CS 319
OBJECT ORIENTED SOFTWARE ENGINEERING
DESIGN REPORT
PEER-REVIEW

GROUP 1D

MEMBERS:

Eylül Çağlar (Group Leader) 21703949

İsmet Alp Eren 21703786

Kaan Ateşel 21703694

Kerem Alemdar 21702133

Muharrem Berk Yıldız 21802492

Instructor: Eray Tüzün

Teaching Assistant(s): Elgun Jabrayilzade, Erdem Tuna

1.Introduction
    1.1 Purpose of the system
    1.2 Design goals
    1.3 Definitions, acronyms, and abbreviations
    1.4 References
    1.5 Overview
2. Subsystem Decomposition
    2.1 The Application View Subsystem
    2.2 the Application Controller Subsystem
    2.3 The Http Request Handler Subsystem
    2.4 the Database Object Manager Subsystem
3. Hardware / Software Mapping:
4. Access Control:
5. Persistent Data Management:
    5.1 User information
    5.2 Group information
    5.3 Global information
6. Subsystem Services:
    6.1 The Application View Subsystem
    6.2 the Application Controller Subsystem
    6.3 The Http Request Handler Subsystem
    6.4 the Database Object Manager Subsystem
7. Glossary & references

1. Introduction

1.1 Purpose Of The System

Pire that will be designed is a digital peer review application which aims to help Bilkent University students and instructors for their Computer Science courses and projects. Differently from other peer review applications “Pire” offers many additional features that let users to easily access groups, review group tasks, grade other groups projects and at the end of the course see the grades of their projects given by instructors or TAs.

1.2 Design Goals

Following subsections show the design goals that are being considered in the making of the Pire application in the light of nonfunctional requirements. In the Application Controller subsystem we used a **strategy design** pattern. In order to handle updates and new features. In our The Http Request Handler and the Database Object Manager subsystems we used **Singleton Design Pattern.** Therefore the application will not have many same objects in runtime so the application will use more memory.

1.2.1 Nonfunctional Design Goals

Response Time

This Pire (Peer Review) application that was designed for Bilkent University Computer Science courses is a database dependent application. When database factor is considered, response time is important to maintain application efficiently.

Therefore, the system is designed to have a maximum response time of less than 2 seconds. This ensures to keep the users' attention in the application.

Extensibility

Pire application will be implemented by considering the real life cases. It means that application programmers take into account that users may want new features in the application. Thus, Pire application implementation will give a chance to add new features easily by just adding new things such as subclasses or new methods. This type of implementation will give extensibility quality to the implementation.

Reliability

The application is designed to behave in an expected logical way, data loss and delay is not much possible. The unintended system behaviors are minimized by making the system 99% reliable since a single unintended behavior might disrupt the process of the application and flow of the project.

2. Subsystem Decomposition

Our system is divided into four subsystems. The Application View subsystem handles user interfaces. Application Controller subsystem sends and receives from Http Request Handler Subsystem. It sends a request to the server. The Http Request Handler Subsystem answers the requests from the application from the server side. The Http Request Handler calls the Database Object Manager Subsystem which is responsible from database operations. Writing data to the database and retrieving

data from the database.

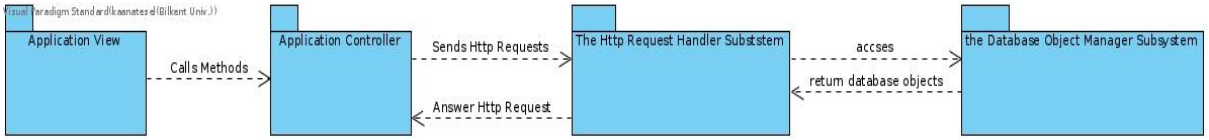

Figure 1: Subsystem Decomposition

2.1 The Application View Subsystem

The application view subsystem is frontend of the web application. Thus it visualises the system to the user. Users can see the object such as groups, polls. This subsystem also has forms and buttons. Therefore the user can interact with the Pire application.

2.2 The Application Controller Subsystem

The Application controller subsystem handles the button events and form submits. It sends http requests to the backend server. It also receives data from backend servers and shows them to the user using Application View subsystem functions.

2.3 The Http Request Handler Subsystem

This subsystem handles the http requests which are sent from the Application controller subsystem. This subsystem works with the database object manager side by side. It sends the request to the database object manager subsystem and it returns the database objects. Then The Http Request Handler sends this data to the application controller.

2.4 the Database Object Manager Subsystem

This subsystem is responsible for database operations. This subsystem adds, deletes, updates or retrieves datas from the database. After it finishes the database operation it returns the result to The Http Request Handler subsystem.

3. Hardware / Software Mapping

Pire is a web application so it has 3 different parts. First one is a react application which can be run in any web server. This class listens and sends http requests to the backend server. Backend server is a JAVA spring boot application which can be runned in any server which has JDK 11. The last part is database. The Pire application uses PostgreSQL-11 as a database. Therefore It can run any server which has PostgreSQL-11 on it.

4. Access Control

The system has to access the database to get and save data and information. The users have to access the database to login the system and have to keep accessing it while using the application. Program keeps the information such as id, e-mail if needed and project details. Since documents and personal information are private and should be kept safe, database security is an important issue for the system.

The system will hashes and encryption all necessary and private data and information to keep users safe and unreachable by both users and non-users.

- Evaluation period can be activated only by the instructor and evaluation forms can be created only by the instructor.
- In Pire application only the instructor and the TA can reach the project details until the instructor activates evaluation period.
- Student ID’s are visible only by the instructor.
- Sending invitations and accepting/rejecting requests are the available functions only for Leader Students.

- Filling incomplete groups is the function that can only be done by the instructor or the TA.
- Students can have invitations but TA's and instructors cannot have any invitations.
- By export group grades, grades are visible by the TA and the instructor but not by students.
- Generating course code can only be done by the instructor.

5. Persistent Data Management

The system needs to store data about:

- User information
- Group information
- Global information

5.1 User information

User information will stored in order to provide following functionalities:

- Authenticate user when tried to login
- Show the name of the user when send a message
- Show the name of the user when post a review.
- Show the user's personal information like email, surname, second name.
- Display user's friends and incoming friend requests.
- Sending and receiving group invitations.
- Check user type and permissions
- Send and show personal messages among users

Therefore the system will be storing following datas:

- User name (firstname, second name)
- User surname
- User email address
- User password
- User ID
- User’s friend list.
- Personal Messages.

