CS102	**Summer 2023**		1E
Instructor:	**Özcan Öztürk**	Project Group	
Assistant:	**Mert Kara**		

Criteria	TA/Grader	Instructor
Presentation		
Overall		

~ BilkenTogether ~

(A Desktop Application)

SocialFive

Ahmet Furkan KIZIL

Erdem Pülat

Gülferiz Bayar

Ufuk Baran Güler

Berfin Örtülü

Detailed Design Stage Report

6 July 2023

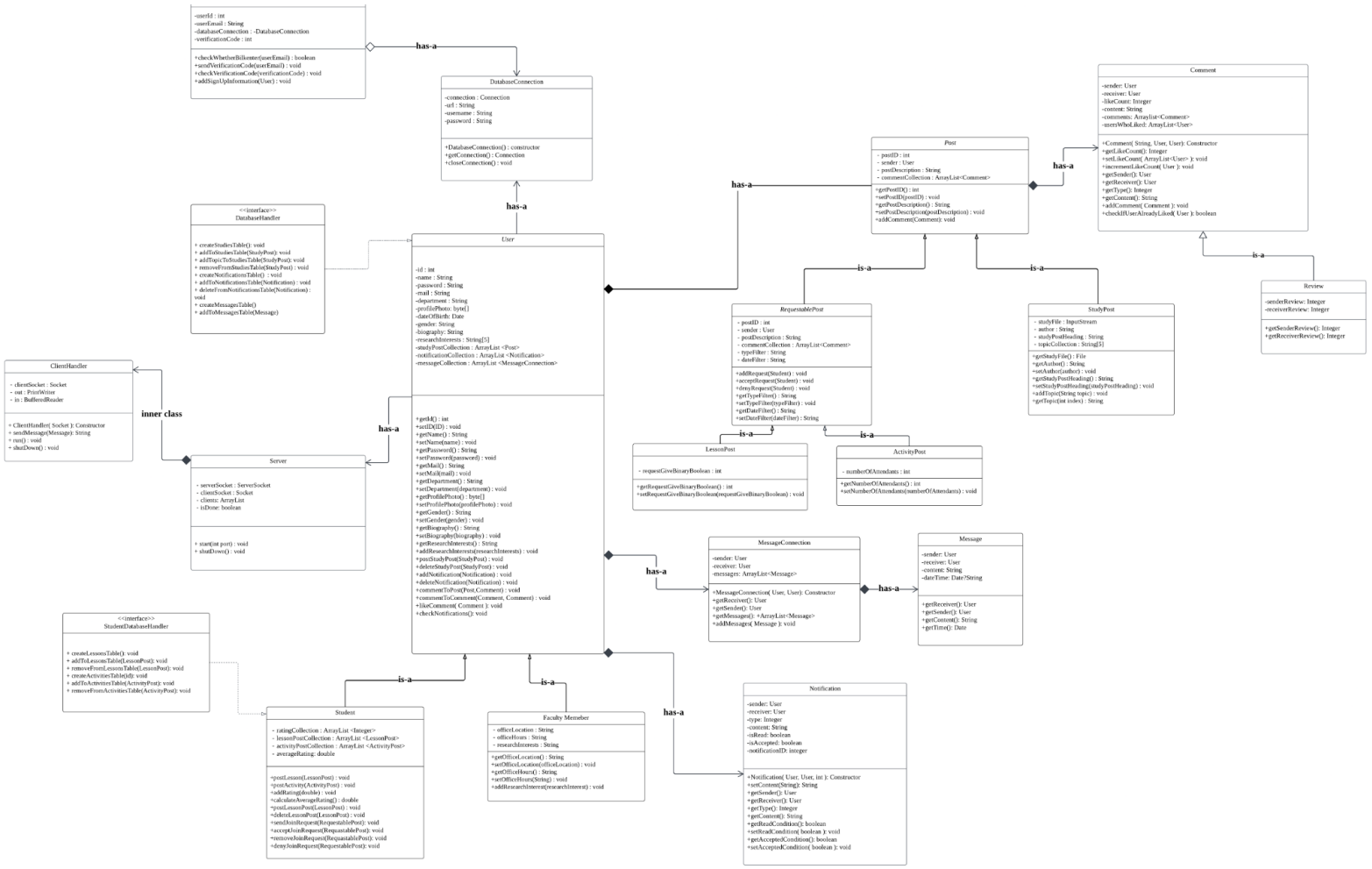
1. Introduction

BilkenTogether is a desktop application aimed to foster collaboration, discussion, and communication in academia. With an all-in-one mindset, BilkenTogether intends to create a fulfilling experience.

