CONCORDIA UNIVERSITY

COEN 6312

MODEL DRIVEN SOFTWARE ENGINEERING

DELIVERABLE 3: Class Diagram

Team : EmBridge

40053363 - Ishaan Sharma
40035779 - Kishor Tare
40043592 - Nidhi Patel
40048878 - Rashmi Narayan
40071278 - Yassine Jebbar

Guided By:
Dr. Wahab Hamou-Lhadj
Mohammad Reza Rejali

March 4, 2019

Contents

1 Domain Model **3**

2 Class Diagram **4**
2.1 Main Classes Description 4

3 Constraints **8**

List of Figures

1 Domain Diagram 3
2 Class Diagram 4

List of Tables

1 Team meetings and milestones. 2
2 Account 4
3 Message 5
4 Post 6
5 Group 7
6 PostReaction 7

Introduction

This document explains the **Class Diagram**, **Main Classes** along with their **attributes**, **methods** and **associations**. In addition, OCL has been utilized to constraint the class diagram.

Team Workflow

During this deliverable, the team often made regular meetings to discuss the tasks and align the doubts. The objective of each meeting is described in Table 1. :

Table 1: Team meetings and milestones.

Date	Discussion
Feb 17	Review on Deliverable Description and Requirements; Upcoming Meetings Schedules.
Feb 22	Detailed discussion on Deliverable 3; Class Diagram creation
Feb 24	Class diagram analysis
March 1	OCL constraints description
March 3	Final discussion and modifications on deliverable 3
March 5	Submission of Deliverable 3

