Bilkent University

Department of Computer Engineering

Senior Design Project

Project Name: StudyB

Low Level Design Report

Alara Yaman
Betim Doğan
Berfu Deniz Kara
Mert Özerdem
Sera Fırıncıoğlu

Supervisor: Özcan Öztürk

Innovation Expert: Aras Bilgen

Jury Members: Varol Akman and Eray Tüzün

Low Level Design Report Feb 17, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

1.1 Object Design Trade-Offs

1.1.1 Usability vs. Functionality

Since the aim of StudyB is to consolidate the university life of students by helping each other, it will be an application that will require minimal time to set up and start using, and therefore we prioritize being minimalist so that it is not disturbing with more functionality than necessary in terms of user interfaces. Simple and understandable settings and buttons will be used in the application for ease of use and complex functionality will be put on the server side.

1.1.2 Space vs. Time

Since StudyB stores a variety of data, including students' comments, messages, documents, images, videos, and private information, the processing time of the system increases and this causes a slowdown in the response time between the operation. But we value time more than space so we will choose to speed up the user interaction processes and due to this, will have a bigger data space, we are willing to pay for the extra space requirements of the server.

1.1.3 Compatibility vs. Extensibility

For an application, compatibility is a crucial factor in reaching potential users. Since our project is planned to be released as only Android application, compatibility is a very important design trade-off for us to ensure to create a big user pool for the application. But in the future, we plan to extend our project as an IOS and a web based application as well to get bigger.

1.1.4 Security vs. Cost

Our project will hold valuable user data such as user name, user password and users’ school e-mail addresses so security is much

more important than the cost. StudyB is designed to favor security over cost by having a well designed database structure.

1.2 Interface Documentation Guidelines

In this report, the guideline for interface is explained as follows; Pascal Case convention is used for class names. Dromedary Case is used for variable names such as variableName. Method names are defined as methodName().

Class Name
Name of the Class
Attributes
Visibility of Attribute, Name of Attribute : Type of Attribute
Methods
Visibility of Method, Name of Method (Parameters of Method) : Signature of Method

1.3 EngineeringStandards

This report meets UML guidelines for the details of the class interfaces, diagrams, scenarios use cases, subsystem compositions and hardware depictions. UML is a widely used, easy-to-use way of generating such diagrams, and since it is the approach taught at Bilkent University, we choose to use it in the pages below. The report meets IEEE standards for the citations, which is a widely used approach and the one favoured at the Bilkent University.

1.4 Definitions, Acronyms and Abbreviations

HTML: Hypertext Markup Language

User: A user of StudyB with a valid account.