5.2 Group information

Group information will stored in order to provide following functionalities:

- Display pdf documents of the group members.
- Show the group members.
- Allocate and display group tasks, deadline and assigned person.
- Check prerequisites of the group for new members.
- Display group polls and its results.
- Display Group schedules assigned by the group leader.
- Provide document information like file url, contributors, artifact type and score.
- Show the reviews of other students.
- Display reviews and grades of instructors and TAs.

Therefore the system will be storing following datas:

- Group name.
- Member IDs and Group Leader.
- Group Tasks
- Group Task Templates
- Prerequisites, its type, condition, and strictness.

- Document url, grade and reviews.

5.3 Global information

Global information will stored in order to provide following functionalities:

- Show global chat where messages and their names are shown.
- Show global polls started by the instructor or TA including its template and answers.
- Show schedules date and their message.

Therefore the system will be storing following datas:

- Chat messages.
- Global polls templates and result values
- Schedule date, content.

6. Subsystem Services

6.1 The Application View Subsystem

The pire is a web application that doesn't have a class diagram for user interface.

6.2 The Application View Subsystem

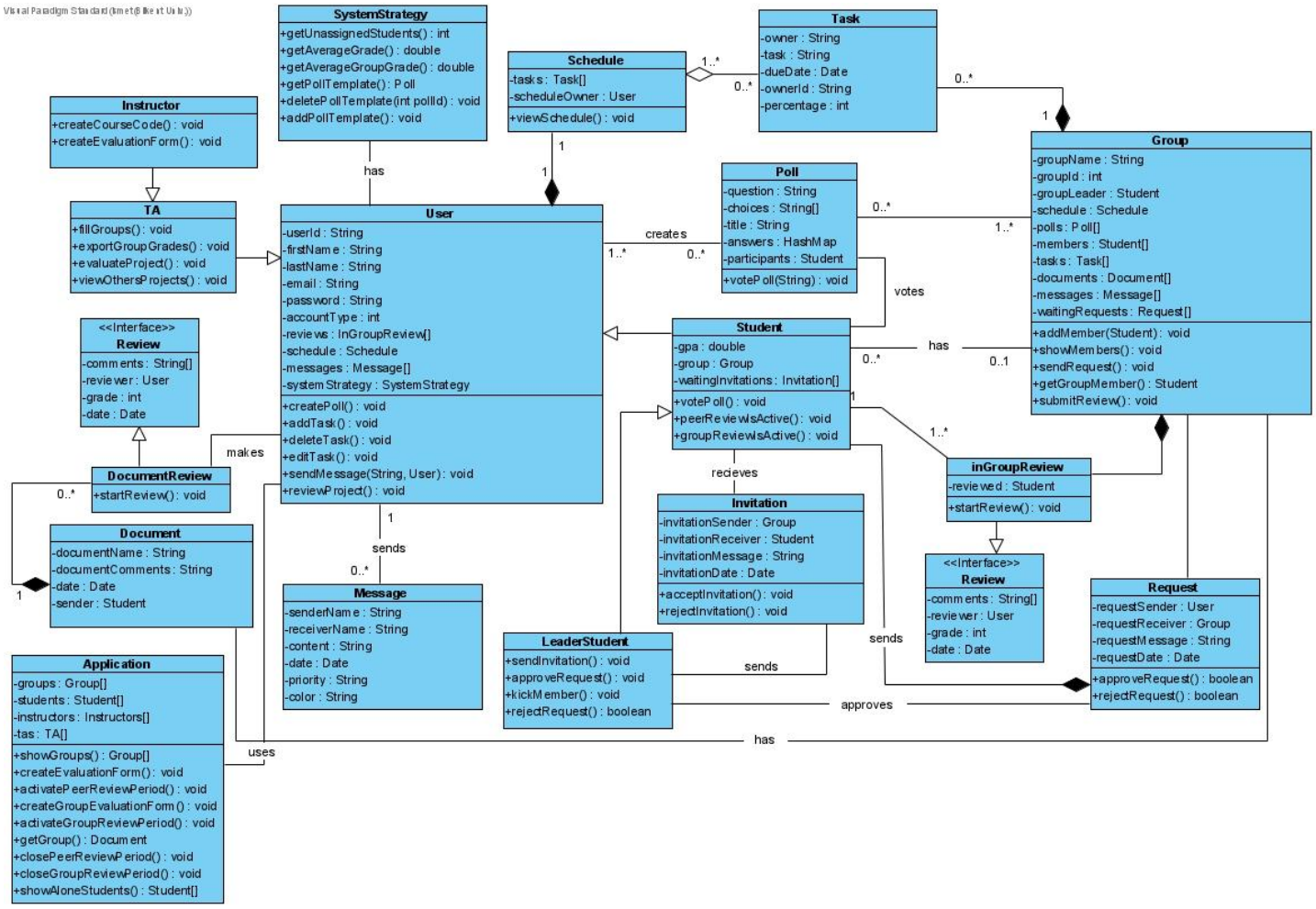

Figure 2: Class Diagram

Group Class:

Explanation: This class is a representation of a group.

Attributes:

- **private Student[] members:** This attribute contains the members of the group as an Account array.
- **private String groupName:** This attribute contains the name of the group.
- **private int groupId:** This attribute contains a unique id of the group.
- **private Student groupLeader:** This attribute contains the leader student of the group.
- **private Document[] documents:** This attribute contains sent documents to the group by group members as a Document array.
- **private Message[] messages:** This attribute contains messages sent to groups by group members as a Message array.
- **private Poll[] polls:** This attribute contains the polls created by that group as a Poll array.
- **private Task[] tasks:** This attribute contains tasks for that group as a Task array.
- **private Request[] waitingRequests:** This attribute contains the join request sent to that group as a Request array.
- **private Schedule schedule:** This attribute contains schulade of the group as a Schedule object.

Methods:

- **public void addMember(Student):** This method lets students add members to the group.
- **public void showMembers():** This method lets users to show the members of the group.
- **public void sendRequest():** This method lets students send a join request to a selected group.
- **public Student getGroupMember():** This method lets users get the specific member of the group.
- **public void submitReview():** This method lets users submit a review.