1 Domain Model

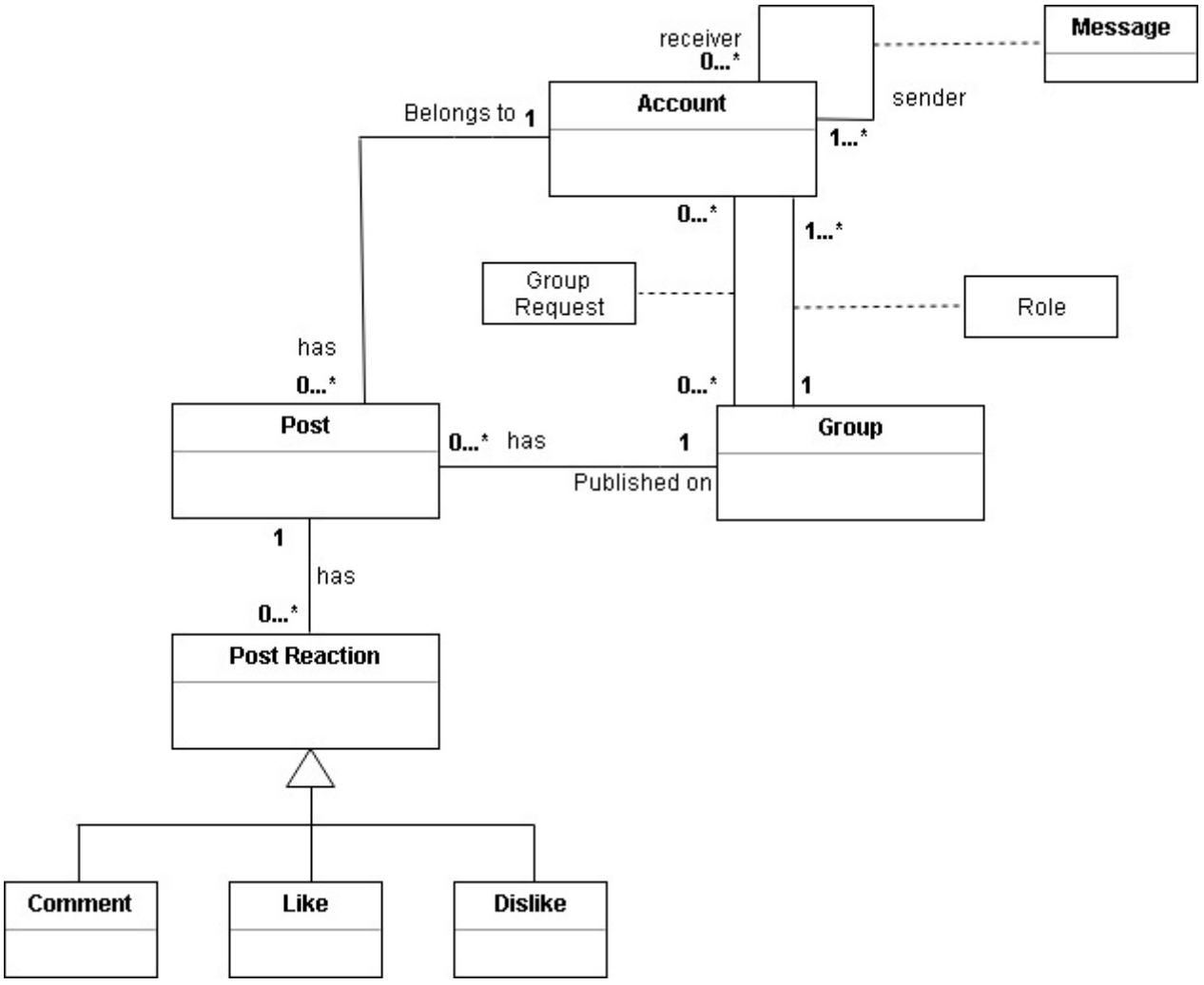

Figure 1: Domain Diagram

2 Class Diagram

2.1 Main Classes Description

Table 2: Account

Class Name	Account
Attributes	-userName: String -fullName: String -email: String -password: String
Methods	(All the methods are handled by ActivityController) +createPost(Post p): void - To create a new post +createPostGroup(Post p, Group g): void - To create post in a group +authenticate(Account a): void - To authenticate the account +reactToPost(Post p): void - To react to a post +sendMessage(Message m): void - To send a message +createGroup(): void - To create a new group +editProfile(Account a): void - To edit the Profile +ChangePassword(Account a):Void - To change password of account
Associated Classes	1. Post 2. Group (via Role and GroupRequest) 3. Account (Reflexive association via Message) 4. Message (Association Class) 5. Role (Association Class) 6. GroupRequest (Association Class) 7. PostReaction
Dessign Patterns	None

Figure 2: Class Diagram

build messages to
1
MessageBuilder
+createText(): void
+createImage(): void
+createVideo(): void
builds
AccountDao
+getAllAccounts(): ArrayList<Account>
+updateAccount(Account a): void
+deleteAccount(Account a): void
+addAccount(Account a): void
1..*
access
1..*
CommentFactory
+getComment(): Comment
1
execute reactions for
PostReactionHandler
+receiveReaction(Post p): void
+addReaction(Post p): void
uses
creates
Message
-message_id: int
-issueDate: DateTime
-content: ArrayList<ContentType)4>
+addContent(ContentType c): void
+showContent(): void
access
1..*
0..*
receiver
Account
-userName: String
-passowrd: String
-fullName: String
-email: String
1
belongs to
0..*
executes
«interface»
PostReaction
-reaction_id:int
+reactToPost(Post p): void
AuthenticationController
+validateLogin(Account a): boolean
1
authenticate accounts
create comments to
subject to
1..*
sender
1
belongs
ActivityController
+createPost(Post p): void
+createPostGroup(Post p, Group g): void
+autehnticate(Account a): void
+reactToPost(Post p): void
+sendMessage(Message m): void
+createGroup():void
+editProfile(Account a): void
+changePassword(Account a): void
initializes
CommentOnPost
-reaction_id: int
+reactToPost(Post p): void
LikePost
-reaction_id: int
+reactToPost(Post p): void
DislikePost
-reaction_id: int
+reactToPost(Post p): void
Role
-title:String
1
has
1..*
associated
1..*
contains
MessageDao
+getAllMessages(): ArrayList<Message>
+deleteMessage(Message m): void
+addMessage(Message m): void
subject to
0..*
1
created by
publishes
0..*
«interface»
Post
-post_id: int
-creationDate: DateTime
-content:ArrayList<ContenType>
-likeCount: int
-dislikeCount: int
+addContent(ContentType c): void
+showContent(): void
+showComments():void
builds
1..*1
PostBuilder
+createText(): void
+createImage(): void
+createVideo(): void
build posts to
1
1
create posts to
1
PostFactory
+getPost(): Post
1
build comments to
1
CommentBuilder
+createText(): void
+createImage(): void
+createVideo(): void
builds
1 1..*
«interface»
Comment
-comment_id: int
-creationDate: DateTime
-content:ArrayList<ContenType>
+addContent(ContentType c): void
+showContent(): void
access
1..* 1..*
CommentDao
+getAllComments(): ArrayList<Comment>
+deleteComment(Comment c): void
+addComment(Comment c): void
Group
-group_id: int
-groupName: String
-description: String
-creationDate: DateTime
+addAccount(Account a): void
+removeAccount(Account a): void
+removeAll(): void
+addAdmin(Account a): void
+removeAdmin(Account a): void
+acceptRequest(GroupRequest r): void
+declineRequest(GroupRequest r): void
1
operated
0..*
sends
0..*
shows
creates
1
executed on
access
1..*
1..*
access
1..*
1..*
GroupDao
+getAllGroups(): ArrayList<Group>
+updateGroup(Group g): void
+deleteGroup(Group g): void
+addGroup(Group g): void
published
1
GroupRequest
-request_id: int
-issueDate:DateTime
PostDao
+getAllPosts(): ArrayList<Post>
+updatePost(Post p): void
+deletePost(Post p): void
+addPost(Post p): void
RegularComment
-comment_id: int
-creationDate: DateTime
-content:ArrayList<ContenType>
+addContent(ContentType c): void
+showContent(): void
TaggedComment
-comment_id: int
-creationDate: DateTime
-content:ArrayList<ContenType>
+addContent(ContentType c): void
+showContent(): void
RegularPost
-post_id: int
-creationDate: DateTime
-content:ArrayList<ContenType>
-likeCount: int
-dislikeCount: int
+addContent(ContentType c): void
+showContent(): void
+showComments():void
TaggedPost
-post_id: int
-creationDate: DateTime
-content:ArrayList<ContenType>
-likeCount: int
-dislikeCount: int
+addContent(ContentType c): void
+showContent(): void
+showComments(): void
uses
uses
«interface»
ContentType
+createContent(): void
uses
Image
+createContent(): void
Video
+createContent(): void
Text
+createContent(): void

