# CSCI-4448/5448: Object Oriented Programming
# Final Project Report : Find Me A Job

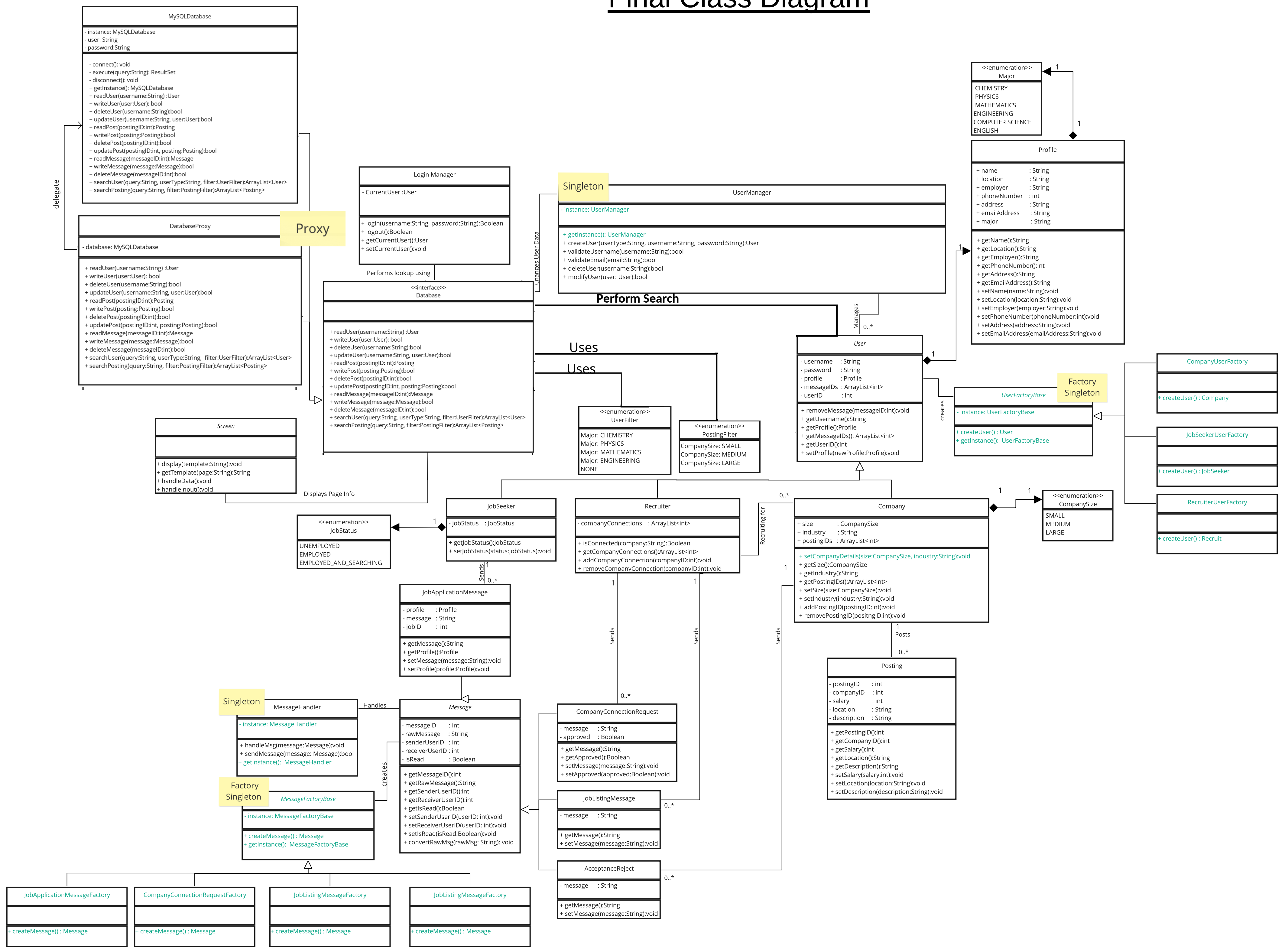
Team # 33
Team Members: Justin Chin, Ryan Baten, Zaid Alali