Review Interface:

Explanation: This interface includes all reviews to the finished project. Both students,TA's and instructors could give reviews to other projects and give grades. In addition to giving grades students could give a suggestion and TA's & Instructors may include feedback in their review.

Attributes:

- **private String[] comments:** This array includes comments and suggestions/feedback related to the project.
- **private User reviewer:** It stores who the reviewer is.
- **private int grade:** Integer array of grades, each index represents different categories of grades. Ex: Reliability, Security, Overall, etc..
- **private Date date:** It contains review date as string.

inGroupReview Class:

Explanation: This class includes all in group review that is given by project members to their own teammates at the end of the project.

Attributes:

- **private Student reviewed:** This attribute stores who has been evaluated.

Methods:

- **public void startReview():** This method lets students start the reviewing process.

Poll Class:

Explanation: Poll class is for enabling users to create polls where they can create their own poll questions.

Attributes:

- **private String question:** Question is a string for where users enter their poll questions.
- **private String[] choices:** This is a private string array for users to indicate their choices for their poll questions.
- **private String title:** Title is a string for users to indicate their polls subject, thus other users of the system can understand the aim of that poll.
- **private HashMap answers:** Result is an integer array for storing the number of votes for every individual choice.

- **private Student[] participants:** Students array that keeps which students voted the poll.

Methods:

- **public votePoll():** This method submits the poll.

Task Class

Explanation: Task class is for arranging schedules.

Attributes:

- **private String owner:** Owner is a string for indicating who will do the task.
- **private Date dueDate:** Due date string shows the date which is the last date of the task to be completed.
- **private String task:** Task string is for showing what should be done before the deadline.
- **private String ownerId:** Owner id is a string which is specific for every individual user. In Task class, the owner id shows the id of the user who should complete the task.
- **private int percentage:** Percentage is an integer in range 0-100. It shows how much the task has been completed by the owner of the task.

Request Class

Explanation: Invitation request from group to student in order to attract their attention.

Attributes:

- **private User requestSender:** This attribute contains which group sends the request.
- **private Group requestReceiver:** This attribute contains who will receive the invitation request.
- **private String requestmessage:** This attribute contains messages related to the request.
- **private Date requestDate:** This attribute contains request creation date.

Methods:

- **public boolean approveRequest():** This method accepts the request.
- **public boolean rejectRequest():** This method lets a student to reject the join request.

Message Class

Explanation: This class includes all attributes related to messages.

Attributes:

- **private String senderName:** This attribute contains who sent the message.
- **private String receiverName:** This attribute contains who will receive the message, this could also contain group name.

- **private String content:** This attribute contains messages to send.
- **private Date date:** This attribute contains the date which the message was sent.
- **private String priority:** This attribute contains priority of the message. Depends on the priority, even if the user closed notifications off, they will still receive notification.
- **private String color:** The message sent may have different colors depending on priority of the message, or how often users go back to a specified message.

Document Class:

Explanation: Document class is for documents which are uploaded to the system by users.

Attributes:

- **private documentName:** Document name is a string for showing the name of the uploaded document.
- **private documentComments:** Document comments string includes the users', who uploaded the document, explanation for the document.
- **private date:** Date shows the upload date of the document.
- **private sender:** Sender is a Student which shows the user who uploaded the particular document.

DocumentReview Class:

Explanation: Document class is for documents' reviews.

Method:

- **public void startReview() :** This method lets TA or Instructor to start the reviewing process.

Schedule Class:

Explanation: Both users and groups have schedules which contain which tasks to be done and when to do them.

Attributes:

- **private tasks:** This attribute contains all tasks related to that schedule in the task array.
- **private scheduleOwner:** This attribute keeps the User class as an owner of the schedule.

Methods:

- **public viewSchedule():** This method let User to view the personal Schedule class.

User Class:

Explanation: This class represents the functionalities of the account.

Attributes:

- **private firstname:** The attribute that contains the user's first name.
- **private lastname**: The attribute that contains the user's last name.
- **private email:** The attribute that contains the user's email address.

- **private password:** The attribute that contains the user's hashed password.
- **private accountType:** The attribute that contains the user's type student or instructor or TA.
- **private userId:** The attribute that contains the user's ID.
- **private reviews:** This is an array of user reviews which users have given to other groups' projects.
- **private messages:** This is an array of Message objects. Users private or group chat messages keep in this list.
- **private schedule:** This attribute keeps the Schedule class.
- **private systemStrategy:** This attribute keeps the SystemStrategy class.

Methods:

- **public createPoll():** This method calls the Poll class to create a new Poll.
- **public addTask():** This method calls the Task class to create a new Task.
- **public reviewProject():** This method let User to review other groups' final project documents.
- **public sendMessage(String, User):** This method let User to send a message to other Users.
- **public addTask():** This method let User to add a new task.
- **public deleteTask():** This method let User to delete the created task.
- **public editTask():** This method let User to edit the created task.

Instructor Class:

Explanation: This class is extended from account class. Then it is specialized in order to meet the instructor users.

- **public void createCourseCode() :** This method creates course code for new students to use while they are registering.
- **public void createEvaluationForm() :** This method creates an evaluation form for students to use in peer review and group reviews.

TA Class:

Explanation: This class is extended from account class. Then it is specialized in order to meet the TA users

Methods

- **public void fillGroups() :** This method fills all groups that are not yet full.
- **public void exportGroupGrades() :** This method allows TA and Instructor to export all grades from each group in a single document.
- **public void evaluateProject() :** This method allows TA and Instructor to give final grade to the project.
- **public void viewProject() :** This method shows a list of all project groups.

Student Class:

Explanation: This class is extended from account class. Then it is specialized in order to meet the student users.

Attributes:

- **private double gpa :** The attribute that contains gpa of the student.
- **private Group group :** The attribute that contains the group of the student.
- **private Invitation[] waitingInvitations :** The attribute that contains invitations sent by groups.

Methods:

- **public void votePoll() :** This method allows the user to vote on a poll.
- **public void peerReviewIsActive():** This method lets a student check if peer review period is activated or not.
- **public void groupReviewIsActive():** This method lets a student check if group review period is activated or not.

LeaderStudent Class:

Explanation: This class is extended from student class but also has the ability to manage the group.

Methods:

- **public void sendInvitation():** Send invitation objects to other students which are not in the group.
- **public boolean approveRequest():** Approve group join requests that are sent by students.

