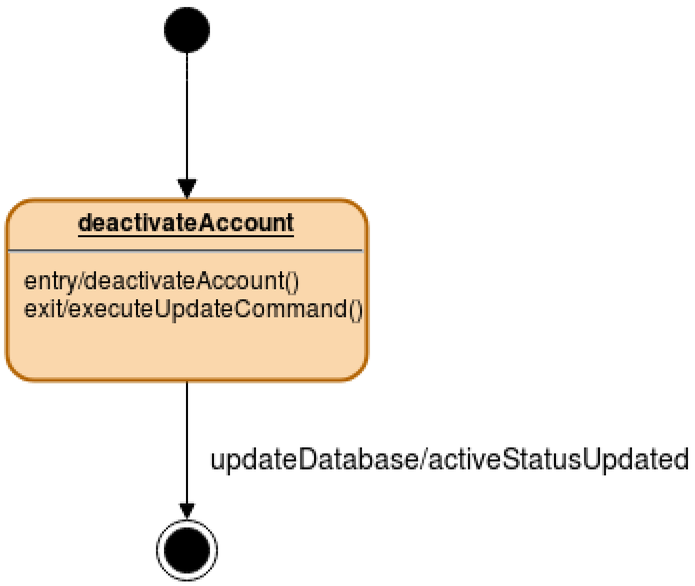
Authentication


giveEmail&Password
entry/signIn()
exit/checkRoleStatus()
checkEmail&Password/gotUsersCredential
checkStatus
entry/checkActiveStatus()
exit/checkBanStaus()
checkActiveOrBannedUser/authenticateAsRole