## Features

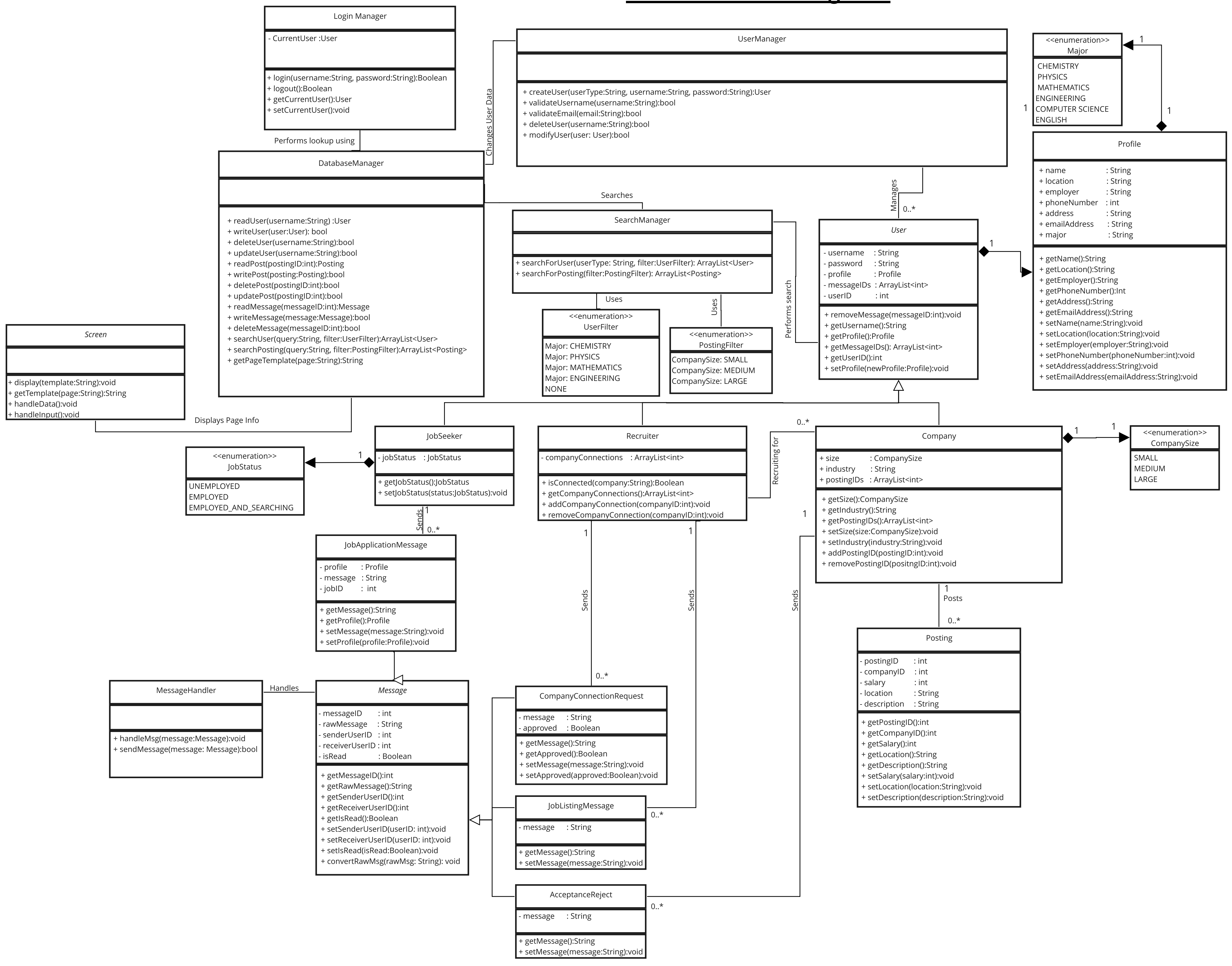
| Features Implemented | |
|---|---|
| ID | Title |
| UC-006 | Search Profile |
| UC-016 | Search for Job Listings |
| UC-019 | View Job Listing |
| UC-018 | Apply for Job Listing |
| UC-025 | View Company Job Listing |
| UC-024 | Send Job Listing |

| Features Not Implemented | |
|---|---|
| ID | Title |
| UC-003 | Sign Up |
| UC-001 | Log in |
| UC-023 | Send Company Connection Request |
| UC-013 | Send Accept/Reject Message |