- **public void kickMember(): This method is to kick members from the group.**
- **public boolean rejectRequest(): This method is for rejecting join requests that are sent by students.**

Invitation Class:

Explanation: This class is responsible for notifying students that they are invited to a group.

Attributes:

- **private Group invitationSender:** This is the LeaderStudent object who sended the invitation.
- **private Student invitationReceiver:** This is the student who will receive the invitation.
- **private Date invitationDate:** This attribute represents the delivery time of invitation.
- **private String invitationMessage:** The invitation message kept in this attribute.

Methods:

- **acceptInvitation() :**This method lets Student to accept an invitation.
- **rejectInvitation() :**This method lets Student to reject an invitation.

Application Class:

Explanation: This class is responsible for activating and disabling functions in our application.

Methods:

- **private Group[] showGroups() :** This method returns all groups in the system.
- **private void createEvaluationForm() :** This method allows the TA / Instructor to create an evaluation form.
- **private void activatePeerReviewPeriod() :** This method allows the TA / Instructor to start the peer review period.
- **private void createGroupEvalutionForm() :** This method allows the TA / Instructor to create an evaluation form for students to evaluate other groups' projects.
- **private void activateGroupReviewPeriod() :** This method allows the TA / Instructor to start the group review period.
- **private void getGroup() :**This method allows the user to get a selected group.
- **private void closePeerReviewPeriod() :** This method allows the TA / Instructor to end the peer review period.
- **private void closeGroupReviewPeriod() :**This method allows the TA / Instructor to end the group review period.
- **private Student[] showAloneStudents() :** This method returns all students that are not in any group.

SystemStrategy Class:

Explanation: This class is responsible for statistical functions in our application. Calculating average and displaying, showing how many students are not in group etc..

Methods:

- **public int getUnassignedStudents() :** This method returns how many students are not assigned in any group.
- **public double getAverageGrade() :** This method returns the average of all groups.
- **public double getAverageGroupGrade() :** This method returns the average grade of a specific group.
- **public Poll getPollTemplate() :** This method returns the poll template.
- **public void deletePollTemplate(int pollId) :** This method allows TA or Instructor to delete already created poll templates.
- **public void addPollTemplate() :** This method allows TA or Instructor to add a poll template.

6.3 Database Object Manager Subsystem

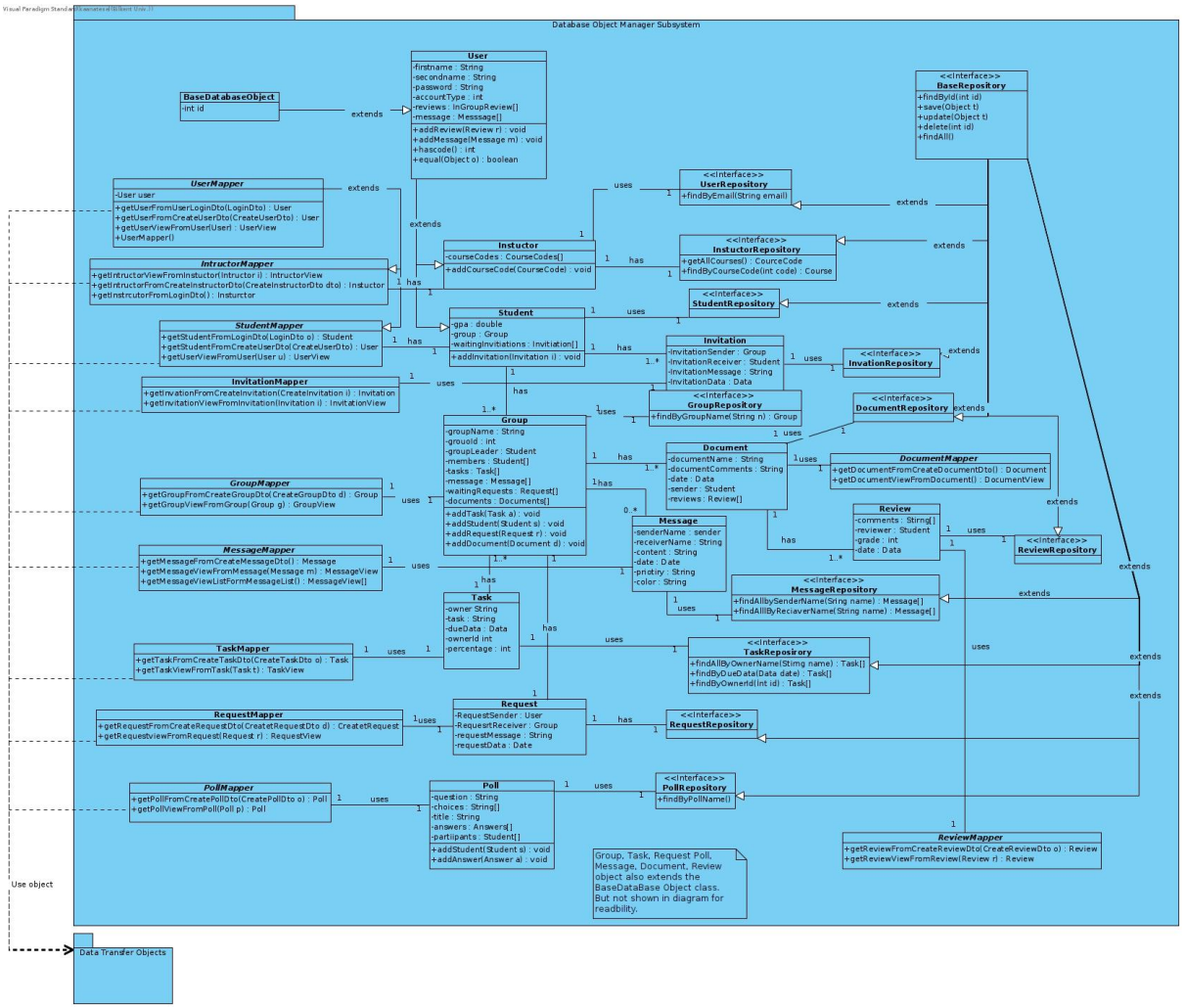

Figure 3: Database Object Manager Subsystem Class Diagram.