Table 3: Message

Class Name	Message
Attributes	-message_id: int -issueDate:DateTime -Content:ArrayList <Content Type>
Methods	+addContent(ContentType c):void - To add content in a message +ShowContent():voidb- To display message content
Associated Classes	1. Account(Reflexive Association) 2. MessageDao
Design Patterns	Builder Pattern DAO Pattern

Table 4: Post

Class Name	Post
Attributes	-post_id: int -creationDate: DateTime -Content:ArrayList<Content Type> -likeCount: Int -dislikeCount:Int
Methods	+addContent(ContentType c):void - To add content such as image, video, text to post +ShowComments():void - To display comments of a post +showContent():void - To show all contents of a post
Child Classes	1. RegularPost 2. TaggedPost
Associated Classes	1. Group 2. Account 3. PostReaction 4. PostBuilder 5. PostDao 6. PostFactory
Design Patterns	1. Builder Pattern 2. Factory Pattern 3. DAO

Table 5: Group

Class Name	Group
Attributes	-group_id: int -groupName:String -Description:String -creationDate:DateTime
Methods	+addAccount(Account a):void - To add an account to group +removeAccount(Account a):void - To remove an account from group +removeAll(): void - To remove everyone from group +addAdmin(Account a): void - To add new admin to group +removeAdmin(Account a): void - To remove an admin from group +acceptRequest(GroupRequest r): void - To accept a group request +declineRequest(GroupRequest r): void - To decline a group request
Associated Classes	1. Account 2. Role (Association class) 3. GroupRequest(Association Class) 4. Post 5. GroupDao
Design Patterns	DAO

Table 6: PostReaction

Class Name	PostReaction
Attributes	reaction_id: int
Methods	+reacttoPost(post p):void - To react to a Post
Child Classes	1. LikePost 2. DislikePost 3. CommentOnPost
Associated Class	1. Account 2. Post 3. PostReactionHandler
Design Patterns	Command Pattern

3 Constraints

A list of constraints that apply to the class diagram are prepared using OCL as follows:

1. ***Each account should have a unique username***

```
Context Account
Inv: allinstances() -> forAll(a1,a2: Account| a1<>a2 implies
a1.username <> a2.username)
```

2. ***Only admins of a group can promote members of the group as admins using AddAdmin(Account: a) method***

```
Context Group :: AddAdmin(Account: a)
Pre:    self.contains.title = 'admin' AND
        self.contains->select(title='member').associated->includes(a) AND
        self.contains->select(title='admin').associated->excludes(a)
Post:   self.contains->select(title='admin').associated->includes(a)
```

3. ***An account holder cannot send group request to the same group twice***

```
Context Account
Inv: self.GroupRequest -> forAll (g1, g2 : Group |
        g1<>g2 implies g1.group_id<>g2.group_id)
```

4. ***A message should be between 2 different accounts. i.e., a user cannot message himself***

```
Context Account:
Inv:self.receiver -> forAll(a1: Account| self.username <> a1.username)
```

5. ***AddAccount(Account: a) adds an account to the group. An account can be added only if it was not already added. Only Admin of the group can add an account to the group***

```
Context Group::AddAccount(Account: a)
Pre:    self.contains.associated->excludes(a) AND
        self.contains.title = 'admin'
Post:   self.contains.associated->includes(a)
```

6. ***RemoveAccount(Account: a) removes an account from the group. Only Admin of the group can remove account from group***

```
Context Group::RemoveAccount(Account: a)
Pre:    self.contains.associated->includes(a) AND
        self.contains.title = 'admin'
Post:   self.contains.associated ->excludes(a)
```

7. ***An account holder can post in a group only if he is a member of the group***

```
Context Account
Inv: self.has.operated -> includesAll(self.publishes.published)
```

8. ***RemoveAll() removes all account from the group. Only Admin of the group can remove all accounts from the group***

```
Context Group::RemoveAll()
Pre:    self.contains.associated->NotEmpty() AND
        self.contains.title = 'admin'
Post:   self.contains.associated ->isEmpty()
```