SWE 316
Software Design and Architecture

Design Project - Phase 2

By

Team 8
Hadi Albinsaad - 201621460
Hashim Al Ghamdi - 201617460
Salman Al Ghamdi - 201730930
Salem Bamukhier - 201646760

For

Dr. Moataz Ahmed

Phase 3 Modifications of Previous Phases:

- *Signature of methods have been changed and diagrams have been updated.*
- *Package diagram have been inserted to section 4 of the document.*
- *Sequence diagrams have been provided to show integration of facades.*
- *Two different technologies have been provided and compared with the MEAN stack.*

1. INTRODUCTION

1.1. Purpose

The purpose of this software design document is to provide a description of the design of KFUPM Universe fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to build.

1.2. Scope

KFUPM Universe is a web application that offers a variety of social web services that provide its users with the sufficient amount of academic knowledge for them to thrive and prosper in their academic and social life at KFUPM.

1.3. Overview

This document will go through the system overview where it will give a brief description of the system and its different components. Then, it will display the technological choices made for this project and how they will be deployed to work together. Then, it will provide a high-level of the system architecture. After that, it will go through the component design and the user interface design of the system. Then it will go through the requirements matrix to trace the requirements with the functionalities.

1.4. Reference Material

- SRS v3.0
- KFUPM Universe System Model
- Software Engineering by Somerville, 9th Edition

1.5. Definitions and Acronyms

- *Deployment:* The process of making a software system available for use.

2. SYSTEM OVERVIEW

The system is a web-based system that will allow KFUPM students to rate courses, ask questions, organize “Talkabouts” and have an all-round enhanced social experience in the university. Students can also receive badges as a form of encouragement. Students will be able to register using their KFUPM emails.

3. TECHNOLOGY AND DEPLOYMENT ARCHITECTURE

Performance Requirements

Hardware & Software Requirements to Run the System (Server Side) have been selected with putting in consideration that the server should handle running the system for 24/7 and handling 2000 users simultaneously. Any maintenance operation should not exceed 8 hours.

Environmental Requirements

A weekly backup of the database must be created. The server must be placed in a room with good conditions (Temperature < 27 degrees and low humidity).

- **Hardware**

  3.0 MHz Intel Core2 Quad Processor or Greater

  16GB of RAM

  15TB Available Disk Space

  High Bandwidth Internet Connection

- **Software**

  After comparing MEAN, MERN, & LAMP stacks, MERN & LAMP were eliminated due to the following:

  LAMP:

  LAMP has apache, which takes more time to setup than Express.JS in two other stacks. It also uses PHP versus Node in the other two stacks. Node is more modern and faster than PHP.

  MERN:

  MERN is not really different from MEAN. The one and only different is MERN uses React where MEAN uses Angular. React is supported by Facebook where Angular is supported by Google. Moreover, React has a deeper learning curve than Angular, hence it will lead to extended project time. it’s also much powerful than React. Angular provides a routing, animations, UI components shipped within the framework where in React you will have to use external libraries.

MEAN stack has been selected due to the following:

MEAN – The Jock Stack

- MongoDB
- ExpressJS
- AngularJS
- NodeJS

MEAN is simple as in there’s only one language to learn (JavaScript), and everything else consists of libraries and frameworks.

Pros:

MEAN is entirely based on JavaScript, which makes it easy to translate. Frontend developers already know JavaScript, so learning the backend is a breeze compared to other web stacks. Being on the same page also helps communication between frontend and backend developers. MEAN’s use of JavaScript allows it to be incredibly fast and easy to scale.

NodeJS specifically enhances the performance as it runs faster than other web servers like Apache. B

being an event-driven architecture helps, and so does the amount of optimization and development it is witnessing due to Google and Microsoft.

Cons:

MEAN isn’t as stable as LAMP because it’s so new. It doesn’t have a large community and therefore provides less support than other stacks. There’s the possibility of having more problems and not having a forum to turn to for troubleshooting.

Hardware & Software Requirements to Run the System (Client Side) have been defined with putting in consideration that the device will be efficient to run the application.