2. System Overview

In this project, various java libraries and the MySQL database technologies will be employed. Used libraries include java.swing, java.fx, java.io, java.net. Swing and fx will be used for creating the interface. IntelliJ IDEA and its form files will also be used to create the interface. io, net, sql libraries along with the MySQL database will form the back-end of the application.

3. Core Design Details



(The pdf version is available and can be sent through e-mail for better vision.)

User Class (Abstract) *(implements **DatabaseHandler** interface)*

Properties (Instance Variables):

- id : int
- name : String
- password : String
- mail : String
- department : String
- profilePhoto: byte[]
- dateOfBirth: Date
- gender: String
- biography: String
- researchInterests : String[5]
- studyPostCollection : ArrayList <Post>
- notificationCollection : ArrayList <Notification>
- messageCollection : ArrayList <MessageConnection>

Methods:

+ getId() : int
+ setID(ID) : void
+ getName() : String
+ setName(name) : void
+ getPassword() : String
+ setPassword(password) : void
+ getMail() : String
+ setMail(mail) : void
+ getDepartment() : String
+ setDepartment(department) : void
+ getProfilePhoto() : byte[]
+ setProfilePhoto(profilePhoto) : void
+ getGender() : String
+ setGender(gender) : void
+ getBiography() : String
+ setBiography(biography) : void
+ getResearchInterests() : String
+ addResearchInterests(researchInterests) : void
+ postStudyPost(StudyPost) : void

  (This method will be used in both Student and FacultyMember classes. The method receives a StudyPost object and adds it to the studyPostCollection.)

+ deleteStudyPost(StudyPost) : void

  (This method will first check if the passed StudyPost exists in the studyPostCollection and if it exists, it will delete the respective notification from the studyPostCollection)

+ addNotification(Notification) : void

  (The method receives a Notification object and adds it to the notificationCollection and the Notification Table of the SQL Database)

+ deleteNotification(Notification) : void

(This method will first check if the passed Notification exists and if it exists, it will delete the respective notification from the NotificationCollection and the Notifications Table of the SQL Database)

+ commentToPost(Post,Comment) : void

(This method will receive a Comment and adds it to the respective Post's commentCollection array and Comment Table in SQL Database)

+ commentToComment(Comment, Comment) : void
+ likeComment(Comment): void
+ checkNotifications(): void (listens to notifications constantly until the app is closed)

DatabaseConnection Class

Properties (Instance Variables):

- connection : Connection
- url : String
- username : String
- password : String

Methods:

+ DatabaseConnection() : constructor

```
public DatabaseConnection() {
    // Database credentials
    try {
        connection = DriverManager.getConnection(url, username, password);
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
}
```

+ getConnection() : Connection
+ closeConnection() : void

SignUp Class

Properties (Instance Variables):

- userId : int
- userEmail : String
- databaseConnection : DatabaseConnection
- verificationCode : int (Created randomly)

Methods:

+ checkWhetherBilkenter(userEmail) : boolean

(This method would check whether the entered Email is a Bilkent Email or not)

+ sendVerificationCode(userEmail) : void

(This method is executed if the return type of the checkWhetherBilkenter() method is true and sends a verification code to the Email address of the user.)

+ checkVerificationCode(verificationCode) : void

(This method checks whether the verification code is matching or not.)

+ addSignUpInformation(User) : void

(This method adds the respective parameters to the userInformation Table of the MySQL Database)

<<DatabaseHandler Interface>>

Methods:

+ createStudiesTable(): void

(Since this method will be used in both Students and FacultyMember class, we intend to implement it in User class. This method creates a Studies Table in MySQL with a name as "Studies of" + id (getId() method) and this table will be used for storing Studies objects in the Studies Table. The table will have the following rows : PostID, Topic1, … , Topic 5 and File.)

+ addToStudiesTable(StudyPost): void

(This method will receive a StudiesPost object and by using its getPostID(), fillInTopics(), getFile() methods; fill in these data to the respective rows which were created in createStudiesTable().)