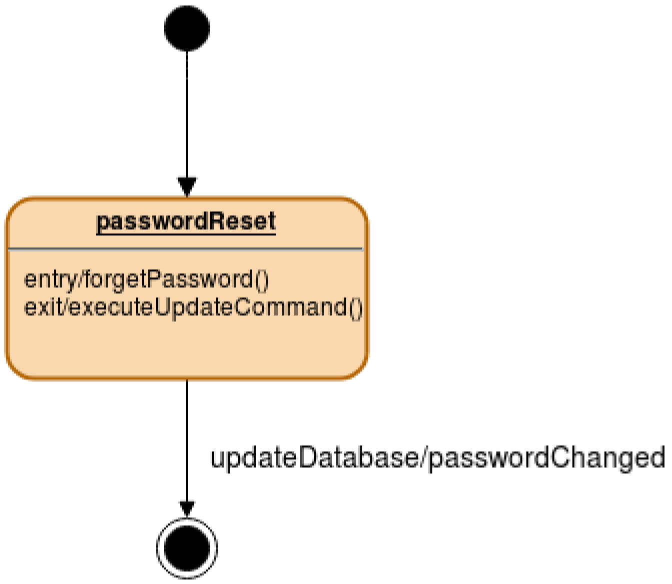
Registration



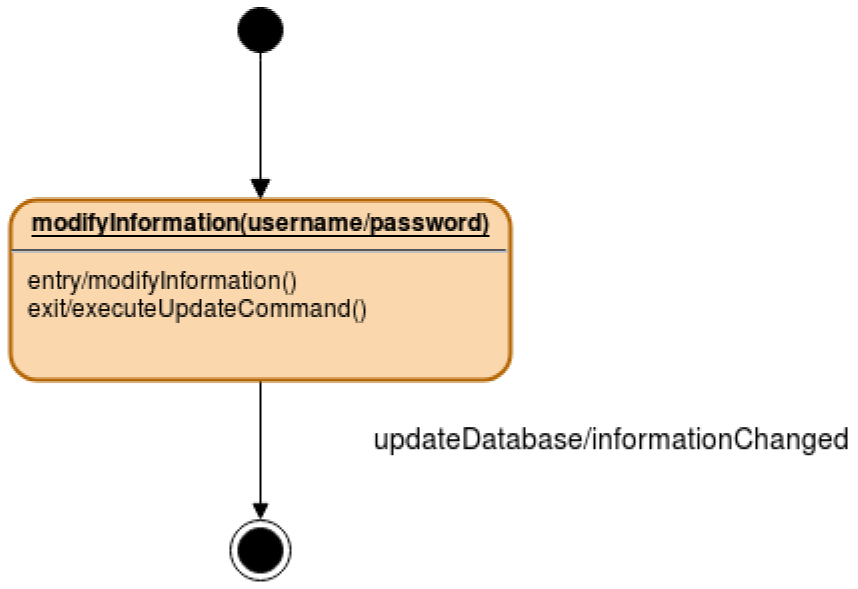
DeactivateAccount


deactivateAccount
entry/deactivateAccount()
exit/executeUpdateCommand()
updateDatabase/activeStatusUpdated

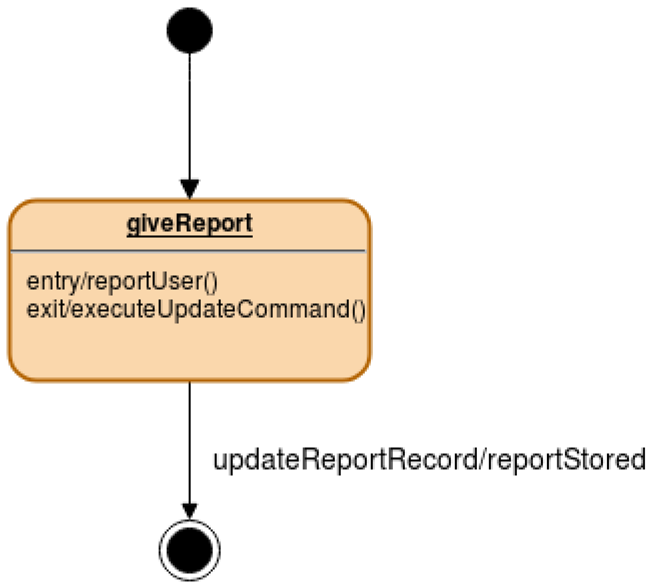
Forget Password



Modify Information


modifyInformation(username/password)
entry/modifyInformation()
exit/executeUpdateCommand()
updateDatabase/informationChanged