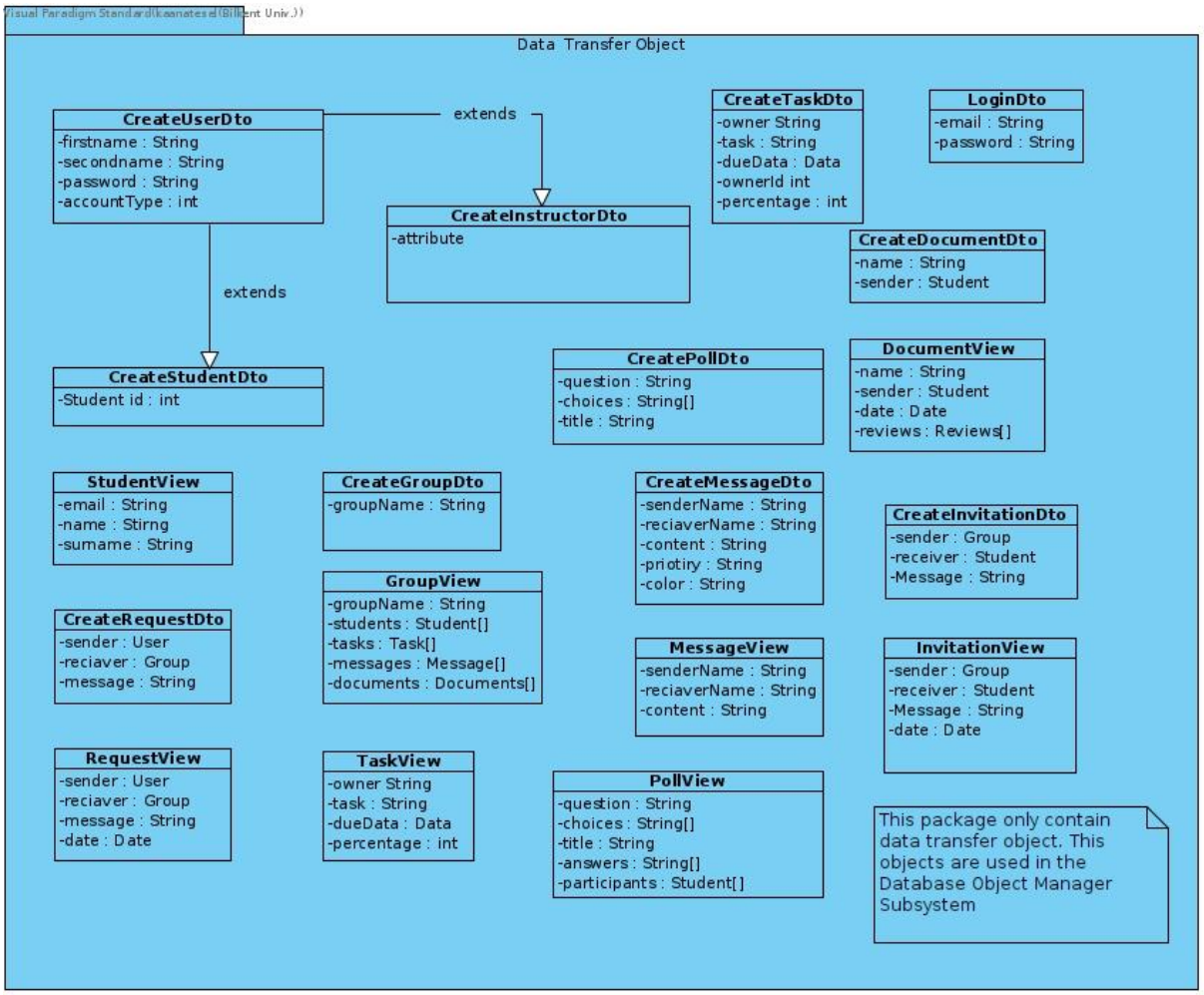

Figure 4: Data Transfer Objects.

BaseDatabaseObject Class:

This class is a base class of any database object. Thus all database objects will extend from this object.

Attributes:

- private int Id : This is id of the database object.

BaseRepository Interface:

Methods:

- **public findById(int):** This finds the object from the database by id.

- **public ObjectView save(Object):** This saves the object to the database.
- **public ObjectView update(Object):** This update the objects in the database.
- **public void delete(int):** Delete the object from database.
- **public List<Object> findAll():** Find all objects in database.

User Class:

Contains properties of User class. It also has getters and setters.

Attributes:

- **private String firstname:** First name of the user.
- **private String lastname:** Last name of the user.
- **private String accountType:** This is the account type of the user. Student,TA or instructor
- **private String password**: password of the user.
- **private Message[] message:** List of messages of the user.
- **private InGroupReview[] reviews:** In group review list of the user

Methods:

- **Getters and Setters**
- **public void addReview(Review):** Add review to object
- **public void addMessage(Message):** Add message to message list
- **public int hashcode():** Return the hash code.
- **public boolean equal(Object):** Check if the object are equal.

UserMapper Class:

Attributes:

- **private User user:** User objects for usermapper

Methods:

- **public User getUserFromUserLoginDto(LoginDto):** Get user from login data transfer object.
- **public User getUserFromCreateUserDto(CreateUserDto):** Get user from create user data transfer object.
- **public UserView getUserViewFromUser(User):** Get user view from user object.
- **public UserMapper():** Constructor

UserRepository Interface:

Methods:

- **public User findByEmail(String):** Find user by user name

Instructor Class:

Contains properties of Instructor class. It also has getters and setters.

Attributes:

- **private CourseCodes[] courseCodes:** List instructors course code.

Methods:

- Getters and Setters
- **public void addCourseCode(CourseCode):** Add course code to course code list.

InstructorMapper Class:

This class maps the instructor and related objects.

Methods:

- **public Instructor getInstructorFromLoginDto(LoginDto):** Get Instructor from login dto.
- **public Instructor getInstructorFromCreateInstructorDto (CreateInstructorDto):** Get Instructor from create Instructor dto.
- **public InstructorView getInstructorViewFromInstructor (Instructor):** Get instructor view object from Instructor object.

InstructorRepository Interface:

This class is responsible for Instructor object database operations.

Methods:

- **public CourseCode findByCourseCode(int):** Find by instructor by course codes.
- **public Course getAllCourses():** Return all courses and corresponding Instructors from the database.

Student Class:

Contains properties of Instructor class. It also has getters and setters.