# Final Class Diagram


MySQLDatabase
- instance: MySQLDatabase
- user: String
- password:String
- connect(): void
- execute(query:String): ResultSet
- disconnect(): void
+ getInstance(): MySQLDatabase
+ readUser(username:String) :User
+ writeUser(user:User): bool
+ deleteUser(username:String):bool
+ updateUser(username:String, user:User):bool
+ readPost(postingID:int):Posting
+ writePost(posting:Posting):bool
+ deletePost(postingID:int):bool
+ updatePost(postingID:int, posting:Posting):bool
+ readMessage(messageID:int):Message
+ writeMessage(message:Message):bool
+ deleteMessage(messageID:int):bool
+ searchUser(query:String, userType:String, filter:UserFilter):ArrayList<User>
+ searchPosting(query:String, filter:PostingFilter):ArrayList<Posting>
delegate
Proxy
DatabaseProxy
- database: MySQLDatabase
+ readUser(username:String) :User
+ writeUser(user:User): bool
+ deleteUser(username:String):bool
+ updateUser(username:String, user:User):bool
+ readPost(postingID:int):Posting
+ writePost(posting:Posting):bool
+ deletePost(postingID:int):bool
+ updatePost(postingID:int, posting:Posting):bool
+ readMessage(messageID:int):Message
+ writeMessage(message:Message):bool
+ deleteMessage(messageID:int):bool
+ searchUser(query:String, userType:String, filter:UserFilter):ArrayList<User>
+ searchPosting(query:String, filter:PostingFilter):ArrayList<Posting>
Login Manager
- CurrentUser :User
+ login(username:String, password:String):Boolean
+ logout():Boolean
+ getCurrentUser():User
+ setCurrentUser():void
Performs lookup using
<<interface>>
Database
+ readUser(username:String) :User
+ writeUser(user:User): bool
+ deleteUser(username:String):bool
+ updateUser(username:String, user:User):bool
+ readPost(postingID:int):Posting
+ writePost(posting:Posting):bool
+ deletePost(postingID:int):bool
+ updatePost(postingID:int, posting:Posting):bool
+ readMessage(messageID:int):Message
+ writeMessage(message:Message):bool
+ deleteMessage(messageID:int):bool
+ searchUser(query:String, userType:String, filter:UserFilter):ArrayList<User>
+ searchPosting(query:String, filter:PostingFilter):ArrayList<Posting>
Changes User Data
Singleton
UserManager
- instance: UserManager
+ getInstance(): UserManager
+ createUser(userType:String, username:String, password:String):User
+ validateUsername(username:String):bool
+ validateEmail(email:String):bool
+ deleteUser(username:String):bool
+ modifyUser(user: User):bool
Perform Search
Uses
Uses
<<enumeration>>
UserFilter
Major: CHEMISTRY
Major: PHYSICS
Major: MATHEMATICS
Major: ENGINEERING
NONE
<<enumeration>>
PostingFilter
CompanySize: SMALL
CompanySize: MEDIUM
CompanySize: LARGE
Screen
+ display(template:String):void
+ getTemplate(page:String):String
+ handleData():void
+ handleInput():void
Displays Page Info
<<enumeration>>
Major
CHEMISTRY
PHYSICS
MATHEMATICS
ENGINEERING
COMPUTER SCIENCE
ENGLISH
1
1
Profile
+ name : String
+ location : String
+ employer : String
+ phoneNumber : int
+ address : String
+ emailAddress : String
+ major : String
+ getName():String
+ getLocation():String
+ getEmployer():String
+ getPhoneNumber():Int
+ getAddress():String
+ getEmailAddress():String
+ setName(name:String):void
+ setLocation(location:String):void
+ setEmployer(employer:String):void
+ setPhoneNumber(phoneNumber:int):void