Report



Ban

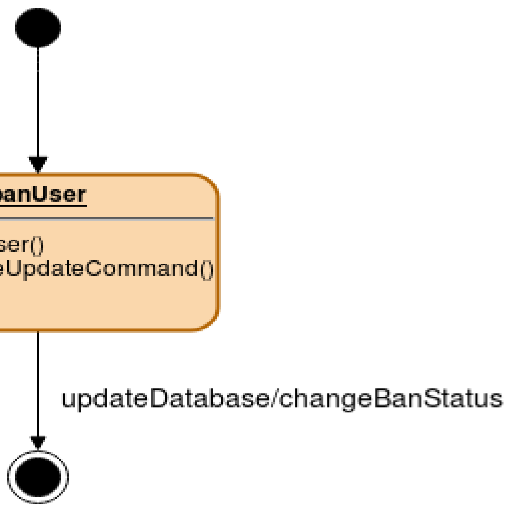
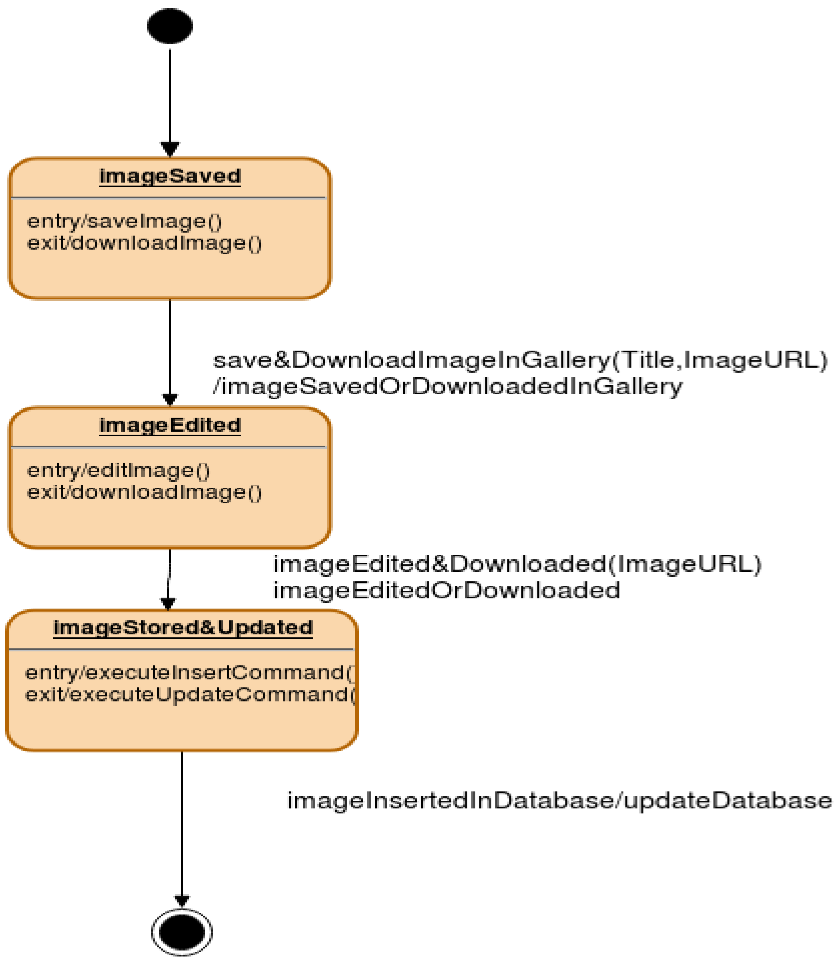
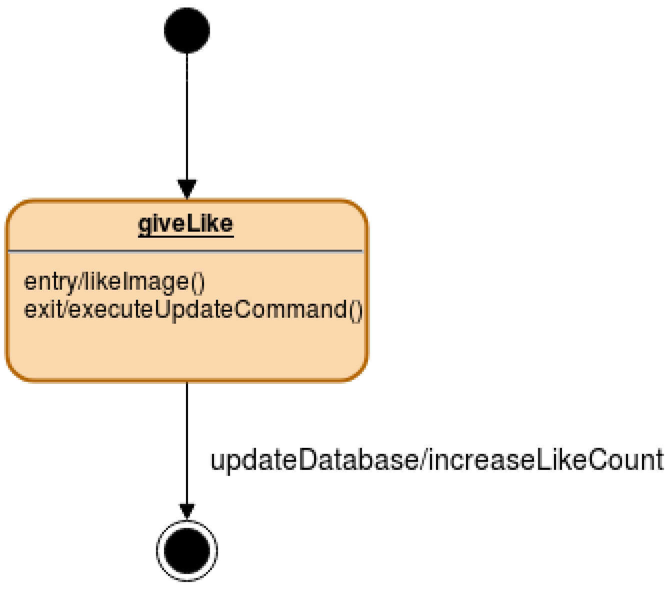


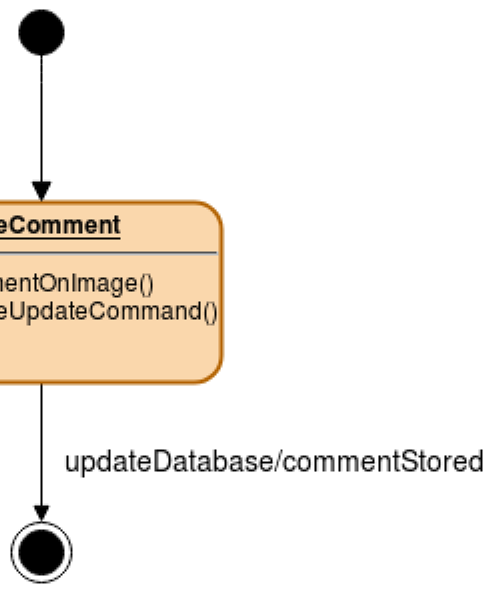
Image Options


imageSaved
entry/saveImage()
exit/downloadImage()
save&DownloadImageInGallery(Title,ImageURL)
/imageSavedOrDownloadedInGallery
imageEdited
entry/editImage()
exit/downloadImage()
imageEdited&Downloaded(ImageURL)
imageEditedOrDownloaded
imageStored&Updated
entry/executeInsertCommand(
exit/executeUpdateCommand(
imageInsertedInDatabase/updateDatabase