Attributes:

- **private double gpa:** Stores gpa of the student

- **private Group group:** The group that the student belongs to.
- **private Invitation[] waitingInvitations:** Unanswered invitations of the users.

Methods:

- **public void addInvitation(Invitation):** Add new invitation to the invitations list.

StudentMapper Class:

This class maps the student and related objects.

Methods:

- **public Student getStudentFromLoginDto(LoginDto):** Get student from login dto.
- **public User getStudentFromCreateStudentDto (CreateStudentDto):** Get student from create user dto
- **public UserView getStudentViewFromStudent(Student):** Get student view object from student.

Group Class:

Explanation: This class is a representation of a group.

Attributes:

- **private String groupName:** Stores group name
- **private int groupId:** Stores group id
- **private Student groupLeader:** Stores group leader student
- **private Student[] members:** Stores groups member list.

- **private Task[] tasks:** Stores list of group tasks
- **private Message[] message:** Message list of group
- **private Request[] waitingRequests:** Groups waiting invitations.
- **private Document[] documents:** Groups document list.

Methods:

- **public void addTask(Task):** Add new task to task list
- **public void addStudent(Student):** Add new student to task list
- **public void addRequest(Request):** Add new request task list
- **public void addDocument(Document):** Add new document to task list

GroupMapper Class:

This class maps the group and related objects.

Methods:

- **public Group getGroupFromCreateGroupDto(CreateGroupDto):** Get group from create group dto.
- **public GroupView getGroupViewFromGroup(Group i):** Get group view from group object.

GroupRepository Interface:

This class is responsible for group object database operations.

Methods:

- **public Group findByGroupName(String):** Find group by name.

Task Class:

Explanation: Task class is for arranging schedules.

Attributes:

- **private User owner:** Owner of the task.
- **private String task:** Task description.
- **private Date dueDate:** Due date of the task.
- **private int ownerId:** Owners id.
- **private int percentage:** Percentage of the task.

TaskMapper Class:

This class maps the task and related objects.

Methods:

- **public Task getTaskFromCreateTaskDto(CreateTaskDto):** Get task from create task dto.
- **public TaskView getTaskViewFromTask(Task):** Get task view from task object.

TaskRepository Class:

This class is responsible for task object database operations.

Methods:

- **public Task[] findAllByOwnerName(String):** find all tasks which belong to given user name.

- **public Task[] findAllByDueDate(String):** find all tasks by their due dates.
- **public Task[] findAllByOwnerId(String):** find all tasks by owner id.

Message Class:

Explanation: This class includes all attributes related to messages.

Attributes:

- **private String senderName:** Sender name.
- **private String receiverName:** Receiver name.
- **private String content:** Content of the message.
- **private Date date:** Send date of the message.
- **private String priority:** Priority of the message.
- **private String color:** Color code of the color.

MessageMapper Class:

This class maps the message and related objects.

Methods:

- **public Message getMessageFromCreateMessageDto():** Get message object from create message dto.
- **public MessageView getMessageViewFromMessage(Message):** Get message view from message object.
- **public MessageView[] getMessageViewListFromMessageList():** Get message view list from message list.

MessageRepository Interface:

This class is responsible for message object database operations.

Methods:

- **public Message[] findAllBySenderName(String):** Find all messages by looking at the sender name.
- **public Message[] findAllByReceiverName(String):** Find all messages by looking at receiver name.

Poll Class:

Explanation: Poll class is for enabling users to create polls where they can create their own poll questions.

Attributes:

- **private String question:** Stores question of the poll.
- **private String[] choices:** Stores list of choices for the poll.
- **private String title:** Stores title of the poll.
- **private HashMap answers:** This attribute keeps the answers and the number of votes of that answer.
- **private Student[] participants:** Stores student list who participated in the poll for avoiding dupes.

Methods:

- **public void addStudent(Student):** Adds student to the poll.
- **public void addAnswer(String):** Adds an answer to the poll.

PollMapper Class:

This class maps the poll and related objects.

Methods:

- **public Poll getPollFromCreatePollDto(CreatePollDto):** Returns the poll instance from the database.
- **public Poll getPollViewFromPoll(Poll):** Returns PollView instance from the database.

PollRepository Interface:

Methods:

- **public Poll findByPollName(String):** Returns poll instance by providing poll name.

Invitation Class:

Explanation: This class is responsible for notifying students that they are invited to a group.

Attributes:

- **private Group invitationSender:** Stores the invitation sender's group instance.
- **private Student invitationReceiver:** Stores the invitation receiver's student instance.
- **private Date invitationDate:** Stores the date information about when the invitation was sent.

- **private String invitationMessage:** Stores the invitation message.

InvitationMapper Class:

This class maps the invitation and related objects.

Methods:

- **public Invitation getInvitationFromCreateInvitationDto (CreateInvitation):** Returns invitation instance from create invitation dto.
- **public InvitationView getInvitationViewFromInvitation(Invitation):** Returns invitation view from invitation dto.

Document Class:

Explanation: Document class is for documents which are uploaded to the system by users.

Attributes:

- **private String documentName:** Stores document name.
- **private String documentComments:** Stores document description.
- **private Date date:** Stores upload date of the artifact.
- **private User sender:** Stores the instance of uploader user.
- **private Review[] reviews:** Stores list of reviews.

DocumentMapper Class:

This class maps the documents and related objects.

Methods:T

- **public Document getDocumentFromCreateDocumentDto():**

  Returns document from create database object.

- **public DocumentView getDocumentViewFromDocument():**

  Returns documentView instance from document.

Request Class:

Explanation: Invitation request from group to student in order to attract their attention.

Attributes:

- **private Student requestSender:** Stores the sender's student instance.
- **private Group requestReceiver:** Stores receiver's group instance.
- **private String requestmessage:** Stores the request message.
- **private Date requestDate:** Stores the date when the request is created.

RequestMapper Class:

Methods:

- **public CreateRequest getRequestFromCreateRequestDto (CreateRequestDto):** Returns the request instance from the given CreateRequestDto instance.

- **public RequestView getRequestViewFromRequest(Request):** Returns the requestView instance from the request object.

Review Class

Explanation: This class includes all reviews to the finished project. Both students,TA's and instructors could give reviews to other projects and give grades. In addition to giving grades students could give a suggestion and TA's & Instructors may include feedback in their review.