+ setAddress(address:String):void
+ setEmailAddress(emailAddress:String):void
1
Manages
0..*
User
- username : String
- password : String
- profile : Profile
- messageIDs : ArrayList<int>
- userID : int
+ removeMessage(messageID:int):void
+ getUsername():String
+ getProfile():Profile
+ getMessageIDs(): ArrayList<int>
+ getUserID():int
+ setProfile(newProfile:Profile):void
1
creates
Factory
Singleton
UserFactoryBase
- instance: UserFactoryBase
+ createUser() : User
+ getInstance(): UserFactoryBase
CompanyUserFactory
+ createUser() : Company
JobSeekerUserFactory
+ createUser() : JobSeeker
RecruiterUserFactory
+ createUser() : Recruit
<<enumeration>>
JobStatus
UNEMPLOYED
EMPLOYED
EMPLOYED_AND_SEARCHING
1
JobSeeker
- jobStatus : JobStatus
+ getJobStatus():JobStatus
+ setJobStatus(status:JobStatus):void
Sends
1
0..*
Recruiter
- companyConnections : ArrayList<int>
+ isConnected(company:String):Boolean
+ getCompanyConnections():ArrayList<int>
+ addCompanyConnection(companyID:int):void
+ removeCompanyConnection(companyID:int):void
Recruiting for
0..*
1
Company
+ size : CompanySize
+ industry : String
+ postingIDs : ArrayList<int>
+ setCompanyDetails(size:CompanySize, industry:String):void
+ getSize():CompanySize
+ getIndustry():String
+ getPostingIDs():ArrayList<int>
+ setSize(size:CompanySize):void
+ setIndustry(industry:String):void
+ addPostingID(postingID:int):void
+ removePostingID(positngID:int):void
1
1
<<enumeration>>
CompanySize
SMALL
MEDIUM
LARGE
1
Posts
0..*
Posting
- postingID : int
- companyID : int
- salary : int
- location : String
- description : String
+ getPostingID():int
+ getCompanyID():int
+ getSalary():int
+ getLocation():String
+ getDescription():String
+ setSalary(salary:int):void
+ setLocation(location:String):void
+ setDescription(description:String):void
JobApplicationMessage
- profile : Profile
- message : String
- jobID : int
+ getMessage():String
+ getProfile():Profile
+ setMessage(message:String):void
+ setProfile(profile:Profile):void
1
1
Sends
Sends
Sends
1
0..*
Singleton
MessageHandler
- instance: MessageHandler
+ handleMsg(message:Message):void
+ sendMessage(message: Message):bool
+ getInstance(): MessageHandler
Handles
Message
- messageID : int
- rawMessage : String
- senderUserID : int
- receiverUserID : int
- isRead : Boolean
+ getMessageID():int
+ getRawMessage():String
+ getSenderUserID():int
+ getReceiverUserID():int
+ getIsRead():Boolean
+ setSenderUserID(userID: int):void
+ setReceiverUserID(userID: int):void
+ setIsRead(isRead:Boolean):void
+ convertRawMsg(rawMsg: String): void
creates
Factory
Singleton
MessageFactoryBase
- instance: MessageFactoryBase
+ createMessage() : Message
+ getInstance(): MessageFactoryBase
CompanyConnectionRequest
- message : String
- approved : Boolean
+ getMessage():String
+ getApproved():Boolean
+ setMessage(message:String):void
+ setApproved(approved:Boolean):void
JobListingMessage
- message : String
+ getMessage():String
+ setMessage(message:String):void
0..*
AcceptanceReject
- message : String
+ getMessage():String
+ setMessage(message:String):void
0..*
JobApplicationMessageFactory
+ createMessage() : Message
CompanyConnectionRequestFactory
+ createMessage() : Message
JobListingMessageFactory
+ createMessage() : Message
JobListingMessageFactory
+ createMessage() : Message