+ addTopicToStudiesTable(StudyPost): void

(This method will access the individual topics of the passed StudyPost object StudyPost.getTopic(int index) and will add the respective topics to the Topic rows of the Studies Table)

+ removeFromStudiesTable(StudyPost) : void
+ createNotificationsTable() : void

(This method creates a Notification Table in MySQL with a name as "Notification of " + id and this table will be used for storing Notification instances in Notification Table)

+ addToNotificationsTable(Notification) : void
+ deleteFromNotificationsTable(Notification) : void
+ createMessagesTable()
+ addToMessagesTable(Message)

<<StudentDatabaseHandler Interface>>

Methods:

+ createLessonsTable(): void

(This method creates a Lessons Table in MySQL with a name as "Lessons of" + id (getId() method) and this table will be used for storing Lessons objects in the Lessons Table. The table will have the following rows : PostID, typeFilter, dateFilter, requestGiveBoolean)

+ addToLessonsTable(LessonPost): void

(This method will receive a LessonPost object and by using its getPostID(), getTypeFilter(), getDateFilter(), getRequestGiveBinaryBoolean() methods; fill in these data to the respective rows which were created in createLessonsTable().)

+ removeFromLessonsTable(LessonPost): void

+ createActivitiesTable(id): void

  (This method creates an Activity Table in MySQL with a name as "Activities of" + id (getId() method) and this table will be used for storing Activity objects in the Activity Table. The table will have the following rows : PostID, typeFilter, dateFilter, requestGiveBoolean)

+ addToActivitiesTable(ActivityPost): void

(This method will receive a ActivityPost object and by using its getPostID(), getTypeFilter(), getDateFilter(), getRequestGiveBinaryBoolean() methods; fill in these data to the respective rows which were created in createActivitiesTable().)

+ removeFromActivitiesTable(ActivityPost): void

Student Class *(extends **User** implements **StudentDatabaseHandler** interface)*

Properties (Instance Variables):

- ratingCollection : ArrayList <Integer>
- lessonPostCollection : ArrayList <LessonPost>
- activityPostCollection : ArrayList <ActivityPost>
- averageRating: double

Methods:

+ postLesson(LessonPost) : void
+ postActivity(ActivityPost) : void
+ addRating(double) : void
+ calculateAverageRating() : double
+ postLessonPost(LessonPost) : void

  (This method will be used only for Student classes. The method receives a LessonPost object and adds it to the lessonPostCollection.)

+ deleteLessonPost(LessonPost) : void

  (This method will first check if the passed LessonPost exists in the lessonPostCollection and if it exists, it will delete the respective notification from the lessonPostCollection)

+ sendJoinRequest(RequestablePost): void
+ acceptJoinRequest(RequastablePost): void
+ removeJoinRequest(RequastablePost): void
+ denyJoinRequest(RequestablePost): void

FacultyMember Class

Properties (Instance Variables):

- officeLocation : String
- officeHours : String
- researchInterests : String

Methods:

+ getOfficeLocation() : String
+ setOfficeLocation(officeLocation) : void

+ getOfficeHours() : String
+ setOfficeHours(String) : void
+ addResearchInterest(researchInterest) : void

Post Class (Abstract)

Properties (Instance Variables):

- postID : int
- sender : User
- postDescription : String
- commentCollection : ArrayList<Comment>
-

Methods:

+ getPostID() : int
+ setPostID(postID) : void
+ getPostDescription() : String
+ setPostDescription(postDescription) : void
+ addComment(Comment): void

RequestablePost Class (Abstract) *(extends **Post**)*

Properties (Instance Variables):

- typeFilter : String
- dayFilter : String
- requestCollection : ArrayList <Student>
- deniedCollection : ArrayList <Student>
- agreementCollection : ArrayList <Student>

  (The first element of the agreementCollection will be the sender instance variable (User) which will be done in the constructor)

Methods:

+ getTypeFilter() : String
+ setTypeFilter(typeFilter) : void
+ getDateFilter() : String
+ setDateFilter(dateFilter) : String
+ addRequest(Student) : void

  (This method receives a student object and adds it to the requestCollection. From this collection, the provider of the lesson will be able to see the Students which requested to the specific lesson)

+ acceptRequest(Student) : void

  (This method will first check whether the request is in the requestCollection or not. If it's there, then the passed Student will be added to the agreementCollection.)

+ denyRequest(Student) : void

  (This method will first check whether the request is in the requestCollection or not. If it's there, then the passed Student will be added to the deniedCollection.)