- Hardware

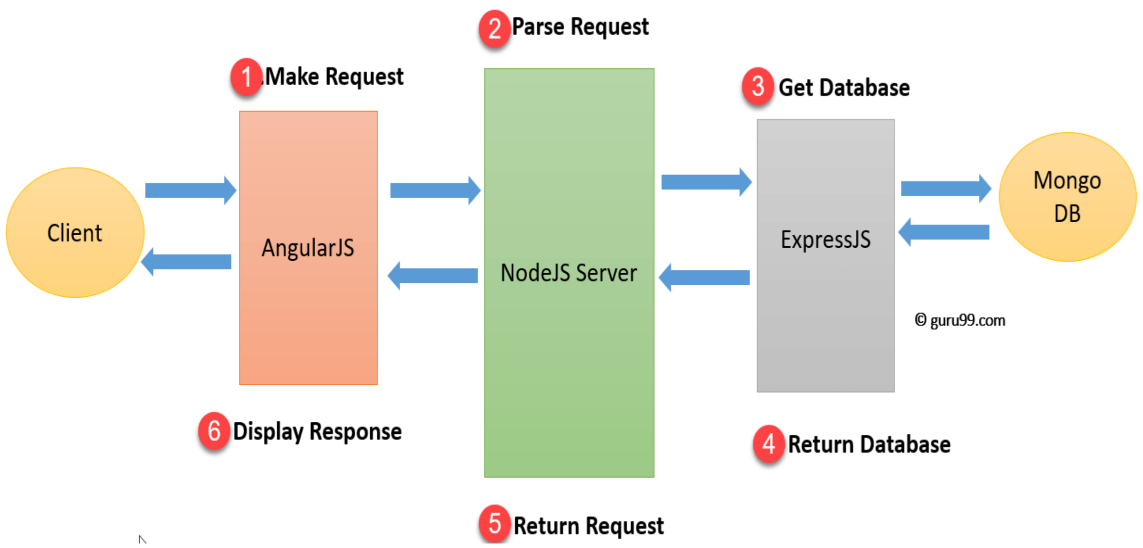
  Average PC Specs (1.5 MHz CPU / 256 RAM)

- Software

An operating system (Windows/Mac/Linux)

A web browser with JavaScript enabled.

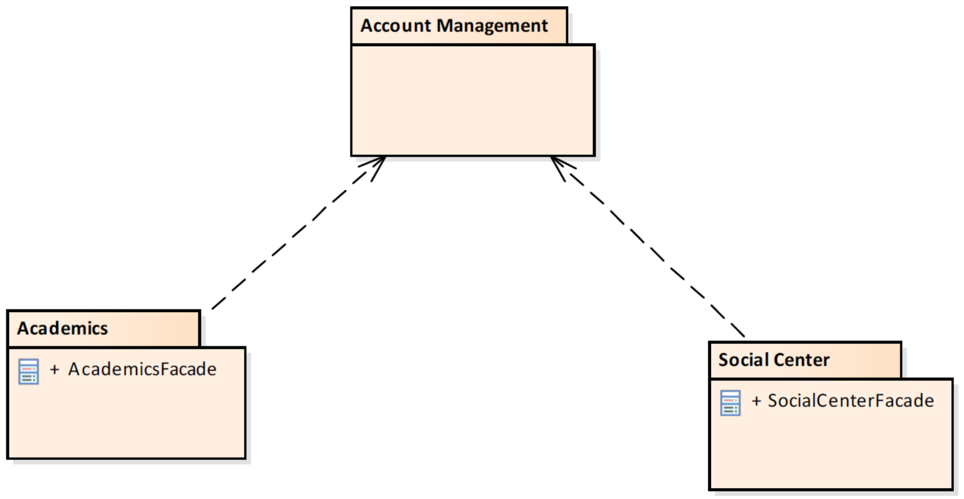
Depiction of the deployment architecture:



Taken from [https://www.guru99.com/mean-stack-developer.html]

4. SYSTEM ARCHITECTURE

4.1. Architectural Design



1. Module 1: Account Management and have will have the following classes:
   - Account
   - Student
   - User
   - Role
   - Privilege
   - AcademicClass

2. Module 2: Academics and have will have the following classes:
   - Major
   - Question
   - Course
   - Answer
   - Comment
   - Review
   - Content
   - Talkabout
   - Tag
   - RelatedMaterial
   - Badge

3. Module 3: Social Center and have will have the following classes:
   - Post-it
   - RoomamateApplication
   - RoommateRequest
   - PublicAnnouncement
   - Event
   - Message
   - LostAndFound
   - KFUPMStatus