# Part 2 Class Diagram


Login Manager
- CurrentUser :User
+ login(username:String, password:String):Boolean
+ logout():Boolean
+ getCurrentUser():User
+ setCurrentUser():void
Performs lookup using
DatabaseManager
+ readUser(username:String) :User
+ writeUser(user:User): bool
+ deleteUser(username:String):bool
+ updateUser(username:String):bool
+ readPost(postingID:int):Posting
+ writePost(posting:Posting):bool
+ deletePost(postingID:int):bool
+ updatePost(postingID:int):bool
+ readMessage(messageID:int):Message
+ writeMessage(message:Message):bool
+ deleteMessage(messageID:int):bool
+ searchUser(query:String, filter:UserFilter):ArrayList&lt;User&gt;
+ searchPosting(query:String, filter:PostingFilter):ArrayList&lt;Posting&gt;
+ getPageTemplate(page:String):String
Changes User Data
UserManager
+ createUser(userType:String, username:String, password:String):User
+ validateUsername(username:String):bool
+ validateEmail(email:String):bool
+ deleteUser(username:String):bool
+ modifyUser(user: User):bool
Searches
SearchManager
+ searchForUser(userType: String, filter:UserFilter): ArrayList&lt;User&gt;
+ searchForPosting(filter:PostingFilter): ArrayList&lt;Posting&gt;
Uses
Uses
&lt;&lt;enumeration&gt;&gt;
UserFilter
Major: CHEMISTRY
Major: PHYSICS
Major: MATHEMATICS
Major: ENGINEERING
NONE
&lt;&lt;enumeration&gt;&gt;
PostingFilter
CompanySize: SMALL
CompanySize: MEDIUM
CompanySize: LARGE
Performs search
Manages
0..*
User
- username : String
- password : String
- profile : Profile
- messageIDs : ArrayList&lt;int&gt;
- userID : int
+ removeMessage(messageID:int):void
+ getUsername():String
+ getProfile():Profile
+ getMessageIDs(): ArrayList&lt;int&gt;
+ getUserID():int
+ setProfile(newProfile:Profile):void
1
&lt;&lt;enumeration&gt;&gt;
Major
CHEMISTRY
PHYSICS
MATHEMATICS
ENGINEERING
COMPUTER SCIENCE
ENGLISH
1
1
1
Profile
+ name : String
+ location : String
+ employer : String
+ phoneNumber : int
+ address : String
+ emailAddress : String
+ major : String
+ getName():String
+ getLocation():String
+ getEmployer():String
+ getPhoneNumber():Int
+ getAddress():String
+ getEmailAddress():String
+ setName(name:String):void
+ setLocation(location:String):void
+ setEmployer(employer:String):void
+ setPhoneNumber(phoneNumber:int):void
+ setAddress(address:String):void
+ setEmailAddress(emailAddress:String):void
Screen
+ display(template:String):void
+ getTemplate(page:String):String
+ handleData():void
+ handleInput():void
Displays Page Info
&lt;&lt;enumeration&gt;&gt;
JobStatus
UNEMPLOYED
EMPLOYED
EMPLOYED_AND_SEARCHING
1
JobSeeker
- jobStatus : JobStatus
+ getJobStatus():JobStatus
+ setJobStatus(status:JobStatus):void
Sends
1
0..*
Recruiter
- companyConnections : ArrayList&lt;int&gt;
+ isConnected(company:String):Boolean
+ getCompanyConnections():ArrayList&lt;int&gt;
+ addCompanyConnection(companyID:int):void
+ removeCompanyConnection(companyID:int):void
Recruiting for
0..*
Company
+ size : CompanySize
+ industry : String
+ postingIDs : ArrayList&lt;int&gt;
+ getSize():CompanySize
+ getIndustry():String
+ getPostingIDs():ArrayList&lt;int&gt;
+ setSize(size:CompanySize):void
+ setIndustry(string):void
+ addPostingID(postingID:int):void
+ removePostingID(positngID:int):void
1
1
&lt;&lt;enumeration&gt;&gt;
CompanySize
SMALL
MEDIUM
LARGE
1
Posts
0..*
Posting
- postingID : int
- companyID : int
- salary : int
- location : String
- description : String
+ getPostingID():int
+ getCompanyID():int
+ getSalary():int
+ getLocation():String
+ getDescription():String
+ setSalary(salary:int):void
+ setLocation(location:String):void
+ setDescription(description:String):void
JobApplicationMessage
- profile : Profile
- message : String
- jobID : int
+ getMessage():String
+ getProfile():Profile
+ setMessage(message:String):void
+ setProfile(profile:Profile):void
MessageHandler
+ handleMsg(message:Message):void
+ sendMessage(message: Message):bool
Handles
Message
- messageID : int
- rawMessage : String
- senderUserID : int
- receiverUserID : int
- isRead : Boolean
+ getMessageID():int
+ getRawMessage():String
+ getSenderUserID():int
+ getReceiverUserID():int
+ getIsRead():Boolean
+ setSenderUserID(userID: int):void
+ setReceiverUserID(userID: int):void
+ setIsRead(isRead:Boolean):void
+ convertRawMsg(rawMsg: String): void
1
Sends
0..*
1
Sends
1
Sends
CompanyConnectionRequest
- message : String
- approved : Boolean
+ getMessage():String
+ getApproved():Boolean
+ setMessage(message:String):void
+ setApproved(approved:Boolean):void
JobListingMessage
- message : String
+ getMessage():String
+ setMessage(message:String):void
0..*
AcceptanceReject
- message : String
+ getMessage():String
+ setMessage(message:String):void
0..*