JSON: JavaScript Object Notation

XML: Extensible Markup Language

2. Packages

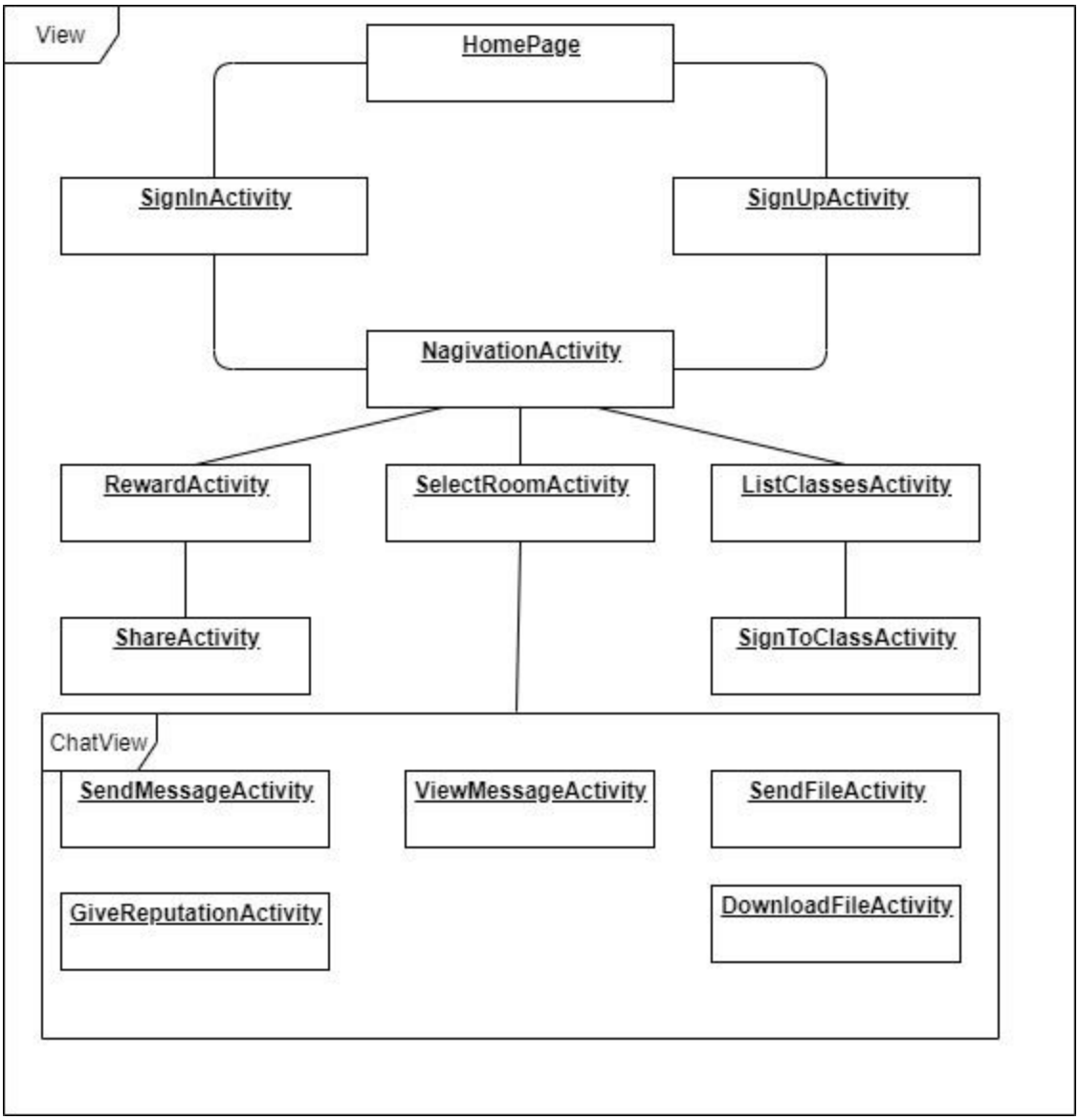
2.1 Client

The client side of StudyB consists of the mobile applications that will be running on Android devices. The client side of the system consists of two packages, View and Controller packages. After getting the user information such as email address (must be our university mail address), the client will navigate to the Home page where user will see the lectures. When sending or retrieving information from StudyB server, the Controller package comes into play so that trip information such as flight details or places to see in the travelled location will be requested by the client side. The View package will consist of classes that will display these information to users by creating the interfaces.

Basically, the client side is the user end of the application program. Using the mobile application that will be developed Android, the user will interact with the system. When user enters their Login or Sign Up information in the application, the client will send the login request to the server and the user will be logged in to the system. When email information is obtained by client, another request will be sent to server and the client will load the schedule and lecture. They can choose which lectures to attend and upload their exam dates to get exam tips from the application.

StudyB client subsystem will be implemented using Model-View-Controller design pattern. Controller subsystem will be responsible for the connection between client and server, when data is sent, controller collects it and when responses are taken for requests, controller collects it. View subsystem will be responsible for interface operations. Displaying pages or taken data on the screen will be done by the view subsystem. Since all the data of the client will be taken from the server, implementing a separate Model subsystem will be trivial, therefore we will only use Controller and View subsystems in the client side.

2.1.1 View



SignInActivity: Class handles sign-in related activity if user has an account on the StudyB application s/he can use this method to class to sign-in.

SignUpActivity: Class handles sign-up related activity if user does not have an account on the StudyB application s/he can use this method to class to create and sign-in.

NavigationActivity: Class handles navigation of the user between reward, classes, chatroom, and homepage of the application.