4.2. Decomposition Description

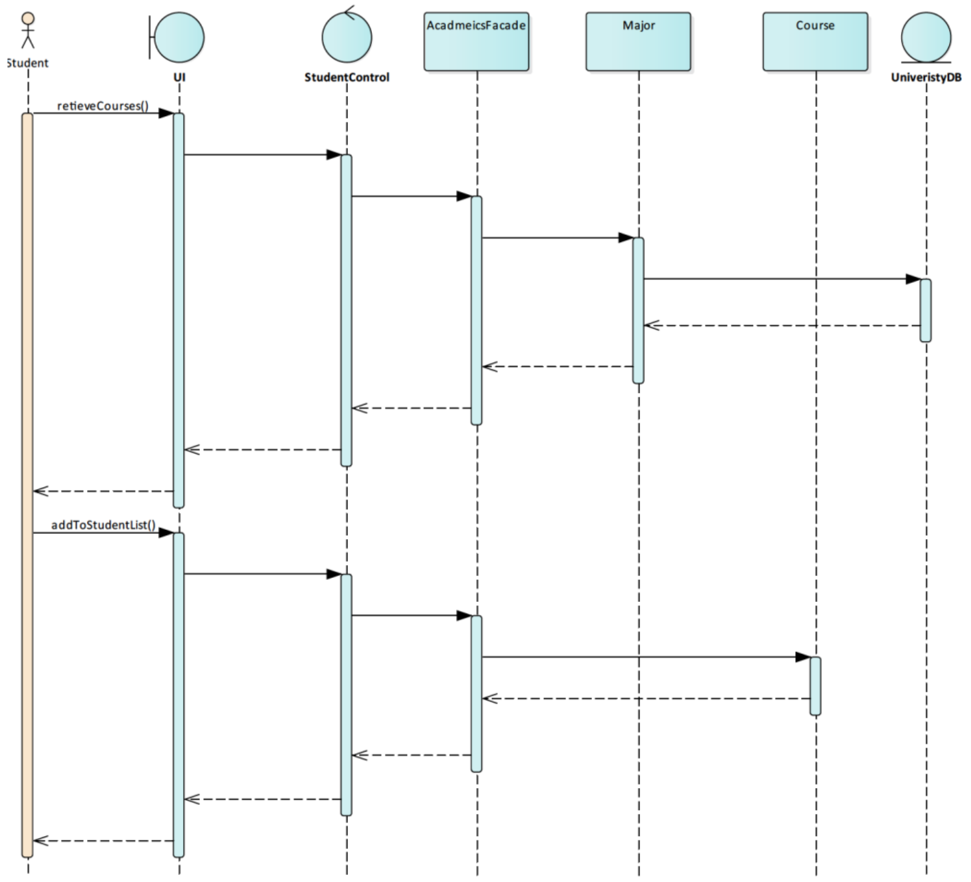
The system was divided into the three modules or packages as seen in the previous segment.

Package one: Account management

This module is responsible for the registration of users and overall management of accounts and will be communicating with the other modules.