## Refactoring the Class Diagram

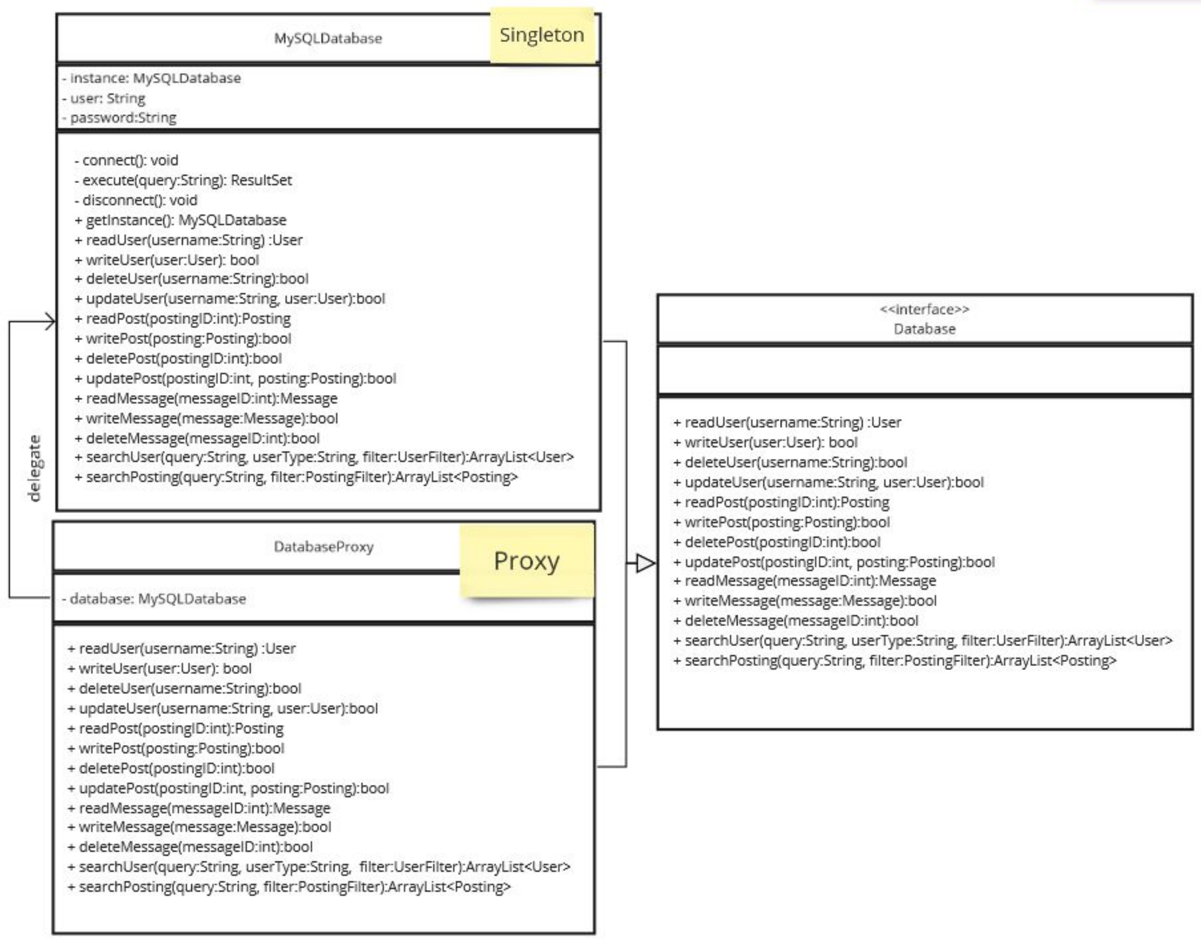
We used the Proxy, Factory and Singleton design patterns. The Proxy design pattern made the database part of the class diagram easier to read. It also made it clear how information will be pulled from the database. The Proxy pattern also makes it easier to change or add another database in the case where another type of database needs to be used instead.

We found that the Factory pattern improved our initial design because the User and Message classes have multiple different child classes that need to be instantiated, but had slightly different implementations of their child methods. Although this added more classes to the diagram, it removed ambiguity in how the child classes would be instantiated. There were classes for which we wanted only one instantiation, for those we used Singleton.