RewardActivity: Class is responsible for showing user's rewards for participation.

ShareActivity: User can share s/he's rewards online via use of this class.

ListClassesActivity: Class is responsible for showing available class chatroom for user to join.

SignToClassActivity: Shows classes syllabus and participants before signing up to class.

SendMessageActivity: Shows user what s/he is typing and about to send.

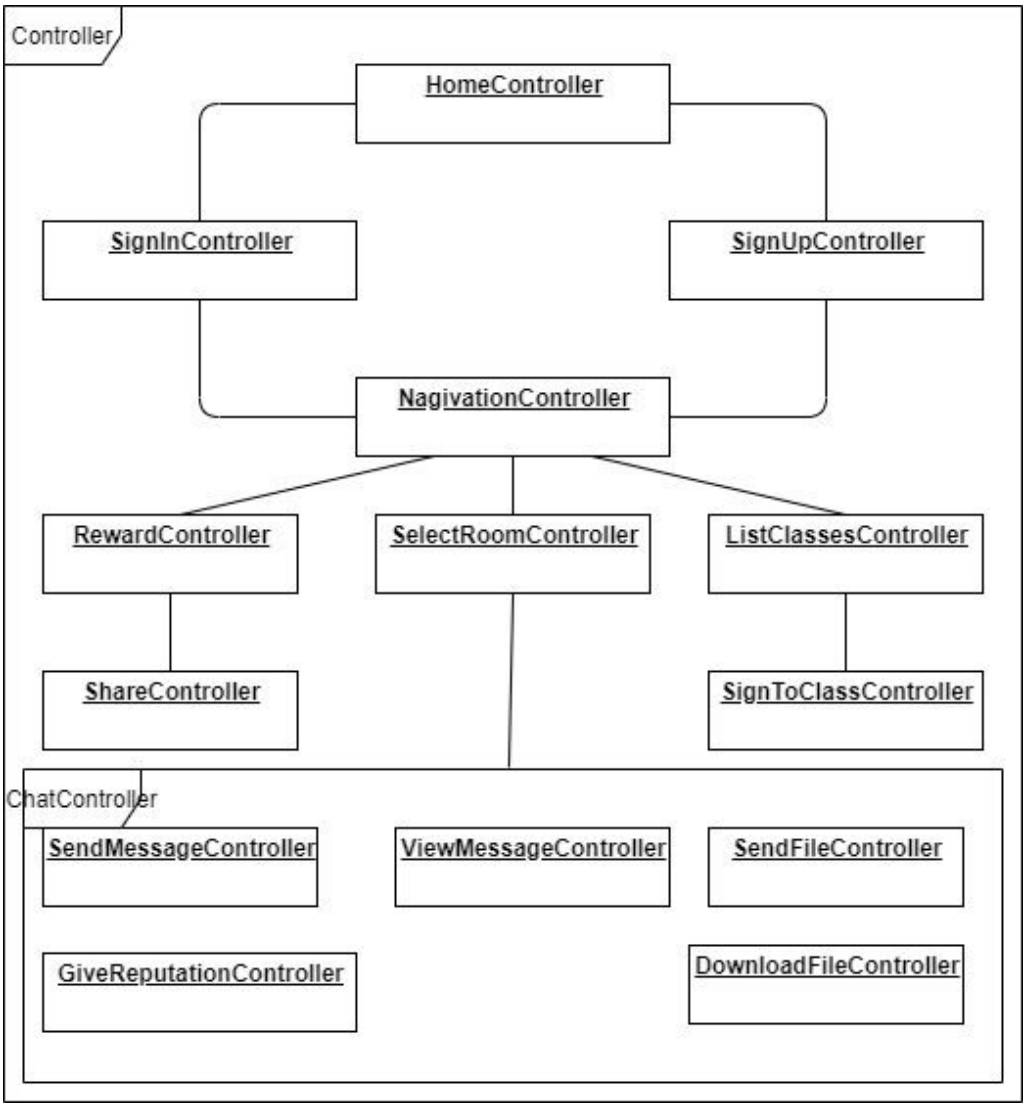
ViewMessageActivity: Class is responsible for displaying previously typed messages on the specific chatroom.