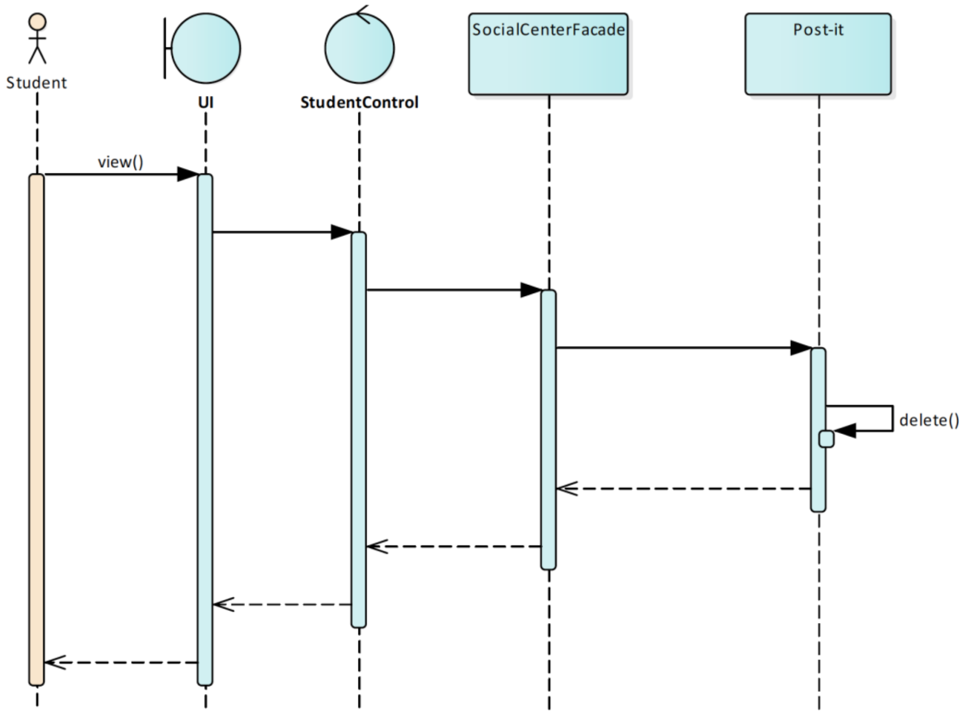
Package two: Academics

This module contains all elements that are related to academics. This package will have a façade to supply the account management package with the needed functionalities. The façade is going to be the wall between The User object from Account Management package and other classes like Course classes from Academics package. An example is provided in the picture bellow, using the methods retrieveComments() and insertToCourseList() in the AcademicsFacade.



Package three: Social

This module contains all the social elements of the system. This package will, also, have a façade, so, the account management package shall be able to communicate with it. The façade is going to be the wall between The User object from Account Management package and other classes like Comment and Post_it from Social Center package. An example is provided in the picture bellow, using the methods view() in the SocialCenterFacade.



4.3. Design Rationale

Rationale used in splitting classes into packages:

- **Coupling**: we have used provided class diagram and given sequence diagrams in the SRS document to determine which classes communicate the most and grouped them into distinctive packages to diminish the communication among packages.
- **Dependency**: The classes with high dependencies are composed together into one package.

5. COMPONENT DESIGN

Looking back at the class diagram provided to us by the requirements team, we noticed some missing – and quite essential- attributes and relations that the software won't work as intended without it. We choose multiple data structures to be used for the attributes, we settled on using Arrays, Array Lists, Stacks, and Priority Queues. Arrays are used for a list with a limited, known amount of entries. Array Lists are used for attributes that needs to be listed dynamically to accommodate for any potential increase in entries. Stacks are used to store entries in order, chronological order in fact. Priority Queues are used to order the objects with their natural order that'd be specified.

List of attributes:

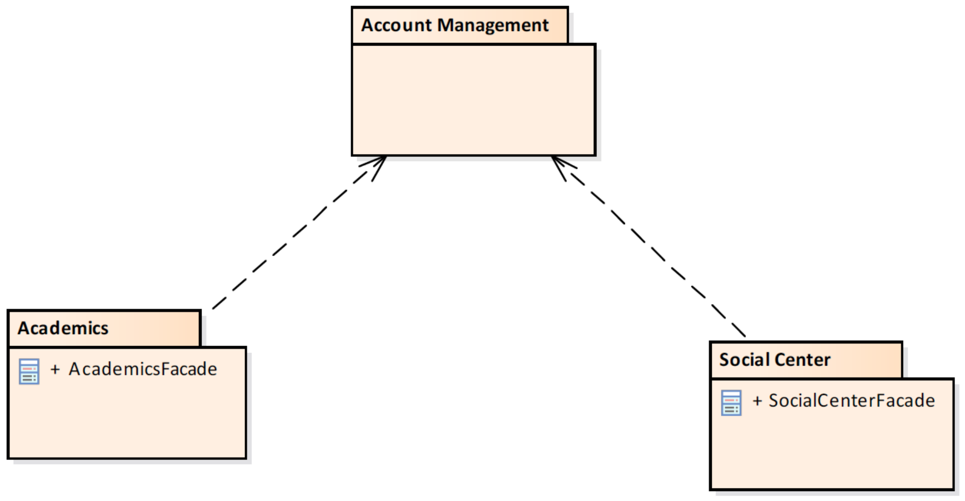
- Array Attributes:
    - .applicationList:RoomateApplication[n] in Student class
- ArrayList Attributes:
    - .studentList:ArrayList<Student> in Course class
    - .questionList:ArrayList<Question> in Course class
    - .reviewList:ArrayList<Review> in Course class
    - .friendList:ArrayList<Student> in Student class
    - .friendRequests:ArrayList<Student> in Student class
    - .courseList:ArrayList<Course> in Student class
    - .tagList:ArrayList<Tag> in Question class
- Stack Attributes:
    - .commentStack:Stack<Comment> in Review, Question, Answer, and Post_It classes
- Priority Queue Attributes:
    - .answerQueue:PriorityQueue<Answer> in Question class