## Design Pattern Implementation

Proxy and Singleton Pattern

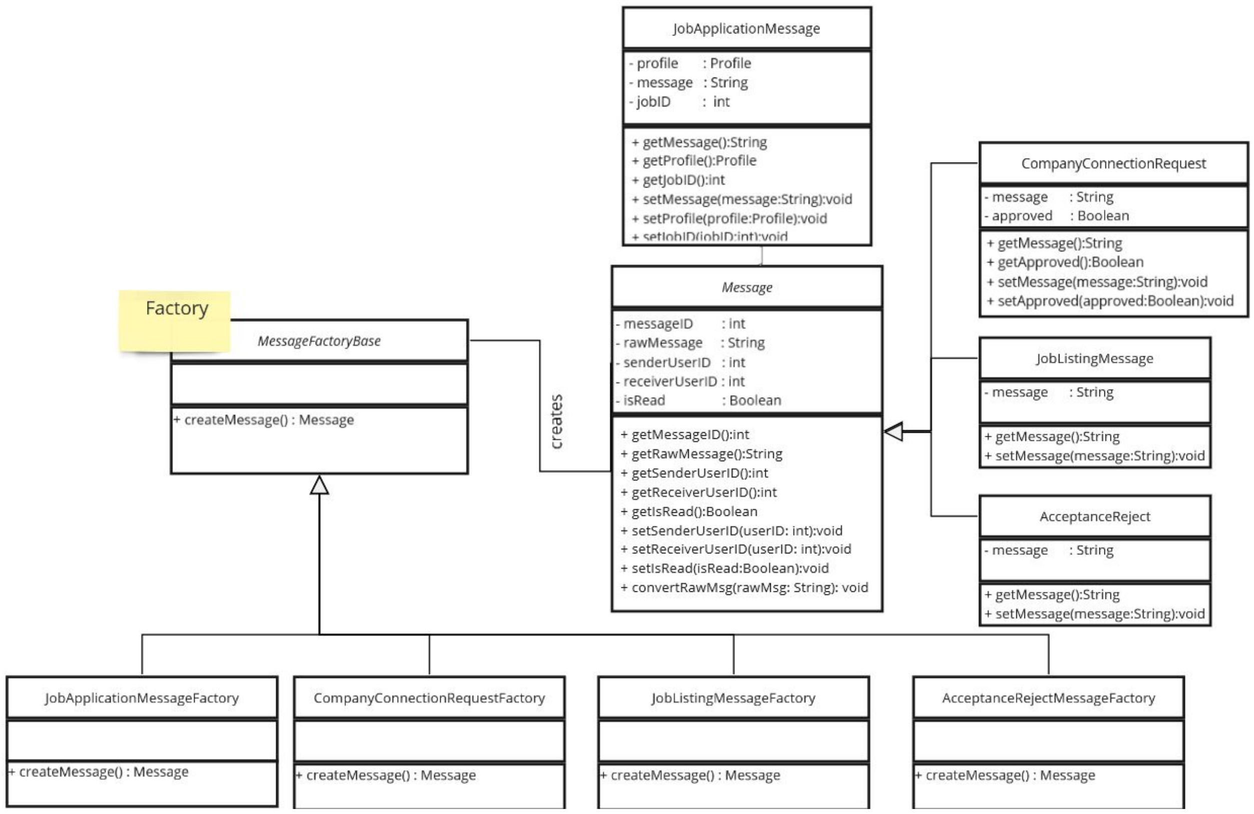
For the implementation of the database we used a combination of Proxy and Singleton Pattern. The Database interface was created enabling common methods such as readPostor writePost for any instance of a database and the DatabaseProxy. A specific instance of a type of database (in our case MySQLDatabase) implemented the Database interface and uses the Singleton pattern ensuring only one instance of this type of database. MySQLDatabase is written to do lazy instantiation so it only creates the instance when getInstance is called. The ProxyDatabase uses the Proxy pattern to refer to the MySQL database singleton instance, decoupling the MySQL database from the client code calling the database methods.