SendFileActivity: Show which file is about to upload to the chatroom.

DownloadFileActivity: Show which file is about to download from the chatroom.

GiveReputationActivity: Users can give reputation for the messages they like or appreciate via using this class's methods.

2.1.2 Controller



SignInController: The class controls functions that are related with the SignInActivity functions which checks validity of the user that wants to enter the site.

SignUpController: The class controls functions that are related with the SignUpActivity functions which checks validity of the user's mail for sign-up the user.

NavigationController: The class controls functions that are related with the NavigationActivity functions which fetches the related information for selected page.

RewardController: The class controls functions that are related with the RewardActivity functions which fetches the related rewards of the user .

ShareController: The class controls functions that are related with the ShareActivity functions which controls users send requests to the outside sources.

ListClassesController: The class controls functions that are related with the ListClassesActivity functions which lists all class chatrooms that are available on the StudyB application .

SignToClassController: The class controls functions that are related with the SignToClassActivity function.

SendMessageController: The class controls functions that are related with the SendMessageActivity functions which sends a request to write a message object to the databese server.

ViewMessageController: The class controls functions that are related with the ViewMessageActivity functions which sends a request to get messages from the database server.

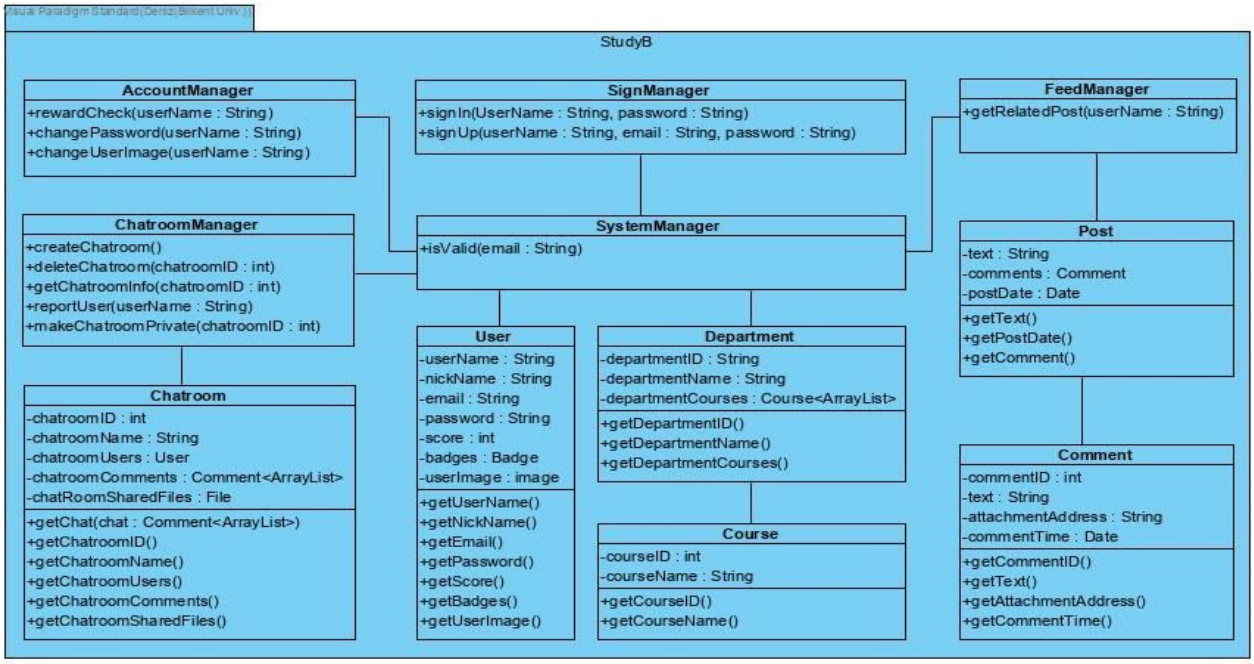
SendFileController: The class controls functions that are related with the SendFileActivity functions which send request to write a file object to the databese server.