List of relationships:

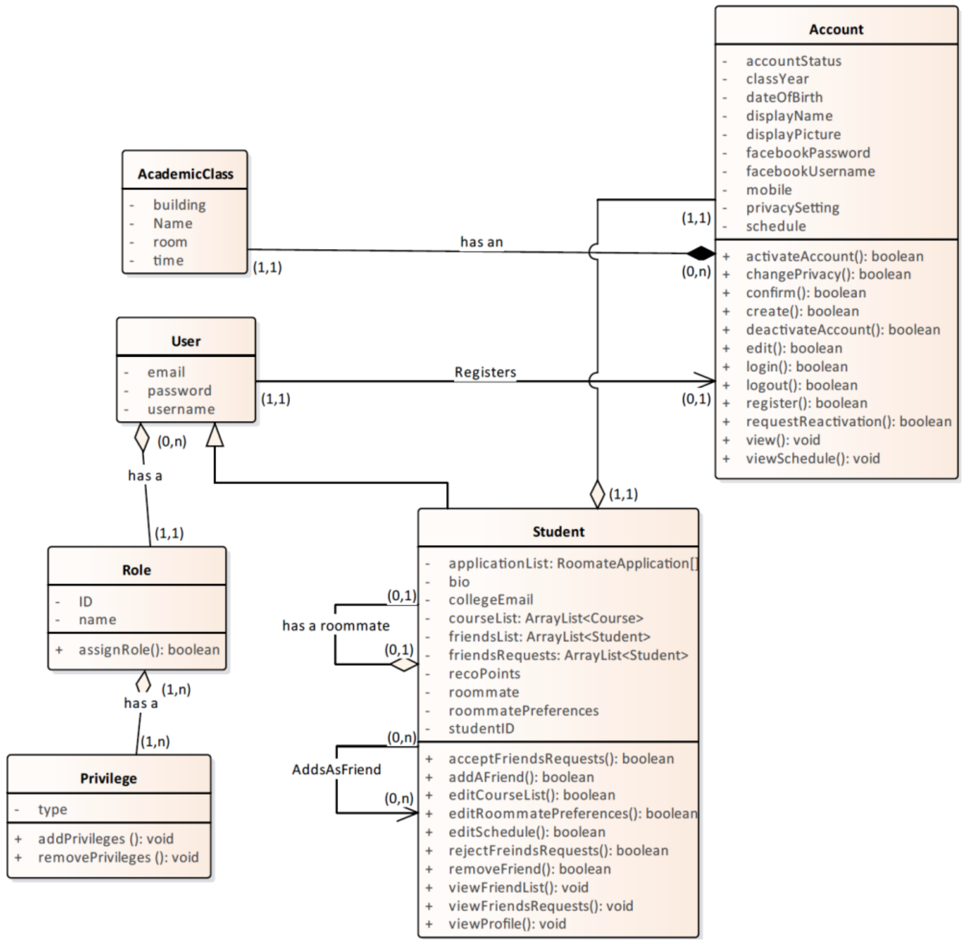
- A composition relationship added between the Comment class and the Question class since there exists a commentOn() method in the Question class with no relationship whatsoever.

Here are the pictures of the modifications to the class diagram:

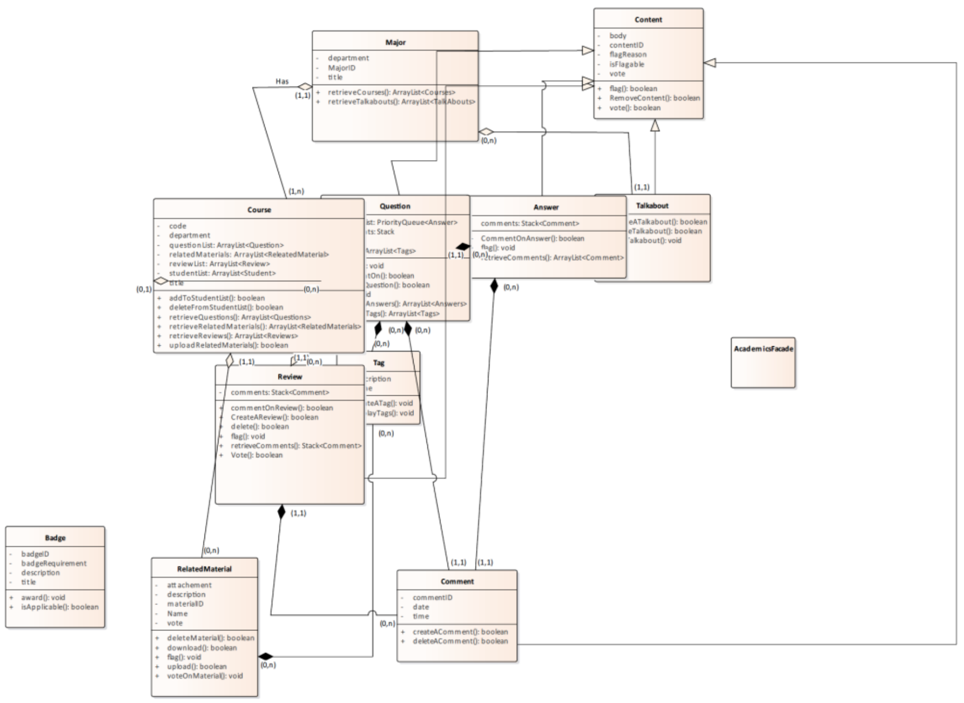
The packaged classes:



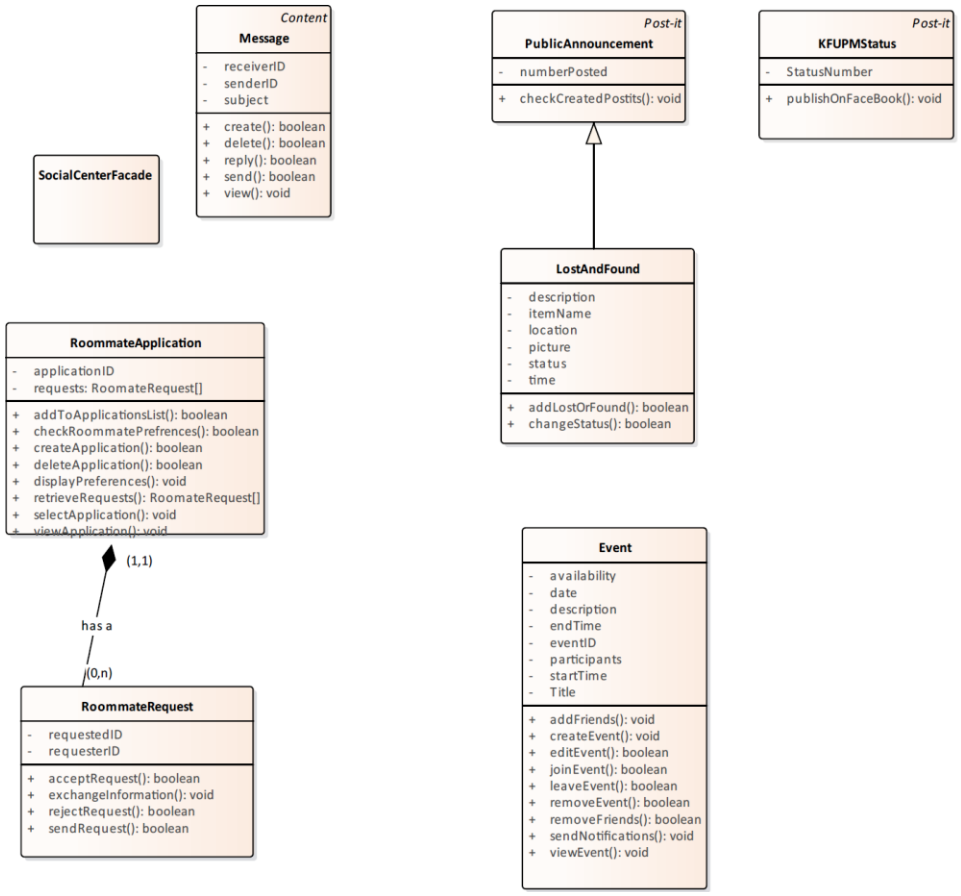
Account Management package:



Academics package:


Content
- body
- contentID
- flagReason
- isFlagable
- vote
+ flag(): boolean
+ RemoveContent(): boolean
+ vote(): boolean
Major
- department
- MajorID
- title
+ retrieveCourses(): ArrayList<Courses>
+ retrieveTalkabouts(): ArrayList<TalkAbouts>
Has
(1,1)
(0,n)
(1,n)
(1,1)
Course
- code
- department
- questionList: ArrayList<Question>
- relatedMaterials: ArrayList<ReleatedMaterial>
- reviewList: ArrayList<Review>
- studentList: ArrayList<Student>
- title
+ addToStudentList(): boolean
+ deleteFromStudentList(): boolean
+ retrieveQuestions(): ArrayList<Questions>
+ retrieveRelatedMaterials(): ArrayList<RelatedMaterials>
+ retrieveReviews(): ArrayList<Reviews>
+ uploadRelatedMaterials(): boolean
(0,1)
(0,n)
Question
ist: PriorityQueue<Answer>
ts: Stack
ArrayList<Tags>
: void
tOn(): boolean
Question(): boolean
d
Answers(): ArrayList<Answers>
Tags(): ArrayList<Tags>
Answer
comments: Stack<Comment>
CommentOnAnswer(): boolean
flag(): void
retrieveComments(): ArrayList<Comment>
(1,1)
(0,n)
Talkabout
eATalkabout(): boolean
eTalkabout(): boolean
alkabout(): void
(0,n)
(0,n)
(0,n)
(0,n)
Tag
cription
e
teATag(): void
layTags(): void
(0,n)
(1,1)
(1,1)
(0,n)
Review
- comments: Stack<Comment>
+ commentOnReview(): boolean
+ CreateAReview(): boolean
+ delete(): boolean
+ flag(): void
+ retrieveComments(): Stack<Comment>
+ Vote(): boolean
(1,1)
AcademicsFacade
Badge
- badgeID
- badgeRequirement
- description
- title
+ award(): void
+ isApplicable(): boolean
(0,n)
RelatedMaterial
- attachement
- description
- materialID
- Name
- vote
+ deleteMaterial(): boolean
+ download(): boolean
+ flag(): void
+ upload(): boolean
+ voteOnMaterial(): void
(0,n)
(1,1)
(1,1)
Comment
- commentID
- date
- time
+ createAComment(): boolean
+ deleteAComment(): boolean
(0,n)

Social Center package:



Pseudocode

Pseudocode will be written for 3 methods, one from each package.

Method 1: create from Account class

```
 IF(register with facebook){
    INPUT: user's facebook credentials, user's KFUPM email
    WHILE(! IsEmailValid() ){
        Display not valid email message;
        Display enter valid KFUPM email;
         }
    Send confirmation message to KFUPM email;
   OUTPUT: The visitor will be registered as a student
   }
 ELSE IF( register with user and password ){
    INPUT: username, password
    WHILE( usernameIsUsed){
    Display username is used message;
    Display enter another username;
    INPUT: username
}
   WHILE( passwordIsShort){
   Display password is short message;
   Display enter a password more than 8 characters message;
   INPUT: password
   }
   INPUT: KFUPM email
   WHILE(! IsEmailValid() ){
   Display not valid email message;
   Display enter valid KFUPM email;
  INPUT: user's KFUPM email
   }
  }
}
ENDIF
OUTPUT: The visitor will be registered as a student
```

Method 2: createAQuestion from class Question:

```
INPUT: Student's question text
IF(the student opens the general questions page){
     1. The student types the question text
     2. The student selects the related tags
     3. The student selects the page the question should be transferred to
}
ELSE IF(the student opens specific page (major/course) in which he wants to raise a question){
     1. The student types the question text
     2. The student selects the related tags
}
ENDIF
OUTPUT: The question is displayed on the proper page
```

Method 3: addToApplicationsList from RoomateApplication class

```
AddToApplicationsList(){
        Take application id;
        try:
```

```
        Requester.getApplicationList()[ add to the last cell ];
        Requested.getApplicationList()[ add to the last cell ];
catch IndexOutOfBoundsException:
        Display [Requester/Requested/Both] reached maximum application
limit
end
}
```

- REQUIREMENTS MATRIX

Three requirements were chosen to be cross referenced with the available components

Component / Requirement	The user shall be able to create an account	The user shall be able to create a question	The user shall be able to view roommate applications
Account Management Package	×		
Academics package		×	
Social Center package			×

Data structures inside listed components satisfies the need of any storage required by the requirements to be fulfilled.

- APPENDICES

  No appendices were needed in the development of this SDD.