MySQLDatabase
Singleton
- instance: MySQLDatabase
- user: String
- password:String
- connect(): void
- execute(query:String): ResultSet
- disconnect(): void
+ getInstance(): MySQLDatabase
+ readUser(username:String) :User
+ writeUser(user:User): bool
+ deleteUser(username:String):bool
+ updateUser(username:String, user:User):bool
+ readPost(postingID:int):Posting
+ writePost(posting:Posting):bool
+ deletePost(postingID:int):bool
+ updatePost(postingID:int, posting:Posting):bool
+ readMessage(messageID:int):Message
+ writeMessage(message:Message):bool
+ deleteMessage(messageID:int):bool
+ searchUser(query:String, userType:String, filter:UserFilter):ArrayList<User>
+ searchPosting(query:String, filter:PostingFilter):ArrayList<Posting>
delegate
DatabaseProxy
Proxy
- database: MySQLDatabase
+ readUser(username:String) :User
+ writeUser(user:User): bool
+ deleteUser(username:String):bool
+ updateUser(username:String, user:User):bool
+ readPost(postingID:int):Posting
+ writePost(posting:Posting):bool
+ deletePost(postingID:int):bool
+ updatePost(postingID:int, posting:Posting):bool
+ readMessage(messageID:int):Message
+ writeMessage(message:Message):bool
+ deleteMessage(messageID:int):bool
+ searchUser(query:String, userType:String, filter:UserFilter):ArrayList<User>
+ searchPosting(query:String, filter:PostingFilter):ArrayList<Posting>
<<interface>>
Database
+ readUser(username:String) :User
+ writeUser(user:User): bool
+ deleteUser(username:String):bool
+ updateUser(username:String, user:User):bool
+ readPost(postingID:int):Posting
+ writePost(posting:Posting):bool
+ deletePost(postingID:int):bool
+ updatePost(postingID:int, posting:Posting):bool
+ readMessage(messageID:int):Message
+ writeMessage(message:Message):bool
+ deleteMessage(messageID:int):bool
+ searchUser(query:String, userType:String, filter:UserFilter):ArrayList<User>
+ searchPosting(query:String, filter:PostingFilter):ArrayList<Posting>

Classes implementing Proxy and Singleton Design Pattern

Factory Pattern

Although we did not implement all features for the Prototype, we still implemented the Factory pattern for creating the JobListingMessage and JobApplicationMessage using our JobApplicationMessageFactory and JobListingMessageFactory that inherit from abstract MessageFactoryBase. Each of these message factories enabled one place for our client code to create a specific type of message.


JobApplicationMessage
- profile : Profile
- message : String
- jobID : int
+ getMessage():String
+ getProfile():Profile
+ getJobID():int
+ setMessage(message:String):void
+ setProfile(profile:Profile):void
+ setJobID(jobID:int):void
CompanyConnectionRequest
- message : String
- approved : Boolean
+ getMessage():String
+ getApproved():Boolean
+ setMessage(message:String):void
+ setApproved(approved:Boolean):void
Factory
MessageFactoryBase
+ createMessage() : Message
creates
Message
- messageID : int
- rawMessage : String
- senderUserID : int
- receiverUserID : int
- isRead : Boolean
+ getMessageID():int
+ getRawMessage():String
+ getSenderUserID():int
+ getReceiverUserID():int
+ getIsRead():Boolean
+ setSenderUserID(userID: int):void
+ setReceiverUserID(userID: int):void
+ setIsRead(isRead:Boolean):void
+ convertRawMsg(rawMsg: String): void
JobListingMessage
- message : String
+ getMessage():String
+ setMessage(message:String):void
AcceptanceReject
- message : String
+ getMessage():String
+ setMessage(message:String):void
JobApplicationMessageFactory
+ createMessage() : Message
CompanyConnectionRequestFactory
+ createMessage() : Message
JobListingMessageFactory
+ createMessage() : Message
AcceptanceRejectMessageFactory
+ createMessage() : Message

Classes implementing Factory Design Pattern

## Learning Outcomes: Creation, Design, Implementation

Our team learned that design decisions early on in the project can have a large influence how the project proceeds. Almost all of the early discussions involving the users and how we wanted them to interact with the system changed how we worked on the diagrams further down the road. It was also important for our process of design to model parts of the diagrams and then update them iteratively. Doing this instead of trying to get everything done the first time it hits the paper helped us understand what each other's ideas were and helped start discussions about particular design decisions.

The design patterns we decided to use improved the class diagram greatly because they offered solutions to some of the issues we were having with ambiguity in how parts of the system should be implemented. Having many discussions and doing much upfront planning reduced the amount of work we needed to do later on when implementing the system. The total time it took to implement our demo was about 5 hours. If we had to have many of the design discussions interrupting the implementation, the time required to implement the demo would have been much longer.