LessonPost Class *(extends **RequestablePost**)*

Properties (Instance Variables):

- requestGiveBinaryBoolean : int

Methods:

+ getRequestGiveBinaryBoolean() : int
+ setRequestGiveBinaryBoolean(requestGiveBinaryBoolean) : void

StudyPost Class *(extends **Post**)*

(java.io.InputStream will be imported)

Properties (Instance Variables):

- studyFile : InputStream
- author : String
- studyPostHeading : String
- topicCollection : String[5] (adding topics is limited to 5)

Methods:

+ getStudyFile() : File
+ getAuthor() : String
+ setAuthor(author) : void
+ getStudyPostHeading() : String
+ setStudyPostHeading(studyPostHeading) = void
+ addTopic(String topic) : void

  (This method first checks if there is an empty (null) space in topicCollection and if there is a space, adds the String (topic) to the topicCollection.

+ getTopic(int index) : String

  (This method will check if the index is less than or equal to 5 first. Then it returns the specified String (topic) from the topicCollection Array with the use of the passed integer index.)

ActivityPost Class *(extends **RequestablePost**)*

Properties (Instance Variables):

- numberOfAttendants : int

Methods:

+ getNumberOfAttendants() : int
+ setNumberOfAttendants(numberOfAttendants) : void

Server Class

(java.io will be imported)

(java.net will be imported)

Properties (Instance Variables):

- serverSocket : ServerSocket
- clientSocket : Socket
- clients: ArrayList
- isDone: boolean

Methods:

+ start(int port) : void (Starts the Server on a port.)
+ shutDown() : void (Closes the server, releases all the resources.)

Inner Class: ClientHandler

By having an inner class, multiple clients are allowed. 2 for this chat.

ClientHandler(Socket client)

Properties (Instance Variables):

- serverSocket : ServerSocket
- clientSocket : Socket
- out : PrintWriter
- in : BufferedReader

Methods:

+ ClientHandler(Socket): Constructor
+ sendMessage(Message): String (Broadcasts the message)
+ run() : void (Connects the client to the server, initializes the input/output streams)
+ shutDown() : void

Message Class

Properties (Instance Variables):

- sender: User
- receiver: User
- content: String
- dateTime: Date

Methods:

+ Message(User, User, String, dateTime): Constructor
+ getReceiver(): User
+ getSender(): User
+ getContent(): String
+ getTime(): Date

MessageConnection Class

Properties (Instance Variables):

- sender: User
- receiver: User
- messages: ArrayList<Message>

Methods:

+ MessageConnection(User, User): Constructor
+ getReceiver(): User
+ getSender(): User
+ getMessages(): ArrayList<Message>
+ addMessages(Message): void

Notification Class

Properties (Instance Variables):

- sender: User
- receiver: User
- type: Integer (there are 3 possible notifications, denoting to 0,1,2)
- content: String
- isRead: boolean
- isAccepted: boolean
- notificationID: integer

Methods:

+ Notification(User, User, int): Constructor
+ setContent(String): String (sets the content according to the notification type, and users)
+ getSender(): User
+ getReceiver(): User
+ getType(): Integer
+ getContent(): String
+ getReadCondition(): boolean
+ setReadCondition(boolean): void
+ getAcceptedCondition(): boolean
+ setAcceptedCondition(boolean): void

Comment Class

Properties (Instance Variables):

- sender: User
- receiver: User
- likeCount: Integer
- content: String
- comments: Arraylist<Comment>
- usersWhoLiked: ArrayList<User>

Methods:

+ Comment(String, User, User): Constructor
+ getLikeCount(): Integer
+ setLikeCount(ArrayList<User>): void
+ incrementLikeCount(User): void
+ getSender(): User
+ getReceiver(): User
+ getType(): Integer
+ getContent(): String

+ addComment(Comment): void
+ checkIfUserAlreadyLiked(User): boolean

Review Class (*Extends* ***Comment***)

Properties (Instance Variables):

- senderReview: Integer
- receiverReview: Integer

Methods:

+ getSenderReview(): Integer
+ getReceiverReview(): Integer

4. Task assignment

Ahmet Furkan KIZIL: DatabaseConnection, SignUp, DatabaseHandler, StudentDatabaseHandler

Erdem Pülat: Post, StudyPost, RequestablePost, ActivityPost, LessonPost, Server, ClientHandler

Gülferiz Bayar: Comment, Review

Ufuk Baran Güler: User, Student, FacultyMember

Berfin Örtülü: Notification, Message, MessageConnection

5. Summary & Conclusions

This report will assist the making of the project extensively. The method serves to divide the functionality of the algorithm to different classes and methods, and to divide the parts of the work among the group members.

6. References

Baeldung, W. by: (2023, June 14). *A guide to java sockets*. Baeldung. https://www.baeldung.com/a-guide-to-java-sockets Accessed 24 July 2023.