DownloadFileController: The class controls functions that are related with the DownloadFileActivity functions which sends a request to get file from the database server.

GiveReputationController: Handles requests that a increase reputation of the specific user via their message.

2.2 Logic

Logic layer is responsible for all major operations of the system. Logic will communicate with database and GUI layers.



SystemManager: this class is used to manage all sub-manager classes.

AccountManager: this class is used to control account activities and scores of the users.

SignManager: this class is used to provide users to sign-in or sign-up to the application.

FeedManager: this class is used to control and manage the main page of the courses.

Post: this class is used to create posts and keep information of the posts in the feed.

Comment: this class is used to create comments and keep information of the comments in posts and chatrooms.

ChatroomManager: this class is used to control the chatroom actions.

Chatroom: this class is used to create a default chat room.

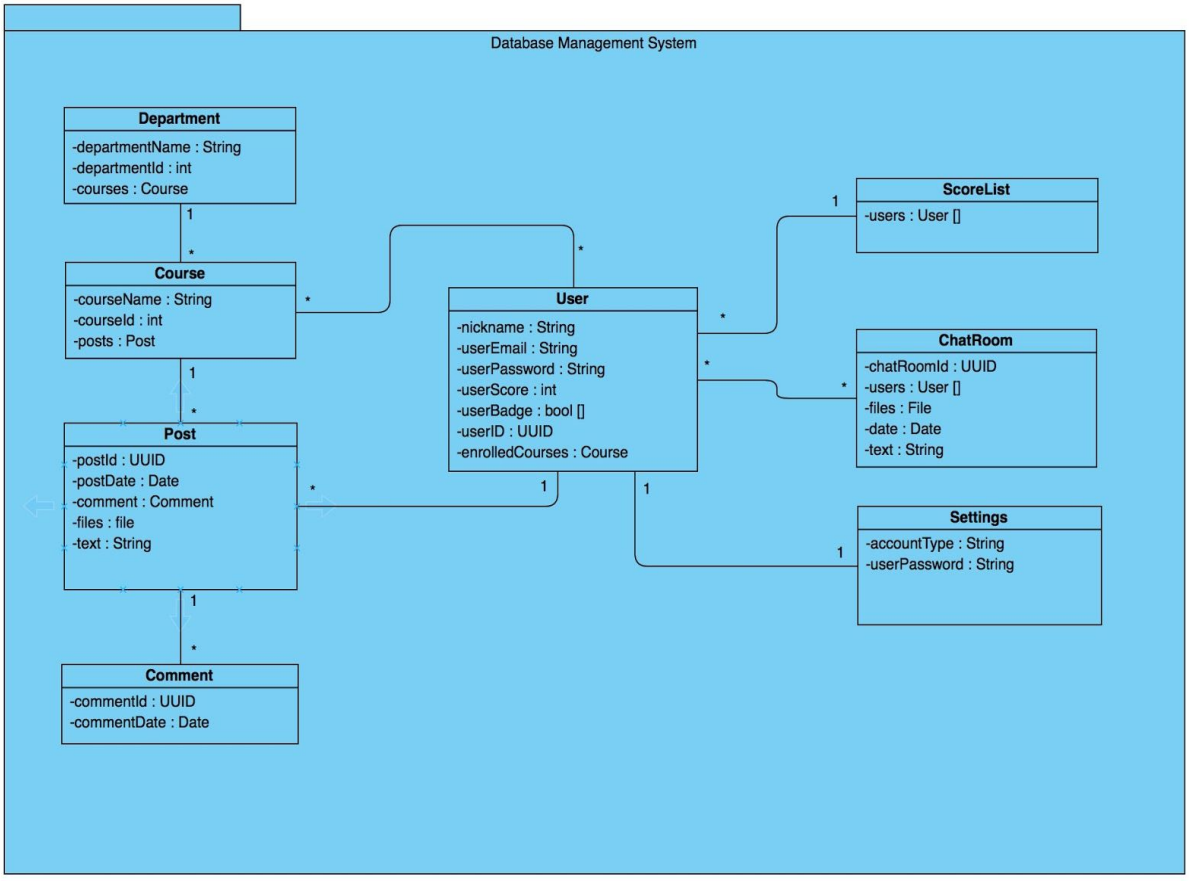
User: this class is used to create users and keep information of the users.