Attributes:

- **private String[] comments:** Stores list of comments in review.
- **private Student reviewer:** Stores student instance of reviewer.
- **private int grade:** Stores grade of review.
- **private Date date:** Stores date of given review.

ReviewMapper Class:

Methods:

- **public Review getReviewFromCreateReviewDto (CreateReviewDto):** Returns review from CreateReview instance.
- **public ReviewView getReviewViewFromReview(Review):** Returns ReviewView instance from Review object.

6.4 Database Object Manager Subsystem

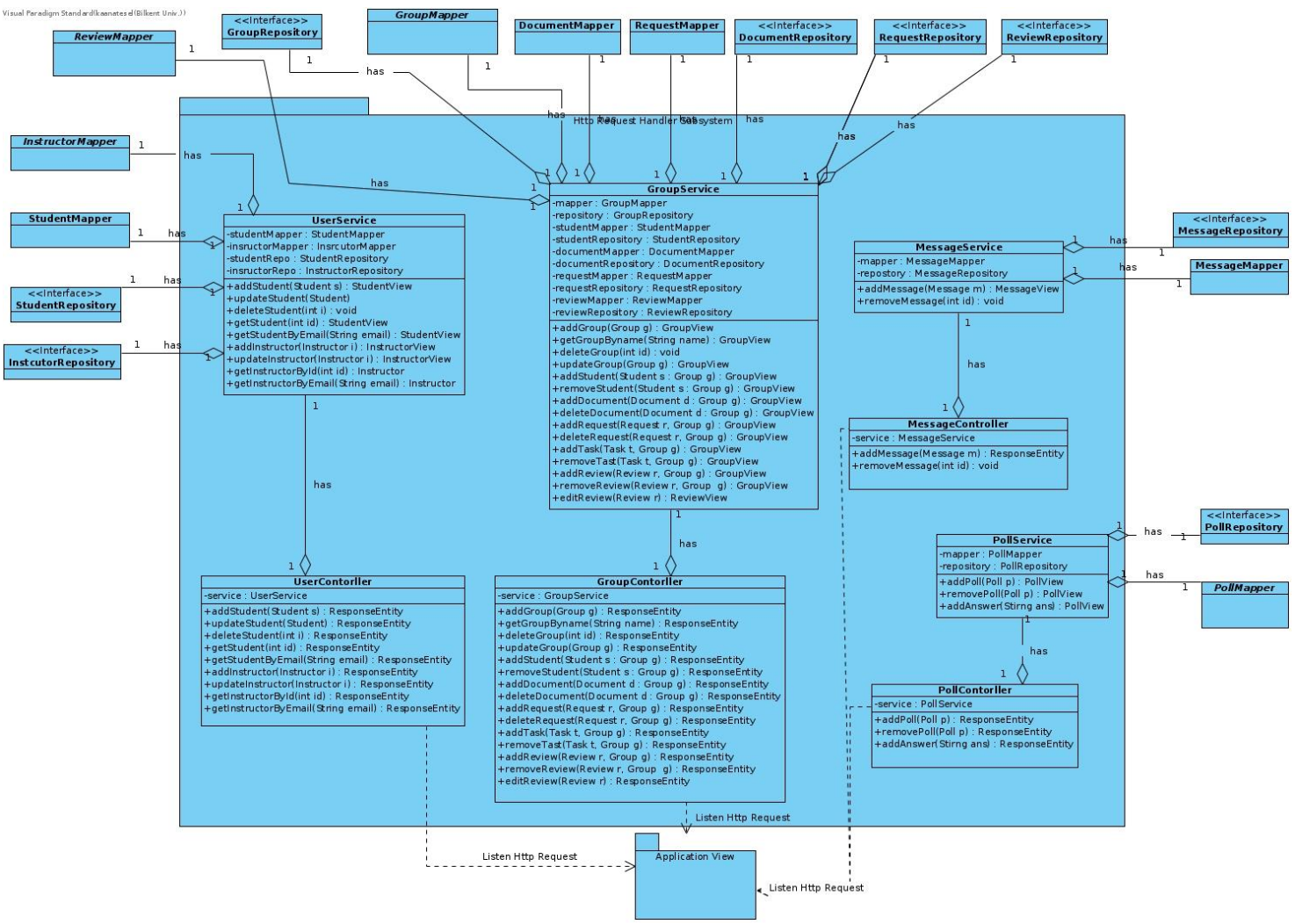

Figure 5: Database Object Manager Subsystem

UserService Class:

Explanation: Handle database operations of user objects.

Attributes:

- private StudentMapper studentMapper
- private instructorMapper instructorMapper
- private studentRepository studentRepo
- private instructorRepository instructorRepo

Methods:

- public StudentView addStudent (Student s)
- public void updateStudent (Student)
- public void deleteStudent (int i)
- public studentView getStudent (int id)
- public studentView getStudentByEmail (string email)
- public instructorView addInstructor (Instructor i)
- public instructorView updateinstructor (Instructor i)
- public Instructor getInstructorById (int id)
- public Instructor getInstructorByEmail (String email)

UserController Class:

Explanation: Handles the user object related http requests.

Attributes:

- private UserService service

Method

- public ResponseEntity addStudent (Student s)
- public ResponseEntity updateStudent (Student)
- public ResponseEntity deleteStudent (int i)
- public ResponseEntity getStudent (int id)
- public ResponseEntity getStudentByEmail (String email)
- public ResponseEntity addInstructor (Instructor i)
- public ResponseEntity updateInstructor (Instructor i)
- public ResponseEntity getInstructorById (int id)

- public ResponseEntity getInstructorByEmail (String email)

GroupService Class:

Explanation: Handle database operations of group objects.

Attributes:

- private GroupMapper mapper
- private GroupRepository repository
- private StudentMapper studentMapper
- private StudentRepository studentRepository
- private DocumentMapper documentMapper
- private DocumentRepository documentRepository
- private RequestMapper requestMapper
- private RequestRepository requestRepository
- private ReviewMapper reviewMapper
- private ReviewRepository reviewRepository

Methods:

- public GroupView addGroup (Group g)
- public GroupView getGroupByName (String name)
- public void deleteGroup (int id)
- public GroupView updateGroup (Group g)
- public GroupView addStrudent (Student s)
- public GroupView removeStudent (Student s : Group g)
- public GroupView addDocument (Document d : Group g)
- public GroupView deleteDocument (Document d : Group g)

- public GroupView addRequest (Request r, Group g)
- public GroupView deleteRequest (Request r, Group g)
- public GroupView addTask (Task t, Group g)
- public GroupView removeTask (Task t, Group g)
- public GroupView addReview (Review r, Group g)
- public GroupView removeReview (Review r, Group g)
- public ReviewView editReview (Review r)

GroupController Class:

Explanation: Handles the group object related http requests.