Like Image

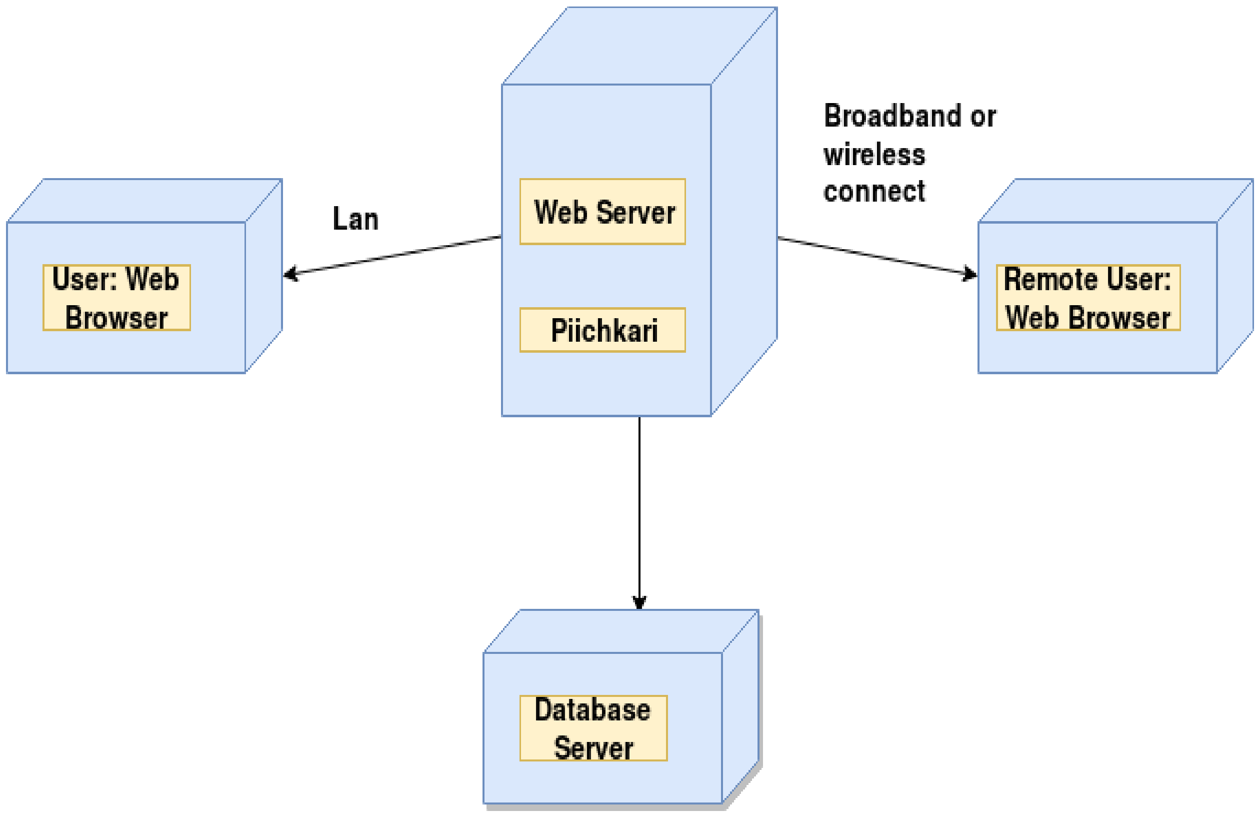

giveLike
entry/likeImage()
exit/executeUpdateCommand()
updateDatabase/increaseLikeCount

Comment on Image



Step 6. Elaborate deployment diagrams to provide additional implementation detail.

Deployment diagrams are used as part of architectural design and are represented in descriptor form. In this form, major system functions (often represented as subsystems) are represented within the context of the computing environment that will house them.


Lan
Web Server
Piichkari
Broadband or wireless connect
User: Web Browser
Remote User: Web Browser
Database Server