Department: this class is used to create departments and keep information of the departments.

Course: this class is used to create courses and keep information of the courses.

2.3 Data

Logic tier will be in communication with Data Tier for implementation. This tier will manage the database system of the application.



Department : Department class will hold the whole names of the departments with their id and the courses under them.

Course : Course class will keep the course names and id's. Posts will be kept in every under course.

Post : Post class will keep the detailed information of posts with their specifications.

Comment : Comment class will keep the comment id and date to manage every post's comments.

User : User class will keep the data of every user and manage it after registration to the application.

ScoreList : Score list class will manage the list of users according to their score.

ChatRoom : ChatRoom class will manage the conversation between users.

Settings : Settings class will be used to manage account types and passwords.

3. Class Interfaces

3.1 Client Interfaces

3.1.1 View Interfaces

class SignInActivity
This class will be used for sign-in related activities for the users. Authentication will be done by using this class
Attributes
private String Password
private String email
Methods
public void signIn(String email, String Password)
public boolean signInCheck()
public void nagivateHomePage()

class SignUpActivity
This class will be used for sign-up related activities for the users. Authenticity of the user will be checked and created if check is valid for criterias.
Attributes
private String username
private int ID
private String Password
private String email
Methods
public void signUp(String email, String Password, String username)
public boolean validityCheck()

public void nagivateHomePage()

class RewardActivity
Displays the rewards of the user and share buttons.
Attributes
private int rewardID
private String username
Methods
public void dislayRewards(int rewardID, String username)

class ListClassesActivity
This class lists all available class chat rooms for users to join. User can enter and check chat room's contents if s/he wishes.
Attributes
private int[] classID
Private int username
Methods
public void displayClasses(int[] classID)
public void displayClassInfo(int classID)

class ViewMessageActivity
Displays all of the messages that is posted on the specific chat room.
Attributes
private String[][] userMessages
Methods
public void displayMessages(String[][] userMessages)

class ViewFileActivity
Displays all of the files that is posted on the specific chat room.
Attributes
private String[][] userFiles
Methods
public void displayMessages(String[][] userFiles)

3.1.2 Controller Interfaces

class SignInController
This class controls sign-in related activities. Checks if the user is valid user and authentication is true.
Methods
public boolean checkUser(String username, String password)

class SignUpController
This class controls sign-up related activities. Checks if the user is eligable to join application and validity of their passwords and usernames.
Methods
public boolean checkNewUser(String username, String password, String email)
public String returnError(String error)

class NavigationController
This class will be used for navigate through the application. User can navigate home page, chatroom page, rewards page, and select class page.
Methods
public void nagivate(int chatroomID)

class NavigationController
This class will be used for navigate through the application. User can navigate home page, chatroom page, rewards page, and select class page.
Methods
public void nagivate()

class ChatController
This class is responsible for fetching and sending CRUD requests to the back end. Also this class handles chat room related activities
Methods
public message fetchAllMessages(int chatroomID)
public void writeMessage(String username, String message)
public file fetchAllFiles(String username, String fileAddress)
public void writeFile(String fileAddress, String Username)
public void rewardUser(String username)

3.2 Logic Interfaces

class SystemManager
This class is used to manage all sub-manager classes.
Methods
public bool isValid(email : String)

class AccountManager
This class is used to control account activities and scores of the users.
Methods

public void rewardCheck(userName : String)
public void changePassword(userName : String)
public void changeUserImage(userName : String)

class SignManager
This class is used to provide users to sign-in or sign-up to the application.
Methods
public void signIn(userName : String, password : String)
public void signUp(userName : String, email : String, password : String)

class FeedManager
This class is used to control and manage the main page of the courses.
Methods
public void getRelatedPost(userName : String)