Attributes:

- private service

Methods:

- public ResponseEntity addGroup (Group g)
- public ResponseEntity getGroupByName (String name)
- public ResponseEntity deleteGroup (int id)
- public ResponseEntity updateGroup (Group g)
- public ResponseEntity addStudent (Student s)
- public ResponseEntity removeStudent (Student s : Group g)
- public ResponseEntity addDocument (Document d : Group g)
- public ResponseEntity deleteDocument (Document d : Group g)
- public ResponseEntity addRequest (Request r, Group g)
- public ResponseEntity deleteRequest (Request r, Group g)
- public ResponseEntity addTask (Task t, Group g)
- public ResponseEntity removeTask (Task t, Group g)

- public ResponseEntity addReview (Review r, Group g)
- public ResponseEntity removeReview (Review r, Group g)
- public ResponseEntity editReview (Review r)

MessageService Class:

Explanation: Handle database operations of message objects.

Attributes:

- private MessageMapper mapper
- private MessageRepository repository

Methods:

- public MessageView addMessage (Message m)
- public void removeMessage (int id)

MessageController Class:

Explanation: Handles the message object related http requests.

Attributes:

- private MessageService service

Methods:

- public ResponseEntity addMessage (Message m)
- public void removeMessage (int id)

PollService Class:

Explanation: Handle database operations of poll objects.

Attributes:

- private PollMapper mapper
- private PollRepository repository

Methods:

- public PollView addPoll (Poll p)
- public PollView removePoll (Poll p)
- public PollView addAnswer (String ans)

PollController Class:

Explanation: Handles the poll object related http requests.

Attributes:

- private PollService service

Methods:

- public ResponseEntity addPoll (Poll p)
- public ResponseEntity removePoll (Poll p)
- public ResponseEntity addAnswer (String ans)

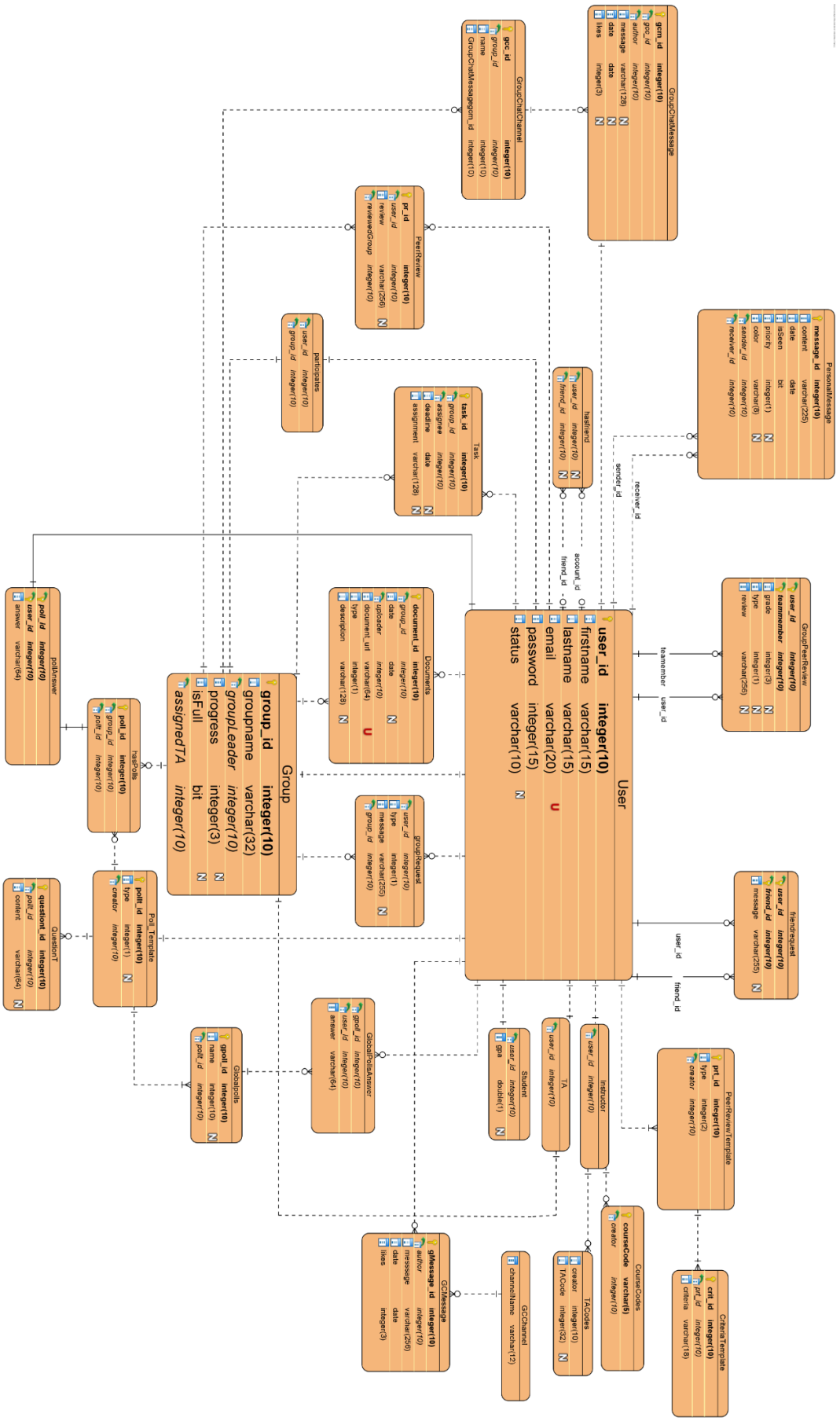

Figure 6: ER diagram

7. Glossary & references

[1] Produle, “Wireframe Tools, Prototyping Tools, UI Mockups, UX Suite, Remote designing,” *MockFlow*. [Online]. Available: https://mockflow.com/. [Accessed: 16-Mar-2021].