class Post
This class is used to create posts and keep information of the posts in the feed.
Attributes
private String text
private Comment comments
private Date postDate
Methods
public String getText()
public Comment getComment()
public Date getPostDatet()

class Commnet
This class is used to create comments and keep information of the comments in posts and chatrooms.
Attributes
private int commentID
private String text
private String attachmentAddress
Private Date commentTime
Methods
public int getCommentID()
public String getText()
public String getAttachmentAddress()
public Date getCommentTime()

class ChatroomManager
This class is used to control the chatroom actions.
Methods
public ChatRoom cretaeChatroom()
public void deleteChatroom(chatroomID : int)
public ArrayList getChatroomInfo(chatroomID : int)
public void reportUser(userName : String)
public void makeChatroomPrivate(chatroomID : int)

class Chatroom
This class is used to create a default chat room.
Attributes
private int chatroomID
private String chatroomName
private User chatroomUsers
private Comment <ArrayList> chatroomComments
private File chatroomSharedFiles
Methods
public Comment<ArrayList> getChat()
public int getChatroomID()
public String getChatroomName()
public User<ArrayList> getChatroomUsers()
public Comment<ArrayList> getChatroomComments()
public File getChatroomSharedFiles()

class User
This class is used to create users and keep information of the users.
Attributes
private String userName
private String nickName
private String email
private String Password
private int score
private Badge badges

private image userImage
Methods
public String getUserName()
public String getNickName()
public String geEmail()
public String getPassword()
public int getScore()
public Badge getBadges()
public Image getUserImage()

class Department
This class is used to create departments and keep information of the departments.
Attributes
private String departmentID
private String departmentName
private Course<ArrayList> departmentCourses
Methods
public String getDepartmentID()
public String getDepartmentName()
public Course<ArrayList> getDepartmentCourses()

class Course
This class is used to create courses and keep information of the courses.
Attributes

private int courseID
private String courseName
Methods
public int getCourseID()
public String getCourseName()

3.3 Data Interfaces

class Department
This class will be used to keep and manage detailed information about departments of Bilkent University.
Attributes
private String departmentName
private int departmentId
private Course courses

class Course

This class will keep courses of every department with detailed information and management of them.
Attributes
private String courseName
private int courseId
private Post posts

class Post
This class will handle the posts with their specific information on the feed page.
Attributes
private String postId
private Date postDate
private Comment comment
private String text
private File files

class Comment
This class will hold the information of every post that is attached to any post.
Attributes
private UUID commentID
private Date commentDate

class User

This class will store the information of every single user.
Attributes
private String nickname
private String userEmail
private String userPassword
private int userScore
private bool [] userBadge
private UUID userID
private Course enrolledCourses

class ScoreList
This class will store the users as a list according to their userScore.
Attributes
private User [] users

class ChatRoom
This class will store the files that are shared in the chat room.
Attributes
private UUID chatRoomId
private User [] users
private File files
private Date date

private String text

class Settings
This class will store the account type and user password of the user.
Attributes
private String accountType
private String userPassword

4. Glossary

Our server runs on a Host OS, which runs on Ubuntu. Above this layer, we are planning to integrate the Docker Engine in order to manage our server side applications. Docker Engine works with container images, which according to the definition in Docker's website "are an abstraction at the app layer that packages code and dependencies together"[10]. There can be multiple containers in a host OS, and for that reason, in our server, we are planning to have a container/s depending on the traffic to run our server side applications which are in Node JS. The server will still be similar in terms of design. *

5. References

[1]”Study Blue” https://www.studyblue.com [Accessed: February 10]
[2] “Study Buddy Mobile” https://www.studybuddymobile.com [Accessed:February 11]
[3] “My Study Life“ https://www.mystudylife.com [Accessed: February 11]
[4] “E-ödev” https://eodev.com [Accessed: February 12]
[5]”Udemy” https://www.udemy.com [Accessed: February 12]
[6]”Coursera” https://www.coursera.org[Accessed:February 16]
[7]”Tutorials Point” https://www.tutorialspoint.com [Accessed: February 16]
[8]”w3Schools” https://www.w3schools.com[Accessed:February 16]